Android

مباحث ویژه

برنامه نویسی اندروید

مباحث ویژه

- برنامه نویسی اندروید

By:
Sasan Harifi

سرفصل مطالب

نحوه ارزشیابی

تمرین و پروژه: ۳ نمره

کوئیز: ۱ نمره

میان ترم: ---

پایان ترم: ۱۶ نمره

نمره اضافه: در صورتی که پروژه در حد یک برنامه کاربردی یا تجاری باشد، با قرار دادن آن در یکی از مارکت‌ها، ۲ نمره اضافه بر ۲۰ نمره در نظر گرفته خواهد شد. (به صورت گروه های دو نفره و یا انفرادی)

نیازمندی‌های درس

- با توجه به محدودیت‌های کارگاه، لپ تاپ مورد نیاز است.
- اندروید استودیو نسخه ۲.۱.۲ به بعد
  - نسخه فول به همراه SDK (Software Development Kit)
  - پیش نیازهای نصب اندروید استودیو مانند JDK (Java Development Kit) و ...
- در صورت نیاز IntelliJ IDEA برای برنامه نویسی جاوا
- دانلود و نصب اپلیکیشن اطلاع رسانی از سایت یا از طریق اسکن QR Code روبرو (نسخه اندروید):

0 مقدمه

آشنایی با اندروید

- سیستم عامل اپن سورس و استفاده از هسته لینوکس
- کاربردهای اندروید
  - گوشی های هوشمند
  - کامپیوترهای دستی
  - تبلت ها و ...
- ارائه شده توسط گوگل
  - نسخه اولیه، دارای ۱۲ میلیون خط کد
    - ۳ میلیون XML
    - حدود ۸ میلیون خط C
    - حدود ۱ میلیون خط جاوا

مزایا و ویژگی های عمومی

- رابط کاربری غنی و کاربر پسند
- دارای توابع غنی
  - مثلا توابع مدیریت تماس های تلفنی
- دارای برنامه های کاربردی و سودمند برای کاربران نهایی
- دارای کتابخانه های متعدد برای راحتی برنامه نویسی و توسعه
- پشتیبانی مناسب از چند رسانه ای
- و دیگر موارد...

ویژگی های خاص اندروید

- به دلیل اپن سورس بودن به راحتی قابل دستکاری است.
- پشتیبانی از SQLite برای ذخیره داده ها
- پشتیبانی از سخت افزارهای مختلف
  - دوربین، قطب نما، شتاب سنج، وای فای، GPS، 3G، 4G، EDGE، Bluetooth، NFC و ...
- بهینه شده برای کار با سخت افزارهایی با ظرفیت حافظه و باتری پایین
- گرافیک و صدای قابل قبول
  - پشتیبانی از انیمیشن های دوبعدی مبتنی بر فلش و وکتوری
  - پشتیبانی از انیمیشن های سه بعدی مبتنی بر OpenGL

ویژگی های خاص اندروید

- پشتیبانی از انواع فرمت های صوتی و تصویری
  - صوتی: MIDI، AMR، OGG، AAC، MP3، WAV
  - تصویری: JPG، PNG، GIF، BMP
  - دیگر فرمت ها: MPEG4، Adobe Flash
- پشتیبانی از سخت افزارهای جانبی مانند کیبورد، صفحات لمسی و ...
- دارای مرورگر داخلی مبتنی بر موتور اپن سورس WebKit
- مدیریت و ایزوله کردن برنامه ها با کمک لایه های امنیتی و دسترسی ها
- استفاده از مترجم Dalvik

تمرین: در مورد مترجم Dalvik و موتور مرورگر WebKit تحقیق کنید!

انواع برنامه نویسی برای اندروید

- سه شیوه برنامه نویسی برای اندروید وجود دارد:
  - شیوه اول: برنامه نویسی Native که در آن با استفاده از زبان جاوا برای گوشی های اندرویدی برنامه می نویسند. در این روش به تمام ویژگی های سخت افزاری دسترسی وجود دارد.
- مزایای شیوه Native:
  - به دلیل کامپایل شدن، برنامه دارای سرعت بالایی است.
  - دسترسی به تمام امکانات سیستم عامل و سخت افزار.
  - پشتیبانی از API (Application Programming Interface) های مختلف در صورت نیاز.
- معایب شیوه Native:
  - هزینه بالا، به دلیل زمان توسعه زیاد
  - دوباره نویسی کدهای برنامه برای پلتفرم های مختلف

انواع برنامه نویسی برای اندروید

- سه شیوه برنامه نویسی برای اندروید وجود دارد:
- شیوه دوم: برنامه نویسی Mobile Web App است که در آن یک بار کد نوشته می شود و می توان روی پلتفرم های مختلف از آن استفاده کرد. این شیوه در واقع نرم افزار نیست، بلکه وب سایتی است که از پلتفرم های مختلف می توان به آن دسترسی داشت.

- مزایای روش Web App:
  - نوشتن کد، یکبار و استفاده در بسترهای مختلف و رفع مشکلات به صورت آسان
  - مستقل از سیستم عامل مقصد

- معایب روش Web App:
  - عدم دسترسی به سخت افزار و وابسته به مرورگر بودن
  - عدم انتشار مستقیم در مارکت ها

انواع برنامه نویسی برای اندروید

- سه شیوه برنامه نویسی برای اندروید وجود دارد:
- شیوه سوم: برنامه نویسی Hybrid است که مانند شیوه دوم یعنی Web App است با این تفاوت که توسط WebView در اندروید و یا UIWebView در iOS محتوای نرم‌افزار به کاربر نمایش داده خواهد شد.
- معمولا توسط HTML، CSS، JavaScript نوشته می شوند سپس تبدیل به نرم افزار قابل اجرا در گوشی هوشمند می شوند.

- مزایای روش Hybrid:
  - نوشتن کد، یکبار و استفاده در بسترهای مختلف

- معایب روش Hybrid:
  - بازدهی کمتر نسبت به Native (به دلیل استفاده از مرورگر)

مروری بر زبان برنامه‌نویسی جاوا

متغیرها در جاوا

- متغیر، اساسی ترین واحد ذخیره سازی است. یک متغیر به وسیله ترکیبی از یک شناسه، یک نوع و یک مقدار اولیه (اختیاری) تعریف می شود.

- اعلان یک متغیر:

```
type_identifier  identifier;
```

متغیرها در جاوا

- انواع متغیر:

نوع	فضای حافظه اختصاص داده شده
int	4 bytes, عدد صحیح
char	2 bytes, یک کاراکتر
byte	1 byte, عدد صحیح
float	4 bytes, عدد اعشاری
double	8 bytes, عدد اعشاری
short int	2 bytes, عدد صحیح
long int	8 bytes, عدد صحیح
String	N byte
void	برگشت نوع داده ای استفاده شده در آن اگر return مورد نیاز نباشد.
boolean	True, false

متغیرها در جاوا

- مثال تعریف متغیر:

- int a;
- int a,b,c;
- double pi=3.14;
- char x='x';

int d=3, e, f=5;

byte z = 22;

برای نوشتن توضیحات یا کامنت مانند دیگر زبان ها از:

- // برای نوشتن توضیح تک سطری
- /* ---------- */ برای نوشتن توضیح چند سطری استفاده می شود.

تغییر نوع برای استفاده در مقصد

- مثال:

```
class Conversion {
public static void main(String args[] ){
        byte b;
        int i = 257;
        double d = 323.142;
        System.out.println("\n Conversion of int to byte:");
        b = (byte)i;
        System.out.println("i and b" + i + "  " + b);
        System.out.println("\n Conversion of double to int:");
        i =(int)d;
        System.out.println("d and i " + d + "  " + i);
        System.out.println("\n Conversion of double to byte:");
        b =(byte)d;
        System.out.println("d and b " + d + "  " + b);
        }
}
```

خروجی:

Conversion of int to byte:
i and b 257 1

Conversion of double to int:
d and i 323.142 323

Conversion of double to byte:
d and b 323.142 67

متغیر char

- برخلاف C و C++ که یک بایت در نظر می گیرد، ولی مشابه C# است که دو بایت فضا در نظر می گیرد.
- استاندارد unicode برای معرفی کاراکترها استفاده می کند.
- استاندارد unicode کدهای ASCII را نیز شامل می شود.
- دامنه از ۰ تا ۶۵۵۳۶ است و در آن مقدار منفی وجود ندارد.

مثال اول از استفاده متغیر char

- مثال:

```
class CharDemo {
        public static void main(String args[] ){
        char ch1, ch2;
        ch1 = 88;   // code for X
        ch2 = 'Y';
        System.out.print("ch1 and ch2 :");
        System.out.println(ch1 + " " + ch2);
        }
}
```

خروجی:

```
ch1 and ch2 :X Y
```

مثال دوم از استفاده متغیر char

- مثال:

```
class CharDemo2 {
        public static void main(String args[] ){
        char ch1;
        ch1 = 'X';
        System.out.println("ch1 contains " + ch1);
        ch1++;
        System.out.println("ch1 is now " + ch1);
        }
}
```

خروجی:

```
ch1 contains X
ch1 is now Y
```

متغیر boolean

- فقط دو مقدار true و false می تواند داشته باشد.
- مقادیر true و false هرگز به رقم تبدیل نمی شوند.
- در جاوا لفظ true مساوی یک نیست و لفظ false معادل صفر نیست.
- فقط به متغیرهای اعلان شده به عنوان boolean منتسب می شوند.

مثالی از استفاده متغیر boolean

- مثال:

```
class BoolTest {
        public static void main(String args[] ){
        boolean b;
        b = false;
        System.out.println("b is " + b);
        b = true;
        System.out.println("b is " + b);
        if (b)
                System.out.println("This is executed.");
        b = false;
        if (b)
                System.out.println("This is not executed.");
        System.out.println("10 > 9 is " +(10 > 9));
        }
}
```

خروجی:

```
b is false
b is true
This is executed.
10>9 is true
```

وجود پرانتز باعث می شود نتیجه مقایسه برگشت داده شود.

عملگرها در جاوا (مشابه با خانواده C)

- در چهار طبقه بندی قرار می گیرند:
  - حسابی، بیتی، مقایسه ای، منطقی

+ - * /
% ++ --
+= -= *= /=
%=

عملوند مربوط به عملگرها باید عددی باشد. Boolean مجاز نیست. Char مجاز است چون زیر مجموعه int است.

~ & | ^
>> <<

one	two	&	|	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

>
<
>=
<=
==
!=

&& || !

one	two	&&	||
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

مثالی از استفاده از عملگرها

- مثال:

```
class ByteShift {
        public static void main(String args[] ){
        byte a = 64, b;
        int i;
        i = a << 2;
        b =(byte(a) << 2);
        System.out.println("Original value of a:" + a);
        System.out.println("i and b :" + i + "   " + b);
        }
}
```

خروجی:

```
original value of a:64
i and b :256  0
```

عملگر انتساب

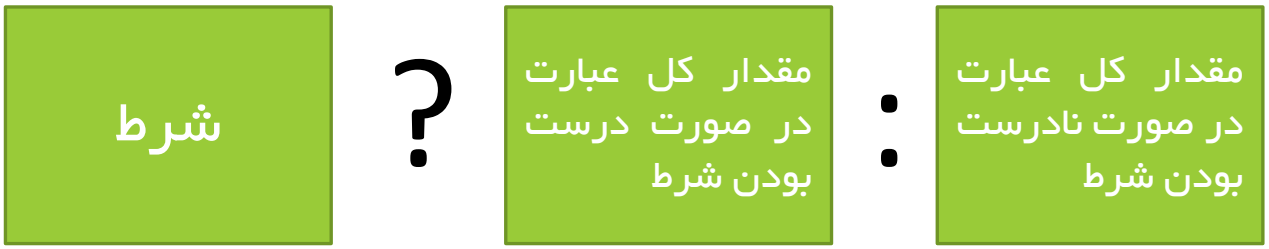
- شکل کلی: var = expression;
- مثال:

```
int x, y, z;
x = y = z = 100;
```

- مثال فوق، مقدار ۱۰۰ را با استفاده از یک دستور در متغیرهای x، y و z قرار می دهد.
- عملگر انتساب عملگری است که مقدار سمت راست خود را جذب می کند.
- چون z=100 است، پس y نیز مقدار سمت راست خود، یعنی مقدار z را جذب می کند. و همینطور x نیز به همچنین.

عملگر ؟ (عملگر سه تایی، شرطی)

- شکل کلی:



مثالی از استفاده از عملگر سه تایی

- مثال:

```
class Ternary {
        public static void main(String args[] ){
        int i, k;
        i = 10;
        k = i < 0 ?- i : i;
        System.out.print("Absolute value of ");
        System.out.println(i + " is " + k);
        }
}
```

خروجی:

value of 10 is 10

اگر نوشته شود:

k = i > 0 ? - i : i;

خروجی برابر است با:

value of 10 is -10

دستور انتخاب if

- دستور if

```
if (condition)
        statement 1;
else
        statement 2;
```

- مثال:

مثال ۱

```
int a, b;
// ...
// ...
if (a < b)
        a = 0;
else
        b = 0;
```

مثال ۲

```
boolean dataAvailable;
// ...
// ...
if (dataAvailable)
        ProcessData();
else
        waitForMoreData();
```

مثال ۳

```
int bytesAvailable;
//...
if (bytesAvailable > 0){
        ProcessData();
        bytesAvailable = n;
}
else
        waitForMoreData();
```

دستور انتخاب if

- دستور if

```
if (condition)
        statement 1;
else
        statement 2;
```

- مثال:

مثال ۵

```
int bytesAvailable;
//...
if (bytesAvailable > 0){
        ProcessData();
        bytesAvailable = n;
}
else {
        waitForMoreData();
        bytesAvailable = n;
}
```

مثال ۴

```
int bytesAvailable;
//...
if (bytesAvailable > 0){
        ProcessData();
        bytesAvailable = n;
}
else
        waitForMoreData();
bytesAvailable = n;
```

دستور انتخاب if

- دستور if تو در تو

```
if (condition) {
        if (condition)
                statement 0;
        else
                statement 1;
        }
else
        statement 2;
```

- نردبان if تو در تو

```
if (condition)
        statement;
else if (condition)
        statement;
else if (condition)
        statement;
. . .
else
        statement;
```

مثال

```
if (i == 10){
        if (j < 20 )
                a = b;
        if (k > 100)
                c = d;
        else
                a = c;
}
else
        a = d;
```

مثالی از دستور انتخاب if

- مثال:

```
public class IfElse {
        public static void main(String args[] ){
        int month = 4; // April
        String season;
        if (month == 12 || month == 1 || month == 2)
        season = "Winter";
        else if (month == 3 || month == 4 || month == 5)
        season = "Spring";
        else if (month == 6 || month == 7 || month == 8)
        season = "Summer";
        else if (month == 9 || month == 10 || month == 11)
        season = "Autumn";
        else
        season = "Error";
        System.out.println("April is in the"+ season +".");
        }
}
```

خروجی:

April is in the Spring.

for حلقه

- دستور for

```
for (initialization ; condition ; iteration){
// body
}
```

- مثال:

```
public class ForTick {
	public static void main(String args[] ){
	int n;
	for (n=10 ; n>0 ; n--)
		System.out.println("tick" + n);
	}
}
```

خروجی:

```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

مثال اول از حلقه for

- مثال:

```
class LifeTime {
        public static void main(String args[] ){
        int x;
        for(x = 0; x < 3; x++){
                int y = -1;
                System.out.println("y is :" + y);
                y = 100;
                System.out.println("y is now :" + y);
                }
        }
}
```

خروجی:

```
y is: -1
y is now:100
y is: -1
y is now:100
y is: -1
y is now:100
```

مثال دوم از حلقه for

```
public class FindPrime {
        public static void main(String args[] ){
        int num;
        boolean isPrime = true;
        num = 14;
        for (int i=2 ; i < num/2 ; i ++){
                if ((num % i)== 0){
                        isPrime = false;
                        break;
                }
        }
        if (isPrime)
        System.out.println("Prime");
        else
        System.out.println("Not Prime");
        }
}
```

- مثال:

خروجی:

Not Prime

مثالی از for و بلاک ها

- مثال:

```
class BlockTest {
        public static void main(String args[] ){
        int x, y;
        y = 20;
        for(x = 0; x<5; x++ ){
                System.out.println("This is x :" + x);
                System.out.println("This is y :" + y);
                y = y - 2;
                }
        }
}
```

خروجی:

```
This is x:0
This is y:20
This is x:1
This is y:18
This is x:2
This is y:16
This is x:3
This is y:14
This is x:4
This is y:12
```

چند نکته در مورد حلقه for

- حلقه for می تواند با if ترکیب شود.
- حلقه for می تواند با حلقه for ترکیب شود (حلقه تو در تو)

```
public class Nested{
	public static void main(String args[] ){
	int i, j;
	for (i=0 ; i<9 ; i++){
		for(j=i ; j<9 ; j++)
			System.out.print("*");
		System.out.println();
		}
	}
}
```

خروجی:

```
*********
********
*******
******
*****
****
***
**
*
```

دستور انتخاب switch

- دستور switch

```
switch (expression){
case value1:
        // statement
        break;
case value2:
        // statement
        break;
…
case valueN:
        // statement
        break;
default:
        // default statement
}
```

مثال اول از دستور انتخاب switch

```
public class SampleSwitch {
        public static void main(String args[] ){
        for (int i=0; i<6; i++)
                switch(i){
                        case 0:
                        System.out.println("i is zero.");
                        break;
                        case 1:
                        System.out.println("i is one.");
                        break;
                        case 2:
                        System.out.println("i is two.");
                        break;
                        default:
                        System.out.println("i is greater than 2.");
                        }
        }
}
```

- مثال:

خروجی:

```
i is zero.
i is one.
i is two.
i is greater than 2.
i is greater than 2.
i is greater than 2.
```

مثال دوم از دستور انتخاب switch

```
public class MissingBreak {
        public static void main(String args[] ){
        for (int i=0; i<8; i++)
                switch(i ){
                        case 0:
                        case 1:
                        case 2:
                        System.out.println("i is less than 3");
                        break;
                        case 3:
                        case 4:
                        System.out.println("i is less than 5");
                        break;
                        default:
                        System.out.println("i is 5 or more");
                        }
        }
}
```

- مثال:

خروجی:

```
i is less than 3
i is less than 3
i is less than 3
i is less than 5
i is less than 5
i is 5 or more
i is 5 or more
i is 5 or more
```

دستور انتخاب switch تو در تو

```
switch (count ){
        case 1:
                switch (target){
                        case 0:
                        System.out.println("target is zero");
                        break;
                        case 1:
                        System.out.println("target is one");
                        break;
                }
                break;
        case 2 :
                //...
```

- دستور switch تو در تو

چند نکته:

- برخلاف if در switch مقایسه ها فقط از نوع برابری است.
- مقادیر تکراری برای caseها مجاز نیست.
- در صورت موفقیت آمیز بودن مقایسه برابری، هیچ عملیات مقایسه ای دیگری انجام نمی شود.

دستورات تکرار – حلقه while

- دستور while

```
while (condition){
// body of loop
}
```

- مثال:

```
public class While {
	public static void main(String args[] ){
	int n = 10;
	while (n > 0){
		System.out.println("tick " + n);
		n--;
		}
	}
}
```

خروجی:

```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

دستورات تکرار – حلقه do while

- دستور do while

```
do{
// body of loop
} while (condition);
```

(اجرای بدنه حلقه حداقل یک مرتبه)

- مثال:

```
public class DoWhile {
	public static void main(String args[] ){
	int n = 10;
	do {
		System.out.println("tick "+ n);
		n--;
	} while (n > 0);
	}
}
```

خروجی:

```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

مثال: دو متغیر جدا با یک اسم (مجاز نیست)

- مثال:

```
class ScopeErr {
        public static void main(String args[] ){
        int bar = 1;
                {
                int bar = 2;
                }
        }
}
```

خروجی:

Compile-time error -- bar already defined!

نکته: این شیوه تعریف متغیر با یک اسم با استفاده از بلاک در زبان های خانواده C مجاز است.

آرایه ها در جاوا

- تعریف آرایه

```
array_type  array_name [];
array_name = new array_type [size];
```

معادل

```
array_type  array_name[] = new array_type [size];
```

- مثال:

```
int month_days[];
month_days = new int[12];
```

معادل

```
int month_days[] = new int[12];
```

- مقداردهی به آرایه

```
month_days[5] = 30;
```

- فراخوانی خانه ای از آرایه

```
System.out.println(month_days[5]);
```

مثال اول از آرایه

```
public class Array {
        public static void main(String args[] ){
        int month_days[];
        month_days = new int[12];
        month_days [0] = 31;
        month_days [1] = 28;
        month_days [2] = 31;
        month_days [3] = 30;
        month_days [4] = 31;
        month_days [5] = 30;
        month_days [6] = 31;
        month_days [7] = 31;
        month_days [8] = 30;
        month_days [9] = 31;
        month_days [10] = 30;
        month_days [11] = 31;
        System.out.println("April has " + month_days[3] + " days.");
}       }
```

- مثال:

خروجی:

April has 30 days.

مثال دوم از آرایه

- مثال:

```
public class AutoArray {
        public static void main(String args[] ){
                int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,31 };
                System.out.println("April has " + month_days[3] + " days.");
        }
}
```

خروجی:

April has 30 days.

مثال سوم از آرایه

- مثال:

```
public class Average {
	public static void main(String args[] ){
		double nums[] = {10.1, 11.2, 12.3, 13.4, 14.5};
		double result = 0;
		int i;
		for(i=0; i<5; i++)
			result = result + nums[i];
		System.out.println("Average is " + result / 5);
	}
}
```

خروجی:

Average is 12.2999

آرایه چند بعدی

- مثال: `int twoD[][] = new int[4][5];`

```
public class TwoDArray {
	public static void main(String args[]){
		int twoD[][] = new int[4][5];
		int i, j,k = 0;
		for(i=0; i<4; i++)
			for(j=0; j<5; j++ ){
				twoD[i][j] = k;
				k++;
			}
		for(i=0; i<4; i++ ){
			for(j=0; j<5; j++)
			System.out.print(twoD[i][j] + " ");
			System.out.println();
		}
	}
}
```

خروجی:

```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

آرایه چند بعدی

- نکته: هنگام تخصیص حافظه به آرایه چند بعدی، کافی است فقط برای بعد اول، حافظه تخصیص دهیم. بقیه ابعاد را می توانیم جداگانه حافظه تخصیص دهیم.

```
int twoD[][] = new int[4][];
twoD[0] = new int[5];
twoD[1] = new int[5];
twoD[2] = new int[5];
twoD[3] = new int[5];
```

مثال دیگری از آرایه چند بعدی

- مثال:

```
public class Matrix {
	public static void main(String args[] ){
		double m[][] = {
		{ 0.0, 1.0, 2.0, 3.0 },
		{ 0.1, 1.1, 2.1, 3.1 },
		{ 0.2, 1.2, 2.2, 3.2 },
		{ 0.3, 1.3, 2.3, 3.3 }
		};
		int i, j;
		for(i=0; i<4; i++ ){
			for(j=0; j<4; j++)
			System.out.print(m[i][j] + " ");
			System.out.println();
		}
	}
}
```

خروجی:

```
0.0  1.0  2.0  3.0
0.1  1.1  2.1  3.1
0.2  1.2  2.2  3.2
0.3  1.3  2.3  3.3
```

مثالی از آرایه سه بعدی

```
public class threeDmatrix {
        public static void main(String args[] ){
                int threeD[][][] = new int[3][4][5];
                int i, j,k;
                for(i=0; i<3; i++)
                        for(j=0; j<4; j++)
                                for(k=0; k<5; k++)
                                        threeD[i][j][k] = i * j * k;
                for(i=0; i<3; i++ ){
                        for(j=0; j<4; j++ ){
                                for(k=0; k<5; k++)
                                        System.out.print(threeD[i][j][k] + " ");
                                System.out.println();
                        }
                System.out.println();
                }
        }
}
```

- مثال:

خروجی:

```
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12

0 0 0 0 0
0 2 4 6 8
0 4 8 12 16
0 6 12 18 24
```

طول آرایه (تابع length)

- مثال:

```
public class Length {
	public static void main(String args[] ){
		int a1[] = new int [10];
		int a2[] = {3, 5, 7, 1, 8, 99, 44, -10};
		int a3[] = {4, 3, 2, 1};
		System.out.println("length of a1 is " + a1.length());
		System.out.println("length of a2 is " + a2.length());
		System.out.println("length of a3 is " + a3.length());
	}
}
```

خروجی:

```
length of a1 is 10
length of a2 is 8
length of a3 is 4
```

اعلان نمودن اشیاء

- بدست آوردن اشیاء از یک کلاس، دو مرحله ای است.
  - مرحله اول: اعلان یک متغیر از نوع همان کلاس.
  - مرحله دوم: یک کپی فیزیکی از شیء به متغیر منتسب کنیم.
- عملیات فوق با استفاده از new انجام می شود.
- مثال:

```
Box mybox;
mybox = new Box();
```

معادل

```
Box mybox = new Box();
```

اعلان نمودن اشیاء

- خط اول:
  - mybox را به عنوان ارجاعی از شیء از نوع box اعلان می کند.
- پس از اجرای این خط، mybox محتوی تهی (null) خواهد داشت. یعنی شیء به صورت کامل وجود ندارد.
- استفاده از mybox بدون خط دوم سبب خطای کامپایل می شود.

```
Box mybox;
mybox = new Box();
```

اعلان نمودن اشیاء

- خط دوم:
  - یک شیء واقعی را تخصیص داده و یک ارجاع از آن به mybox انجام می دهد.
- پس از اجرای خط دوم، می توان از mybox به عنوان یک شیء box استفاده کرد.

```
Box mybox;
mybox = new Box();
```

نگاه دقیق تر به new

- شکل کلی: class_var = new class_name();
- class_var یک متغیر از نوع کلاسی است که ایجاد کرده ایم.
- class_name نام کلاسی است که می خواهیم معرفی کنیم.
  - نام کلاس که بعد از آن پرانتزها آمده اند، مشخص کننده سازنده (constructor) کلاس است.
  - سازنده تعریف می کند که وقتی یک شیء از یک کلاس ایجاد شود، چه خواهد شد.
- از new برای اشیاء استفاده می شود.
  - اعداد صحیح و کاراکتر، اشیاء نیستند. متغیر عادی هستند.
  - new حافظه را برای یک شیء طی زمان اجرا، تخصیص می دهد.

معرفی class در جاوا

- کلاس، یک ساختار منطقی است که تمامیت زبان جاوا بر آن استوار است.
- کلاس، شکل دهنده اساس برنامه نویسی شیء گرا در جاوا است.
- برای پیاده سازی مفاهیم در برنامه جاوا، باید داخل یک کلاس قرار گیرد.
- کلاس یک نوع جدید داده را تعریف می کند.
  - هر بار که این نوع تعریف شود، می توان از آن برای ایجاد اشیایی از همان نوع استفاده کرد.
  - کلاس قالبی از یک شیء است. شیء نمونه ای از یک کلاس است.

مثالی از کلاس

- مثالی از تعریف یک کلاس ساده با نام box

```
public class Box {
        double width;
        double height;
        double depth;
}
```

- فراخوانی و تخصیص حافظه به شیء

```
Box mybox = new Box();
```

- مقدار دهی یک کپی از یک متغیر داخل شیء mybox

```
mybox.width = 100;
```

مثال اول از کلاس

- مثال:

```
public class Box {
double width;
double height;
double depth;
}
```

```
public class BoxDemo {
        public static void main(String args[]){
        Box mybox = new Box();
        double vol;
        mybox.width = 10;
        mybox.height = 20;
        mybox.depth = 15;
        vol = mybox.width * mybox.height * mybox.depth;
        System.out.println("Volume is " + vol);
        }
}
```

خروجی:

Volume is 3000

مثال دوم از کلاس

```
public class Box {
double width;
double height;
double depth; }
```

```
public class BoxDemo2 {
        public static void main(String args[] ){
        Box mybox1 = new Box();
        Box mybox2 = new Box();
        double vol;
        mybox1.width = 10;
        mybox1.height = 20;
        mybox1.depth = 15;
        mybox2.width = 3;
        mybox2.height = 6;
        mybox2.depth = 9;
        vol = mybox1.width * mybox1.height * mybox1.depth;
        System.out.println("Volume is " + vol);
        vol = mybox2.width * mybox2.height * mybox2.depth;
        System.out.println("Volume is " + vol); }
}
```

- مثال:

خروجی:

Volume is 3000
Volume is 162

شکل کلی تعریف یک متد در کلاس

- به شکل زیر است:

```
type_of_method   method_name (parameter_list){
//body of method
}
```

توصیف گرهای دسترسی در جاوا

- سه توصیف گر دسترسی وجود دارد: public، private، protected
- public: اگر یک عضو کلاسی داشته باشیم، آن عضو توسط هر کد دیگری در برنامه قابل دسترسی خواهد بود.
- private: اگر یک عضو کلاسی داشته باشیم، آن عضو فقط توسط سایر اعضای همان کلاس قابل دسترسی است.
- protected: زمانی استفاده می شود که وراثت وجود داشته باشد.

توصیف گرهای دسترسی در جاوا

- توصیف گر قبل از سایر مشخصات نوع عضو یک کلاس نیز قرار می گیرد.
- مثال:

```
public int i;
private double j;
private int myMethod(int a, char b ){
//...
```

مثالی از درک تفاوت توصیف گرها

```
public class Test {
        int a;
        public int b;
        private int c;
        void setc (int i){
                c = i; }
        int getc (){
                return c; }
}
public class AccessTest {
        public static void main(String args[] ){
        Test ob = new Test();
        ob.a = 10;
        ob.b = 20;
        ob.c = 100;  // Error!
        ob.setc(100); // OK
        System.out.println("a/b/and c:" + ob.a + " " + ob.b + " " + ob.getc());
        }
}
```

- مثال:

خروجی:

a/b/and c:10 20 100

وراثت

- امکان ایجاد طبقه بندی های سلسله مراتبی را به وجود می آورد.
- امکان ایجاد یک کلاس عمومی دارای ویژگی های مشترک یک مجموعه اقلام به هم مرتبط را دارد.
- این کلاس بعدا ممکن است توسط سایر کلاس ها به ارث برده شود.
- کلاسی که به ارث برده می شود را superclass می گوییم.
- کلاسی که عمل ارث بری انجام می دهد را subclass گوییم.

شکل کلی تعریف زیر کلاس که از کلاسی به ارث می برد

- به شکل زیر است:

```
class subclass_name extends superclass_name {

// body of class

}
```

مثالی از ارث بری

- برای ارث بری کافیست تعریف یک کلاس را با استفاده از واژه کلیدی extends در کلاس دیگری قرار دهیم.

```
public class A {
        int i, j;
        void showij (){
        System.out.println("i and j :" + i + " " + j); }
        }
public class B extends A {
        int k;
        void showk (){
        System.out.println("k: " + k); }
        void sum (){
        System.out.println("j+j+k: " +(i+j+k)); }
        }
public class SimpleInheritance {
        public static void main(String args[] ){
        A superOb = new A();
        B subOb = new B();
        superOb.i = 10;
        superOb.j = 20;
        System.out.println("Contents of superOb:");
        superOb.showij();
        subOb.i = 7;
        subOb.j = 8;
        subOb.k = 9;
        System.out.println("Contents of subOb :");
        subOb.showij();
        subOb.showk();
        System.out.println("Sum of i/j and k in subOb:");
        subOb.sum();
        }
}
```

خروجی:

```
Contents of superOb:
i and j :10 20
Contents of subOb:
i and j :7 8
k: 9
Sum of i/ j and k in subOb:
i+j+k: 24
```

فراخوانی سازندگان (constructors)

- مثال:

```
public class A {
        A (){
        System.out.println("Inside A's constructor."); }
    }
public class B extends A {
        B (){
        System.out.println("Inside B's constructor."); }
    }
public class C extends B {
        C (){
        System.out.println("Inside C's constructor."); }
    }
public class CallingCons {
    public static void main(String args[] ){
    C c = new C(); }
}
```

سازندگان به ترتیب مشتق شدنشان فراخوانی می شوند. چون یک سوپرکلاس نسبت به زیر کلاس های خود آگاهی ندارد.

خروجی:

```
Inside A's constructor
Inside B's constructor
Inside C's constructor
```

واژه کلیدی this

- گاهی لازم است به شیء که آن را فراخوانی کردیم ارجاع نماییم. برای این کار از واژه کلیدی this استفاده می شود.
- this همواره ارجاعی است به شیء که روی آن فراخوانی شده است.

```
Box (double w, double h, double d){
        this.width = w;
        this.height = h;
        this.depth = d;
}
```

استفاده از اشیاء به عنوان پارامتر

```
public class Test {
        int a, b;
        Test(int i, int j){
                a= i;
                b = j; }
        boolean equals(Test o ){
        if(o.a == a && o.b == b ) return true;
        else return false; }
}
public class PassOb {
        public static void main(String args[] ){
        Test ob1 = new Test(100, 22);
        Test ob2 = new Test(100, 22);
        Test ob3 = new Test(-1,- 1);
        System.out.println("ob1 == ob2: " + ob1.equals(ob2));
        System.out.println("ob1 == ob3: " + ob1.equals(ob3));
        }
}
```

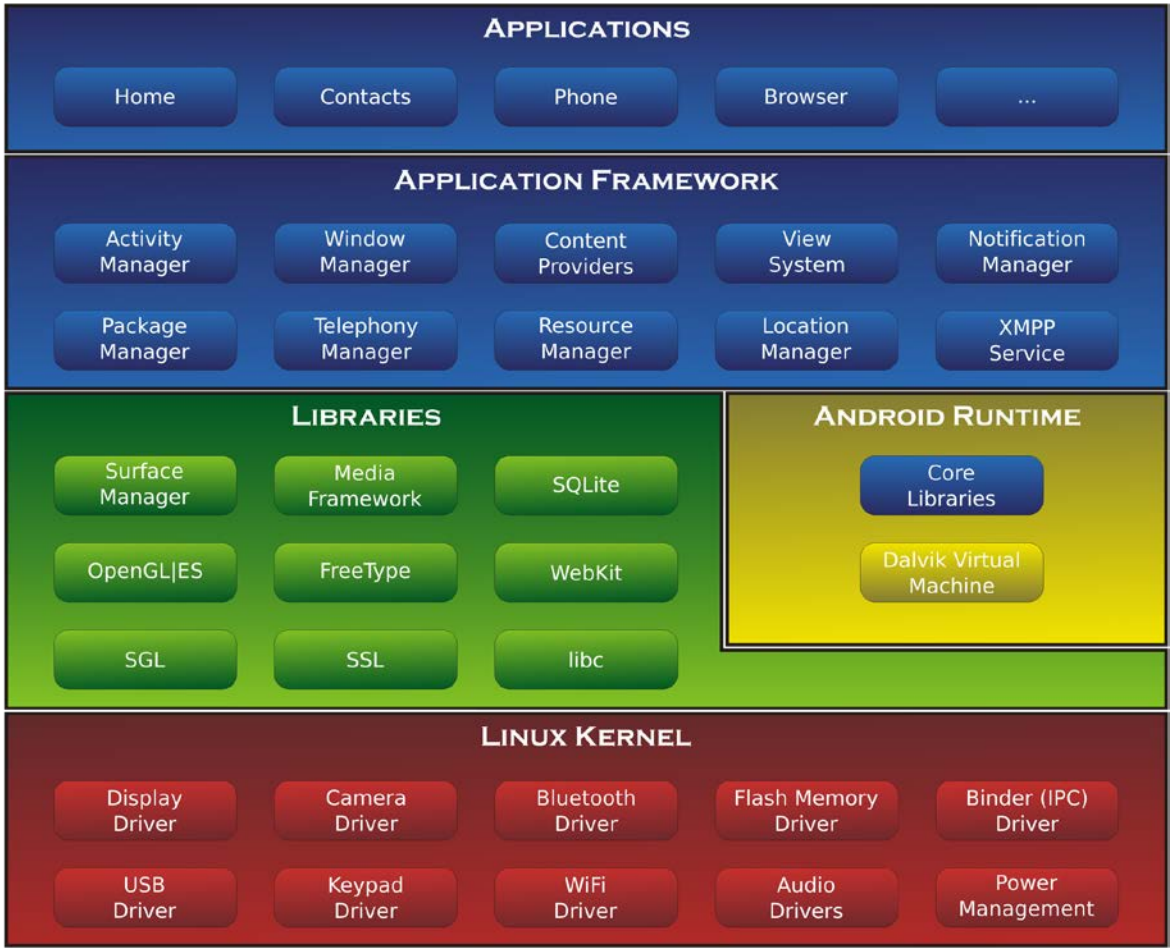
- مثال:

خروجی:

ob1 == ob2: true
ob1 == ob3: false

مفاهیم مقدماتی اندروید

معماری اندروید



معماری اندروید

- لایه هسته لینوکس (Linux Kernel)
  - جهت مدیریت حافظه، مدیریت پردازش ها، سرویس های شبکه و ... از لایه هسته لینوکس استفاده می شود.
  - نزدیکترین لایه به سخت افزار
  - قابل اطمینان بودن

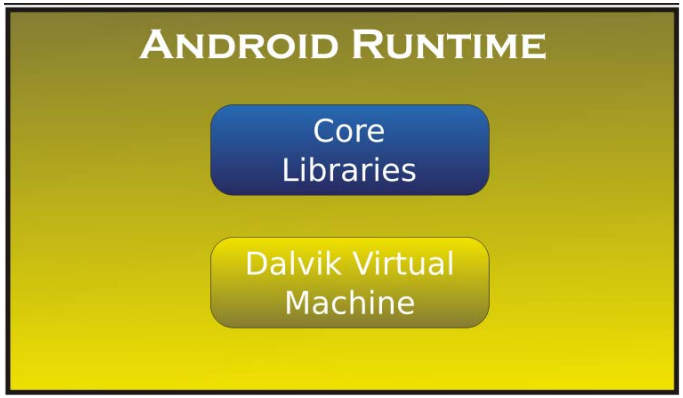
معماری اندروید

- لایه کتابخانه های اختصاصی
- با استفاده از کدهای c و c++ نوشته شده اند. مجموعه ای از کلاس های c و c++ که توسط کامپوننت های اندروید استفاده می شوند.
- برخی از این کتابخانه ها:

نام کتابخانه	کاربرد
Media Libraries	کتابخانه برای ضبط و پخش، تصویر و صدا. کار با فرمت های رایج مثل PNG و MP3 و ...
SGL	موتورهای گرافیکی دو بعدی
WebKit Library SSL	شامل یک موتور برای مرورگرهای پیشرفته که باعث قدرت و امنیت بیشتر مرورگر می شود.
OpenGL	برای کار با گرافیک سه بعدی است.
FreeType	برای ترجمه یا رندر فونت های بیت مپ و وکتوری طراحی شده است.
SQLite	برای کار با پایگاه داده طراحی شده است.

معماری اندروید

- لایه اندروید در زمان اجرا
- شامل ماشین مجازی دالویک و کتابخانه های هسته جاوا
- دالویک: طی فرایند ترجمه، کدها به دستورات مستقل از ماشین (بایت کد) تبدیل می شوند که بر روی ماشین دالویک اجرا می شوند. دالویک ماشین مجازی جاوایی است که برای استفاده در دستگاه های موبایل بهینه شده است. مصرف کم RAM و CPU.

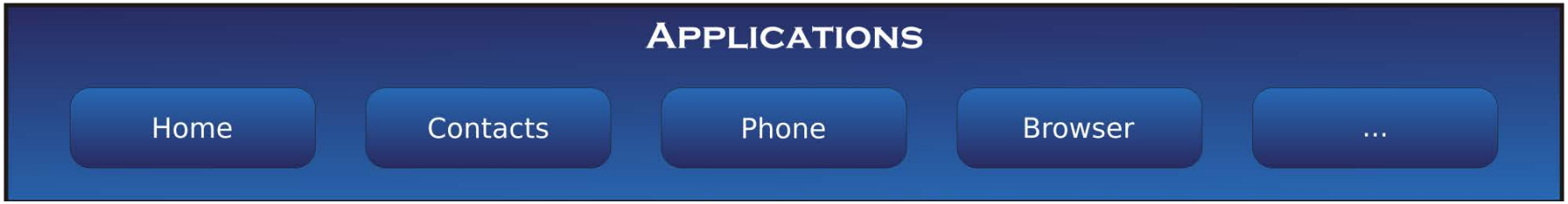


معماری اندروید

- لایه چارچوب برنامه
  - فراهم کردن ویژگی های مختلف برای برنامه نویسان و دسترسی به سخت افزارها.
  - دسترسی کامل به API های هسته اصلی. منظور توابعی که سازنده سیستم عامل منتشر کرده است.

معماری اندروید

- لایه برنامه های کاربردی
  - ارتباط کاربران نهایی فقط با این لایه است.
  - نرم افزارهای این لایه شامل، نرم افزارهای اختصاصی گوشی، بازی ها و نرم افزارهای ایجاد شده توسط شخص ثالث.



مدیریت برنامه ها در اندروید

- لینوکس چند کاربره است. در اندروید هر برنامه، کاربر در نظر گرفته می‌شود.
- سیستم به هر برنامه ای کد احراز هویت می دهد.
  - کد قابل شناسایی برای سیستم است و برای برنامه شناخته شده نیست.
- سیستم برای تمام فایل های برنامه مجوز صادر می کند.
- برنامه با کد هویتی خود به فایل ها دسترسی دارد. در نتیجه حفظ حریم خصوصی و عدم دسترسی به فایل های برنامه های دیگر.

مدیریت برنامه ها در اندروید (ادامه)

- در اندروید پردازش زمانی اجرا می شود که کامپوننت مورد نیاز برای اجرا را داشته باشد.
- پردازش زمانی متوقف می شود که دیگر نیازی به آن برنامه نباشد و یا سایر برنامه های در حال اجرا به حافظه بیشتری نیاز داشته باشد.
- اجرای اصل حداقل امتیاز: هر برنامه ای تنها به کامپوننت هایی دسترسی خواهد داشت که برای اجرا به آن ها نیاز داشته باشد و نه بیشتر. پس تا زمانیکه مجوز را نداشته باشد نمی تواند به منابع سیستم دسترس داشته باشد.

اشتراک داده ها در اندروید

- استفاده از یک کد هویتی مشترک.
- برنامه هایی با کد هویتی مشترک می توانند از یک ماشین مجازی استفاده کنند. در نتیجه حفظ منابع سیستم را خواهیم داشت.
- هر برنامه می تواند درخواست مجوز برای دسترسی به دیتای دستگاه را نیز داشته باشد. مانند مخاطبین، پیام ها، حافظه جانبی و ...
- تمامی مجوز ها در زمان نصب برنامه از کاربر سوال می شود.

کامپوننت ها در اندروید

- با استفاده از کامپوننت ها می توان رفتار کلی برنامه را مشخص کرد.
- چهار نوع کامپوننت وجود دارد که هرکدام هدف خاص خود را دارند. هر کامپوننت چرخه حیات (ایجاد شدن و از بین رفتن) مخصوص به خود را دارد.
- چهار نوع کامپوننت:
  - فعالیت ها (Activities)
  - سرویس ها (Services)
  - آماده سازهای محتوا (Content Providers)
  - دریافت کننده های اعلانات (Broadcast Receivers)

کامپوننت ها در اندروید – (Activities)

- بیانگر یک اسکرین کامل (صفحه نمایشگر) به همراه یک واسط کاربری است.
- تمامی فعالیت های مربوط به برنامه که از دید کاربر عبور می کند.
- مثال: برنامه چک کردن ایمیل: ممکن است حداقل دارای یک اکتیویتی برای چک کردن ایمیل، یک اکتیویتی برای نوشتن و ارسال، یک اکتیویتی برای خواندن داشته باشد که هرکدام مستقل از هم کار می کنند.

کامپوننت ها در اندروید – (Services)

- پردازشی است که در پشت صحنه اجرا می شود و از دید کاربر پنهان است.
- این پردازش دارای رابط کاربری نیست و واسط گرافیکی ندارد.
- در واقع کامپوننتی است که در پس زمینه اجرا می شود تا یک کار را برای مدت طولانی انجام دهد.
- یک اکتیویتی می تواند یک سرویس را اجرا کند.
- اجرا و توقف سرویس می تواند توسط کاربر نیز انجام شود.

کامپوننت ها در اندروید – (Content Providers)

- چارچوبی شامل کلاس هایی برای ذخیره داده ها است.
- از طریق آماده ساز محتوا می توان به اطلاعات ذخیره شده در گوشی دسترسی داشت.
- مثال: در اندروید آماده ساز محتوایی برای مدیریت تماس کاربر وجود دارد. بنابراین هر برنامه با داشتن مجوز لازم می تواند به اطلاعات تماس دسترسی پیدا کند.
- مناسب جهت خواندن و نوشتن فایل های خصوصی برنامه که نیازی به اشتراک گذاری ندارند.

کامپونت ها در اندروید – (Broadcast Receivers)

- پخش اعلانات همگانی سیستم را بر عهده دارد.
- مثال: خاموش یا روشن بودن صفحه نمایش، خالی بودن باتری، عکس گرفته شده یا خیر و ...
- واسط گرافیکی ندارد.

اجرای کامپونت ها

- اجرای اکتیویتی، سرویس و Broadcast Receivers توسط Intent انجام می‌شوند.
- Intent، کامپونت ها را در زمان اجرا به هم وصل می کند.
- در اکتیویتی و سرویس، Intent معرف عملی برای اجرا است.
  - مثال: اینتنت ممکن است یک درخواست به اکتیویتی ارسال کند و آن اکتیویتی یک تصویر نمایش دهد، یا صفحه وب باز کند و ...
  - گاهی برگشت نتیجه ممکن است به صورت یک اینتنت برگشت داده شود. مثلا اینتنتی ارسال می شود تا کاربر شماره تماس مورد نظرش را انتخاب کند و این شماره در قالب یک اینتنت برگشت داده شود.

کامپوننت Content Providers توسط اینتنت اجرا نمی شود. زمانی فعال می شود که یک Content Resolver برسد.

فایل مانیفست (Manifest)

- قبل از اینکه کامپوننتی اجرا شود، سیستم باید بداند که کامپوننت وجود دارد یا خیر؟ این کار با خواندن فایل مانیفست انجام می شود.
- تمامی کامپوننت های موجود در برنامه باید در مانیفست معرفی شوند.
- دیگر محتویات فایل مانیفست:
  - تعریف مجوزهای دسترسی
  - تعریف حداقل سطح دسترسی API
  - تعریف version code و version name برنامه
  - تعریف خصوصیات سخت افزاری و نرم افزاری که توسط برنامه مورد استفاده قرار می گیرد. مثل دوربین، بلوتوث، صفحه تاچ و ...
  - تعریف کتابخانه های API استفاده شده در برنامه. مانند کتابخانه گوگل مپ و ...

مثالی از یک فایل مانیفست

- مثال:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <uses-sdk android:minSdkVersion=”10" android:targetSdkVersion=”23" />
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="net.harifi.ept.MainActivity"
            android:label="@string/example_label" ... >
        </activity>
...
</application>
</manifest>
```

نحوه تعریف مجوز دسترسی برای برنامه در فایل مانیفست

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="net.harifi.ept" >
<uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

- در صورتی که چند مجوز نیاز باشد، هر مجوز به صورت جداگانه تعریف می شود:

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.INTERNET" />
```

مجوز	کاربرد	مجوز	کاربرد
INTERNET	دسترسی به اینترنت	RECEIVE_SMS	نظارت بر پیام های ورودی
READ_CONTACTS	فقط خواندن اطلاعات مخاطبان	ACCESS_COARSE_LOCATION	دسترسی به وسایلی مثل wifi
WRITE_CONTACTS	نوشتن اطلاعات در مخاطبان	ACCESS_FINE_LOCATION	دسترس به وسایلی مثل GPS

نحوه مدیریت حافظه اکتیویتی ها

- هر برنامه ممکن است چندین اکتیویتی به هم مرتبط داشته باشد.
- به طور معمول یک اکتیویتی اصلی در برنامه وجود دارد (main_activity)
- با شروع هر اکتیویتی جدید، اکتیویتی قبلی متوقف می شود و در استک نگه داشته می شود (همچنان حافظه به آن تخصیص داده شده است).
- با اجرای چند اکتیویتی، اکتیویتی ها به ترتیب اجرا در داخل استک قرار می‌گیرند.
- مکانیزم خاتمه اکتیویتی، LIFO است.
  - خاتمه توسط کلید back، خاتمه توسط کدهای برنامه
  - خاتمه به دلیل عدم وجود فضای کافی (مکانیزم خاتمه: اکتیویتی که اخیراً کمترین دسترسی به آن شده)

وضعیت های مختلف اکتیویتی ها

- در حال ساخت (onCreate): وقتی که اکتیویتی برای اولین بار اجرا می‌شود. در این حالت می تواند مقادیر اولیه تعریف کرد.
- در حال شروع (onStart): اکتیویتی ظاهر می شود و کاربر آن را مشاهده می کند.
- در حال از سر گرفتن (Resumed): اکتیویتی در صفحه نمایش ظاهر شده است و مشغول کار (در حال اجرا است) است.
- در حال توقف (Paused): اکتیویتی به پس زمینه می رود یا در پشت برنامه دیگری که کل یا مقداری از صفحه را در اختیار دارد، در حال اجرا است.

وضعیت های مختلف اکتیویتی ها (ادامه ۱)

- در حالت قطع (Stopped): اکتیویتی به صورت کامل توسط اکتیویتی دیگری پنهان می شود. این اکتیویتی در دسترس نیست و هر زمان که کمبود حافظه پیش آید می تواند توسط سیستم حذف شود.
- در حالت شروع مجدد (onRestart): حالتی که اکتیویتی از توقف به شروع مجدد می رود، یعنی اکتیویتی دوباره ظاهر می شود.
- در حال نابودی (onDestroy): برای از بین بردن اکتیویتی و آزاد کردن حافظه، اکتیویتی به این حالت می رود.

وضعیت های مختلف اکتیویتی ها (ادامه ۲)

- در حال ذخیره وضعیت (onSaveInstanceState): برای ذخیره وضعیت اکتیویتی استفاده می شود. مثلا موقعیت مکان نما. این حالت به صورت پیش فرض پیاده سازی شده است.

- حالت بازگردانی وضعیت قبلی (onRestoreInstanceState): وضعیت حالت ذخیره وضعیت را بر می گرداند. این حالت نیز به صورت پیش فرض پیاده سازی شده است.

در هر حالتی از اکتیویتی می توانیم با اجرای متد finish()
اکتیویتی را به پایان رسانده و حافظه را پاک کنیم.

انواع API ها در اندروید

نام	نسخه	API
No codename	1.0	1
Petit Four	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Eclair	2.0 - 2.1	5-7
Froyo	2.2 - 2.2.3	8
Gingerbread	2.3 - 2.3.7	9-10
Honeycomb	3.0 - 3.2.6	11-13
Ice Cream Sandwich	4.0 - 4.0.4	14-15
Jelly Bean	4.1 - 4.3.1	16-18

نام	نسخه	API
KitKat	4.4 - 4.4.4	19-20
Lollipop	5.0 - 5.1.1	21-22
Marshmallow	6.0 - 6.0.1	23
Nougat	7.0 - 7.1.2	24-25
اعلام نشده	8.0	??

- شروع کار با اندروید استودیو
  - مفاهیم مقدماتی اندروید استودیو

مفهوم گریدل (Gradle)

- یک بیلد سیستم (Bulid System) اپن سورس بر اساس زبان Groovy است. بیلد سیستم یک ابزار نرم افزاری کامپایل کردن کد به صورت اتوماتیک است. اندروید استودیو از Gradle استفاده می کند.
- هدف: کامپایل و اجرای کد.

Groovy، زبان برنامه‌نویسی پویایی است که برای ماشین مجازی جاوا ایجاد شده است که هم می‌تواند مفسر اجرا شود و هم کامپایل شود.
همه چیز در گرووی یک شی است. گرووی تمام عملگرها را به صورت فراخوانی تابع پیاده‌سازی می‌کند. به عنوان مثال ۱ + ۱ به صورت زیر است:

```
1.plus(1)
```

دیگر بیلد سیستم ها:

Ant بر اساس زبان xml

و

Maven بر اساس زبان xml

مزایای گریدل (Gradle)

- گریدل می تواند وابستگی ها را مدیریت کند یا به پروژه اضافه کند.
- گریدل می تواند عملیات تست را روی برنامه انجام دهد.
- گریدل می تواند چند خروجی از برنامه بگیرید.
- در زمان ایجاد پروژه، گریدل فایل های مورد نیاز برای پروژه را می سازد.
  - فایل bulid.gradle (تنظیمات اصلی کل پروژه)(تنظیمات هر ماژول)

مفهوم پروگارد (ProGuard)

- پروگارد ابزار رایگان و اپن سورس است که بر روی کلاس های جاوا موارد زیر را انجام می دهد:
  - فشرده سازی یا کوچک کردن برنامه (minification)،
  - بهینه سازی (optimization)،
  - ناخوانا کردن (obfuscation)،
  - بسته بندی مجدد (repackaging)

برخی دستورات مهم در پروگارد

- keep- : کلاس ها نگه داشته شود و نام و بسته را تغییر ندهد.
- dontwarn- : اگر ارجاعاتی به/در کلاس یا کلاس‌های مشخص شده وجود دارد که پروگارد آن‌ها را به هر دلیل پیدا نمی‌کند، هشدار ندهد و آن‌ها را نادیده بگیرد.
- ignorewarnings- : عملکرد آن شبیه dontwarn است با یک تفاوت، در لاگ خروجی این ارجاع نامشخص را می‌نویسد اما به کار خود ادامه می‌دهد.
- keepattributes- : ویژگی هایی را که اعلام می کنیم با این دستور نگه می‌دارد و آن ها را حذف نمی کند.

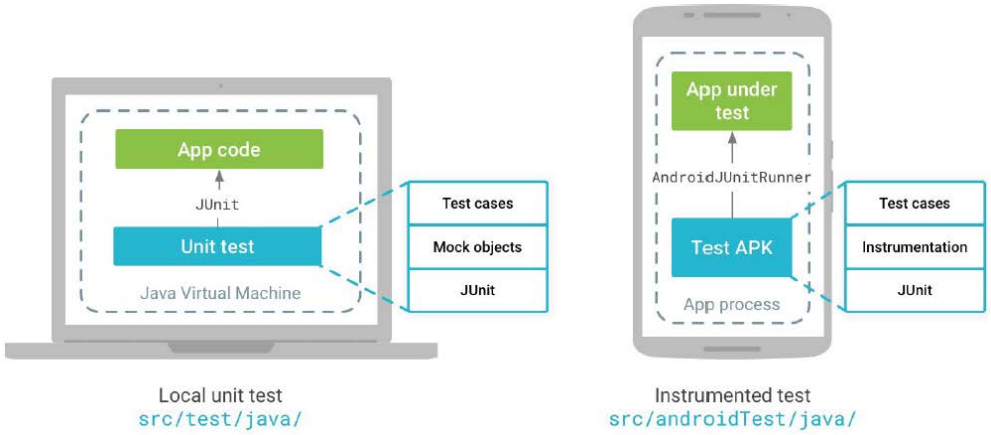
لاگ

- ثبت واقع و گزارشات برنامه در حین اجرا توسط لاگ انجام می شود.
- مشخص شدن مسیر خطا در یک اکتیویتی خاص.
- کمک به توسعه دهنده برای ثبت لاگ های شخصی.
- موارد موجود در لاگ فایل:
  - اطلاعات عمومی از اجرای برنامه
  - هشدارهای برنامه
  - خطاهای برنامه
  - موارد مربوط به دیباگ

تست جی یونیت (Junit)

- یکی از فریمورک های جاوا برای انجام تست واحد، بر روی پروژه انجام می‌شود.
- دو نوع است: لوکال (Local)، ابزاری (Instrumented)

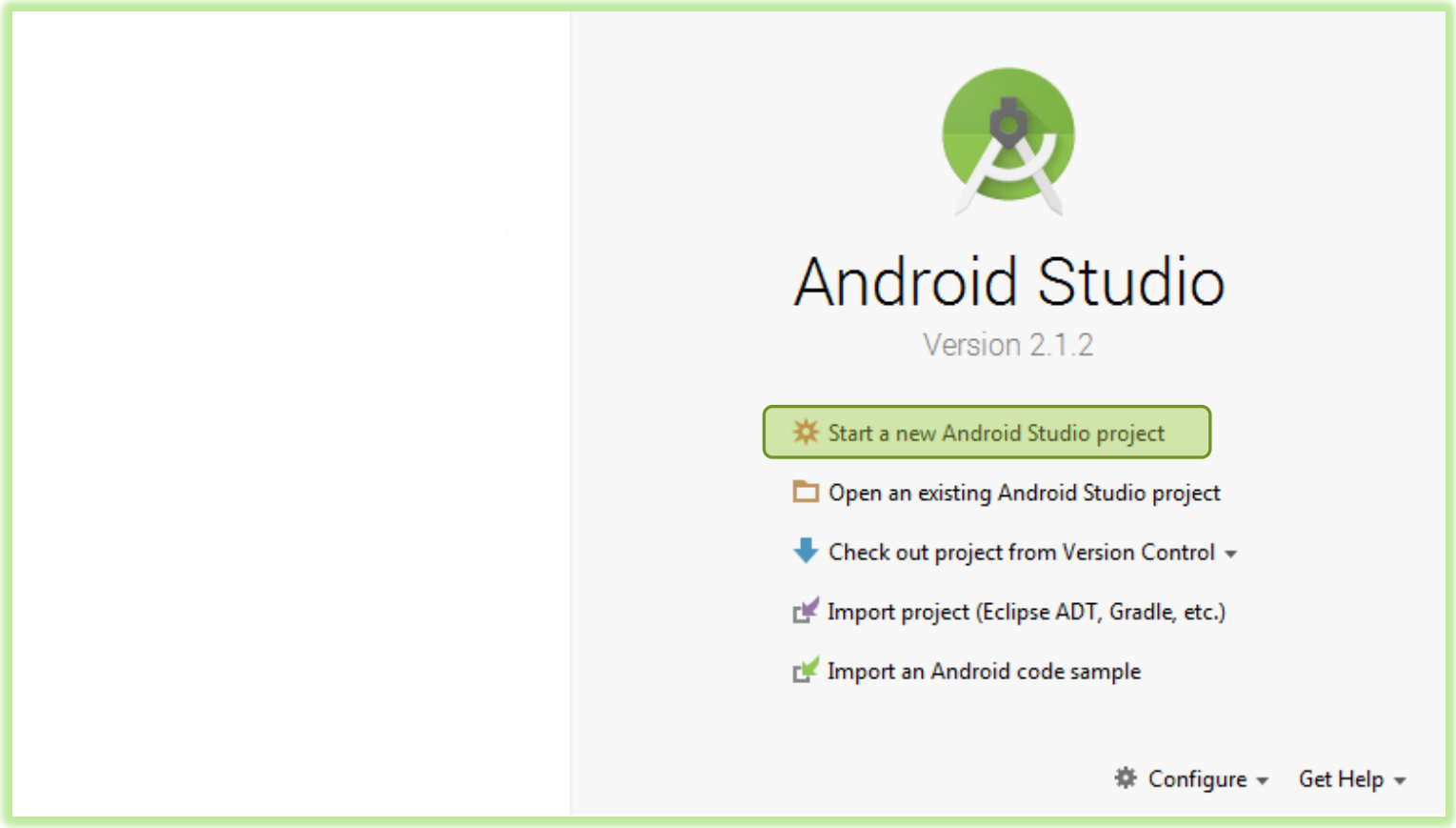
تست واحد بر روی دستگاه شبیه سازی انجام می شود. زمانی استفاده می شود که برنامه دارای وابستگی باشد.

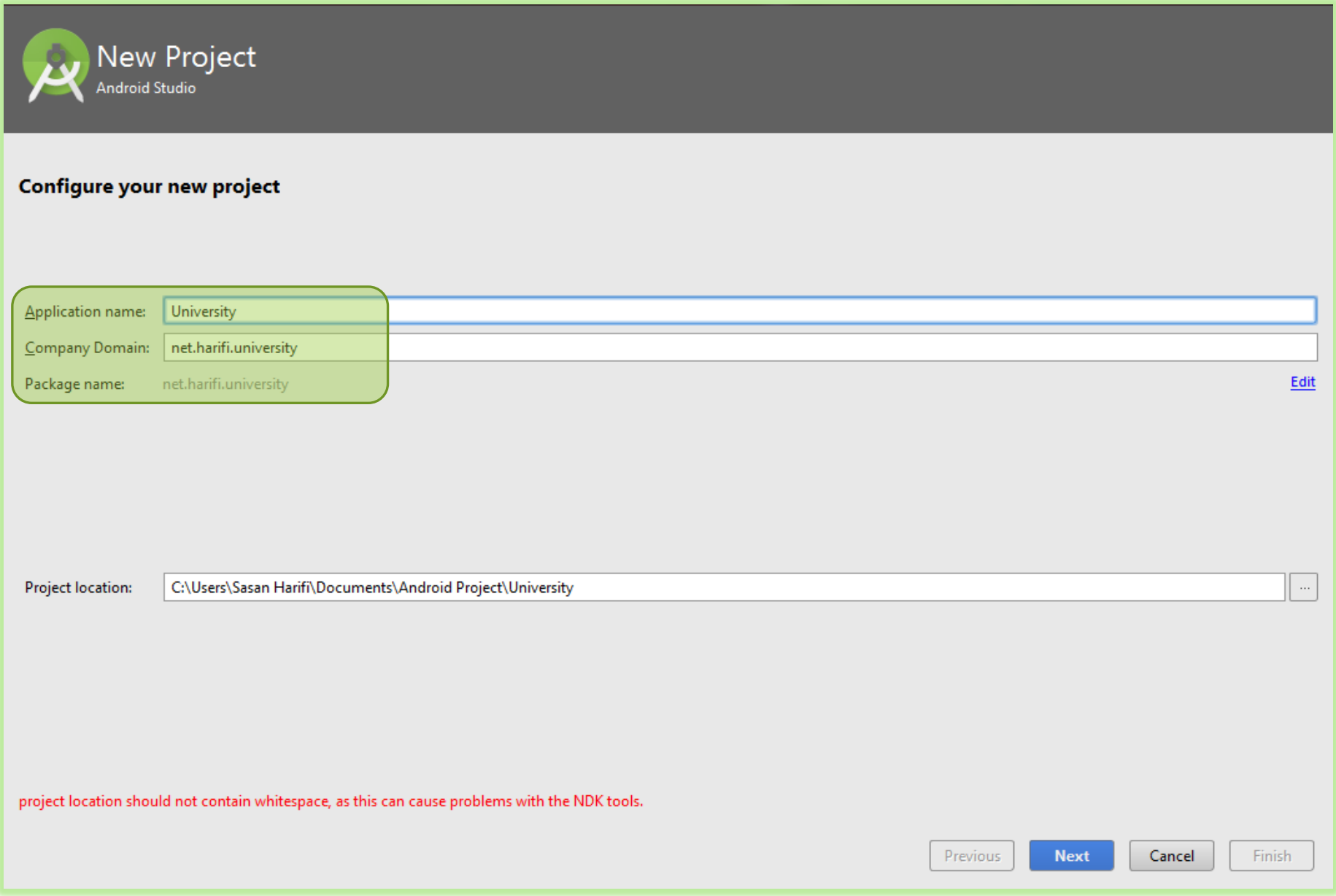


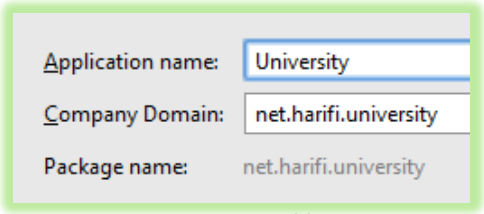
تست واحد به صورت محلی در ماشین مجازی جاوا انجام می شود. زمانی استفاده می شود که برنامه وابستگی نداشته باشد. در نتیجه کاهش زمان اجرای تست را خواهیم داشت.

- شروع کار با اندروید استودیو
  - مفاهیم مقدماتی اندروید استودیو
  - ایجاد پروژه در اندروید استودیو و مفاهیم آن

ایجاد پروژه جدید

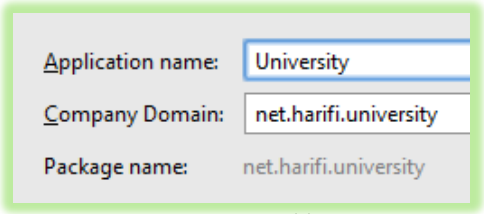



New Project
Android Studio
Configure your new project
Application name:
University
Company Domain:
net.harifi.university
Package name:
net.harifi.university
Edit
Project location:
C:\Users\Sasan Harifi\Documents\Android Project\University
project location should not contain whitespace, as this can cause problems with the NDK tools.
Previous
Next
Cancel
Finish



قوانین نام گذاری نام برنامه و بسته

- Application name نام اصلی برنامه است.
- بعد از نصب برنامه بر روی دستگاه، این نام در لیست برنامه های دستگاه نمایش داده می شود.
- حرف اول نام اپلیکیشن باید با حروف بزرگ نوشته شود.
- Company Domain نام تجاری یا شخصی سازنده برنامه است.



قوانین نام گذاری نام برنامه و بسته

- Company Domain نام تجاری به صورت استاندارد از سه بخش که با نقطه از هم جدا می شوند، تشکیل می شود.
- بهتر است برعکس آدرس وبسایت باشد.
- بخش اول دامنه وبسایت. بخش دوم نام وبسایت. بخش سوم نام برنامه.

آدرس وبسایت:

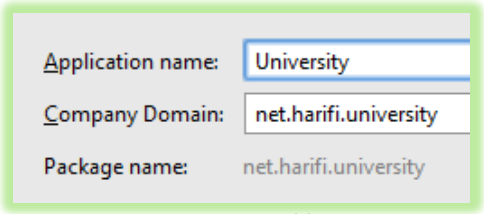
harifi.net

نام برنامه:

University

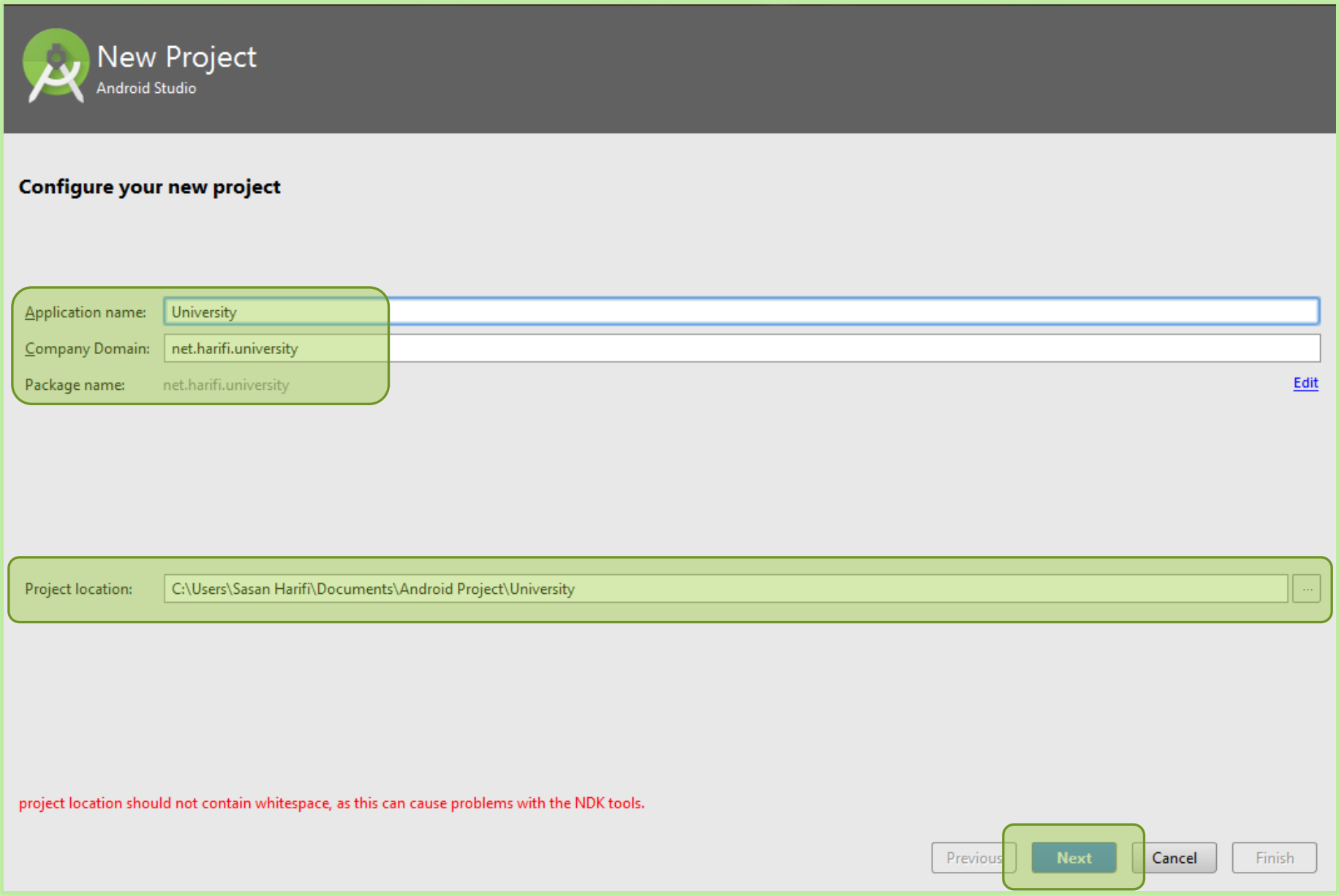
:Company name

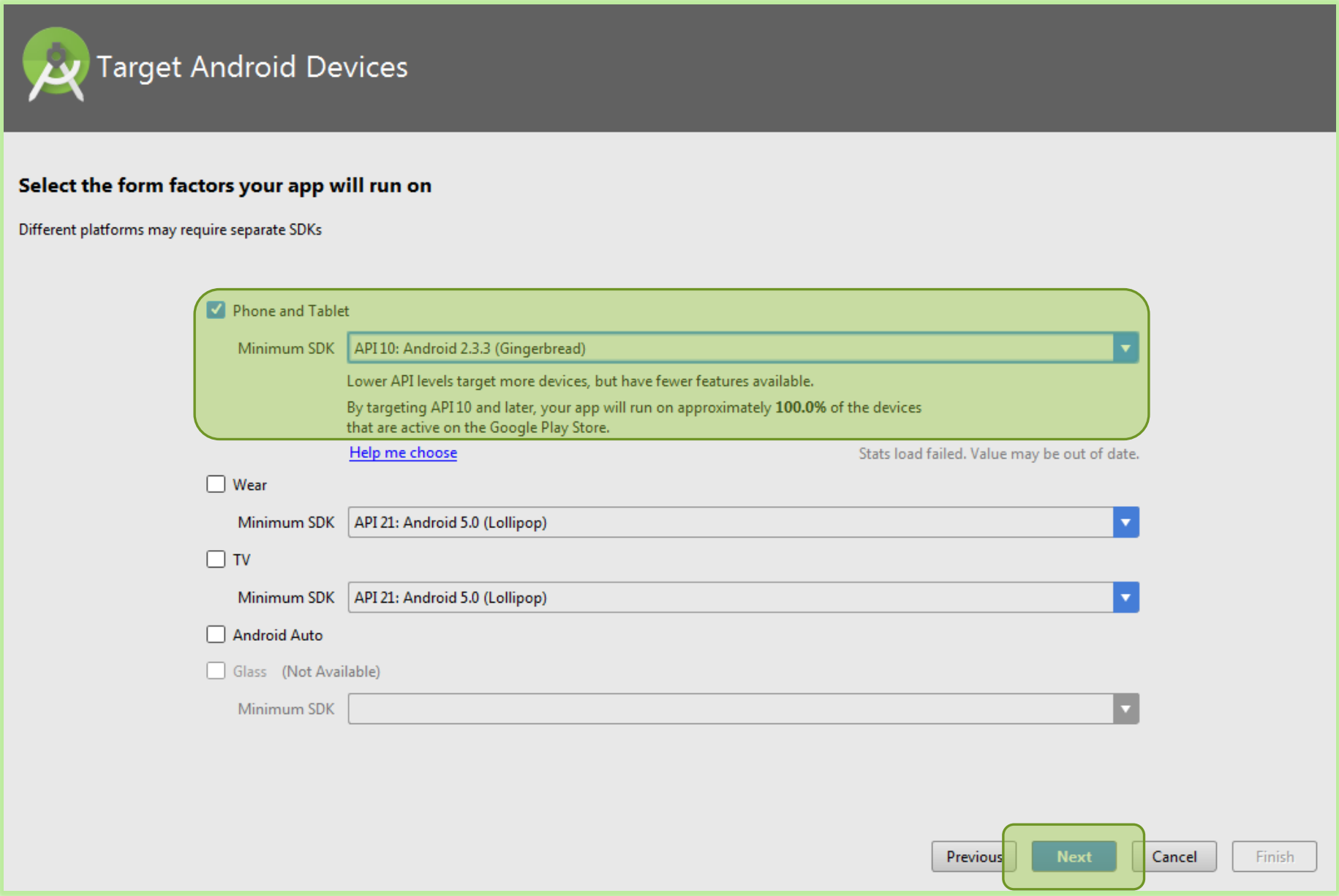
net.harifi.university

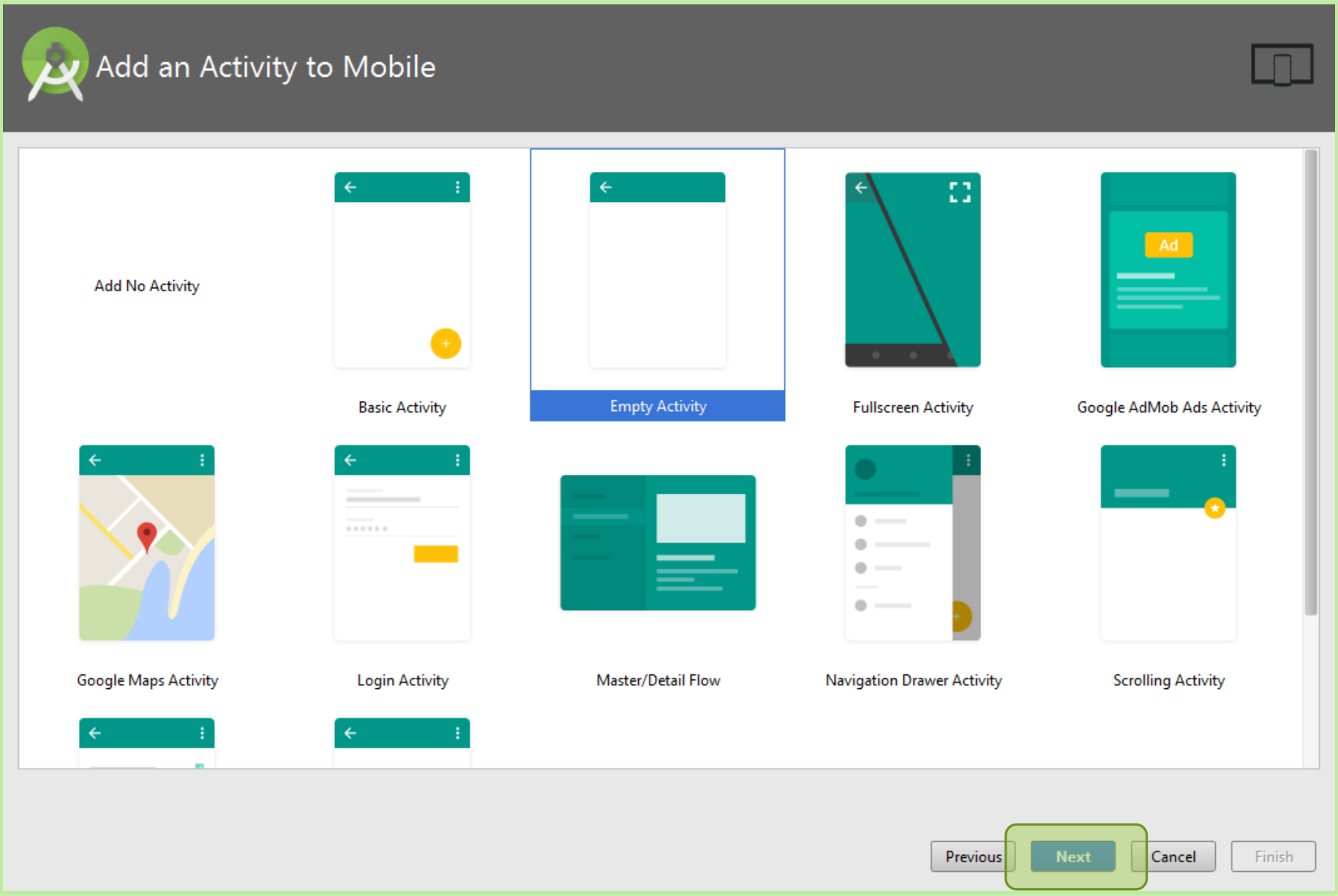


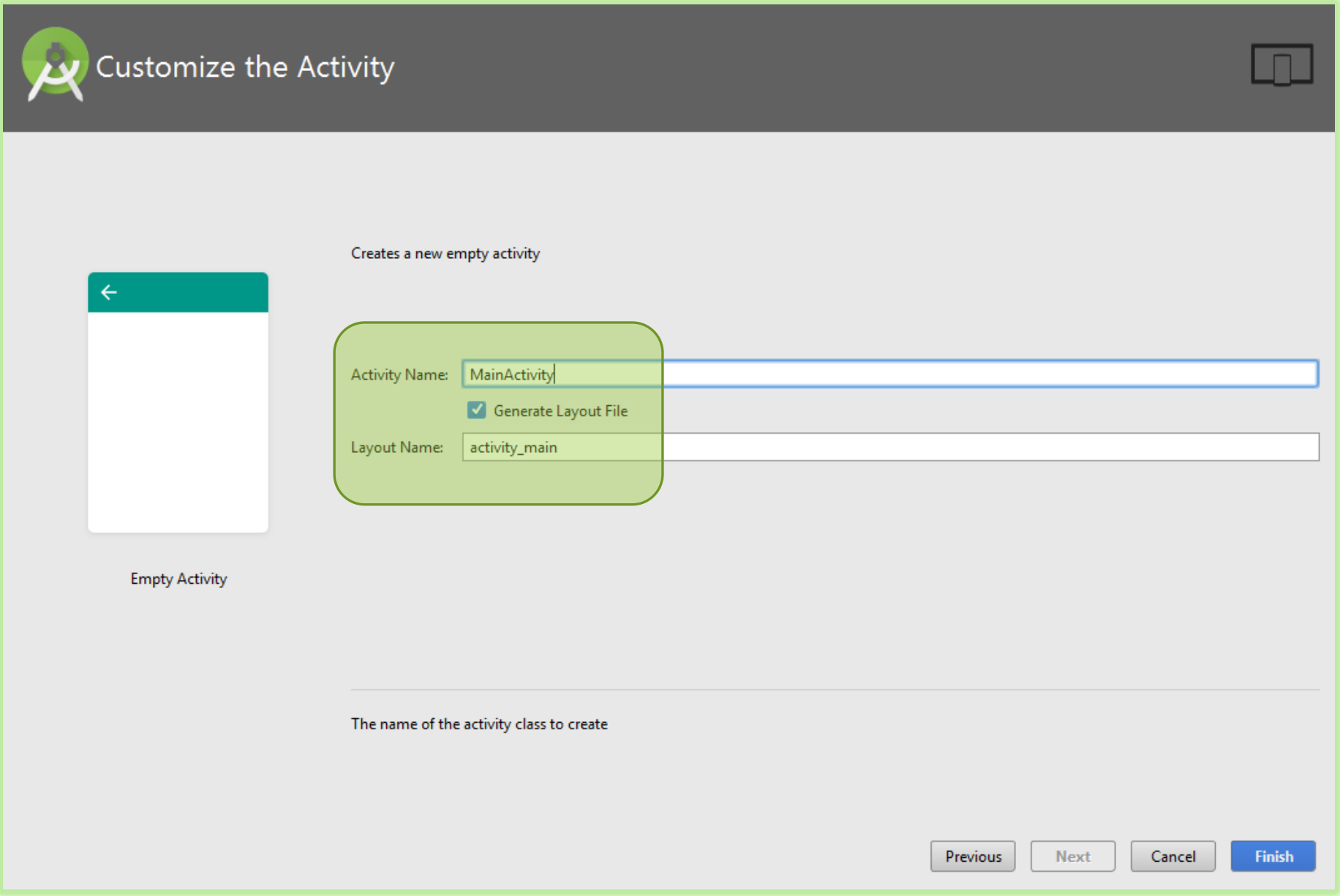
قوانین نام گذاری نام برنامه و بسته

- Package name نام بسته برنامه است که در مارکت ها قرار می گیرد.
- هر برنامه نام بسته منحصر به فرد دارد.
- هر برنامه در طول توسعه نباید نام بسته اش را تغییر دهد.
- در صورت تغییر نام بسته، برنامه در مارکت به صورت جدا در نظر گرفته می شود.
- نام بسته به صورت خودکار در فایل مانیفست برنامه قرار می گیرد.


New Project
Android Studio
Configure your new project
Application name:
University
Company Domain:
net.harifi.university
Package name:
net.harifi.university
Edit
Project location:
C:\Users\Sasan Harifi\Documents\Android Project\University
project location should not contain whitespace, as this can cause problems with the NDK tools.
Previous
Next
Cancel
Finish


Target Android Devices
Select the form factors your app will run on
Different platforms may require separate SDKs
Phone and Tablet
Minimum SDK
API 10: Android 2.3.3 (Gingerbread)
Lower API levels target more devices, but have fewer features available.
By targeting API 10 and later, your app will run on approximately 100.0% of the devices that are active on the Google Play Store.
Help me choose
Stats load failed. Value may be out of date.
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Android Auto
Glass (Not Available)
Minimum SDK
Previous
Next
Cancel
Finish


Add an Activity to Mobile
Add No Activity
Basic Activity
Empty Activity
Fullscreen Activity
Ad
Google AdMob Ads Activity
Google Maps Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Previous
Next
Cancel
Finish

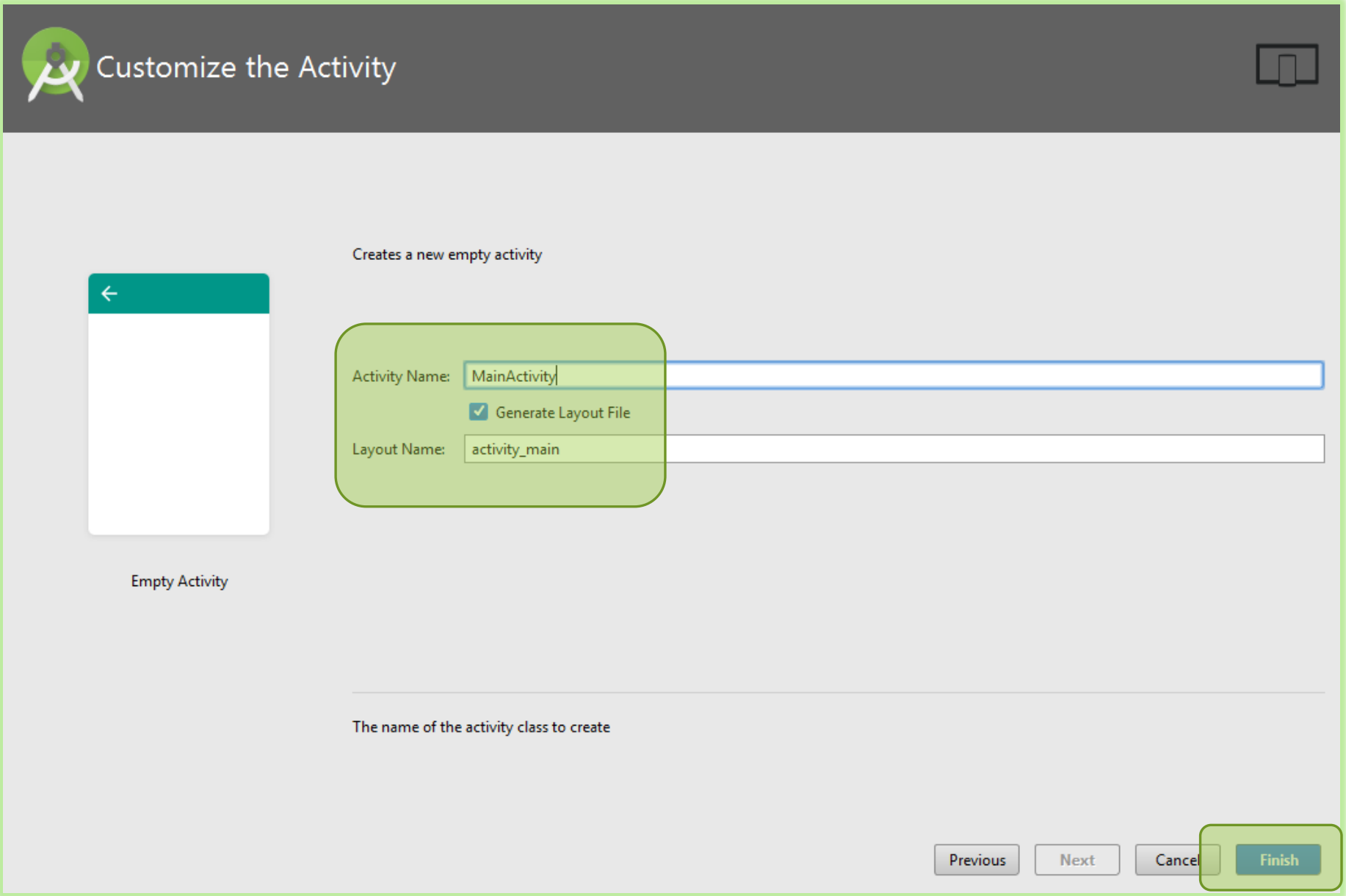

Customize the Activity
Creates a new empty activity
Activity Name:
MainActivity
Generate Layout File
Layout Name:
activity_main
Empty Activity
The name of the activity class to create
Previous
Next
Cancel
Finish

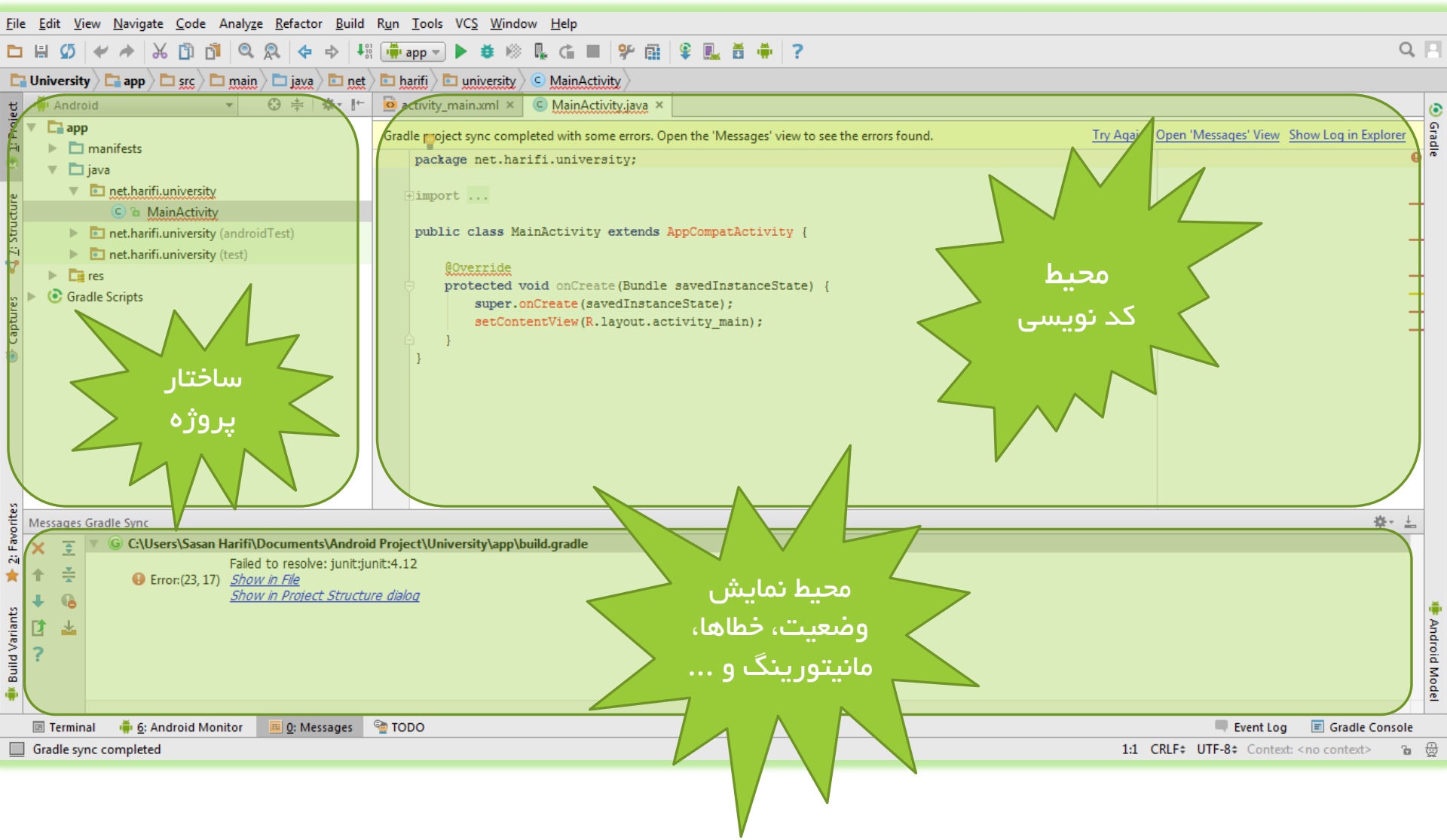
قوانین نام گذاری نام اکتیویتی

- اکتیویتی ایجاد شده در این مرحله اکتیویتی اصلی برنامه است.
- نام اکتیویتی باید با حروف بزرگ شروع شود.
- اگر نام بیشتر از یک کلمه باشد، حرف اول کلمات با حروف بزرگ نوشته می شود.
- می توان برای نام گذاری از ساختار camelCase نیز استفاده کرد.
- برای نام اکتیویتی "فاصله" مجاز نیست.

قوانین نام گذاری نام لایه

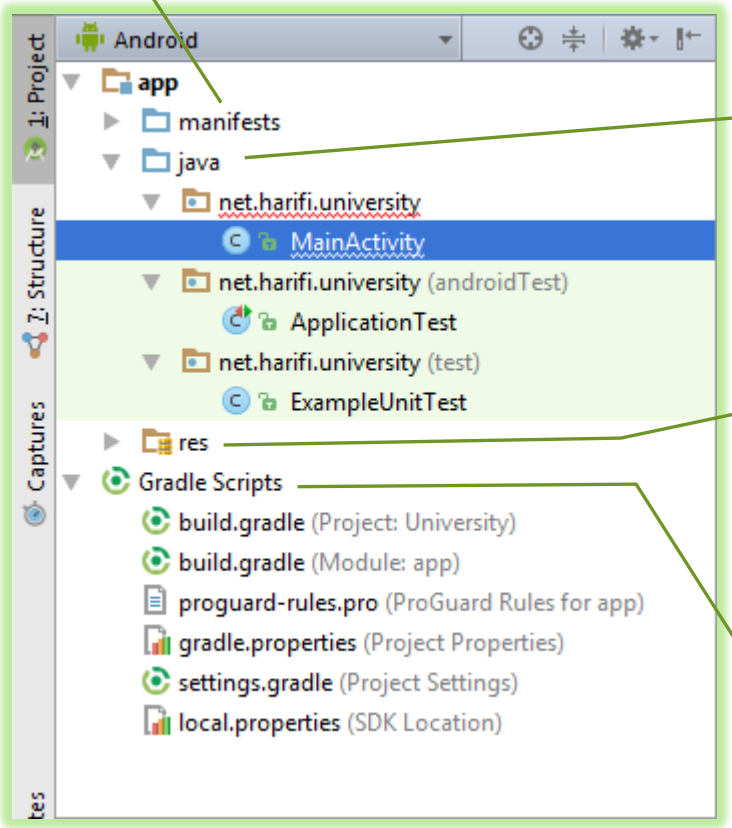
- لایه (Layout) بخش UI برنامه را تشکیل می دهد.
- برای نام گذاری حتما باید از حروف کوچک استفاده شود.
- برای نام لایه کاراکتر "فاصله" مجاز نیست.
- اگر نام از چند کلمه تشکیل شده است می توان از کاراکتر "آندرلاین" برای جدا سازی کلمات استفاده کرد.


Customize the Activity
Creates a new empty activity
Activity Name:
MainActivity
Generate Layout File
Layout Name:
activity_main
Empty Activity
The name of the activity class to create
Previous
Next
Cancel
Finish


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
University app src main java net harifi university MainActivity
Android
app
manifests
java
net.harifi.university
MainActivity
net.harifi.university (androidTest)
net.harifi.university (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
Gradle project sync completed with some errors. Open the 'Messages' view to see the errors found.
Try Agai
Open 'Messages' View
Show Log in Explorer
package net.harifi.university;
import ...
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
}
ساختار
پروژه
محیط
کد نویسی
Messages Gradle Sync
C:\Users\Sasan Harifi\Documents\Android Project\University\app\build.gradle
Failed to resolve: junit:junit:4.12
Error:(23, 17)
Show in File
Show in Project Structure dialog
محیط نمایش
وضعیت، خطاها،
مانیتورینگ و ...
Terminal
6: Android Monitor
0: Messages
TODO
Event Log
Gradle Console
Gradle sync completed
1:1
CRLF
UTF-8
Context: <no context>
Gradle
Android Model
Build Variants
2: Favorites
Captures
Structure
Project

بخش ساختار پروژه

اطلاعات مربوط به فایل مانیفست برنامه



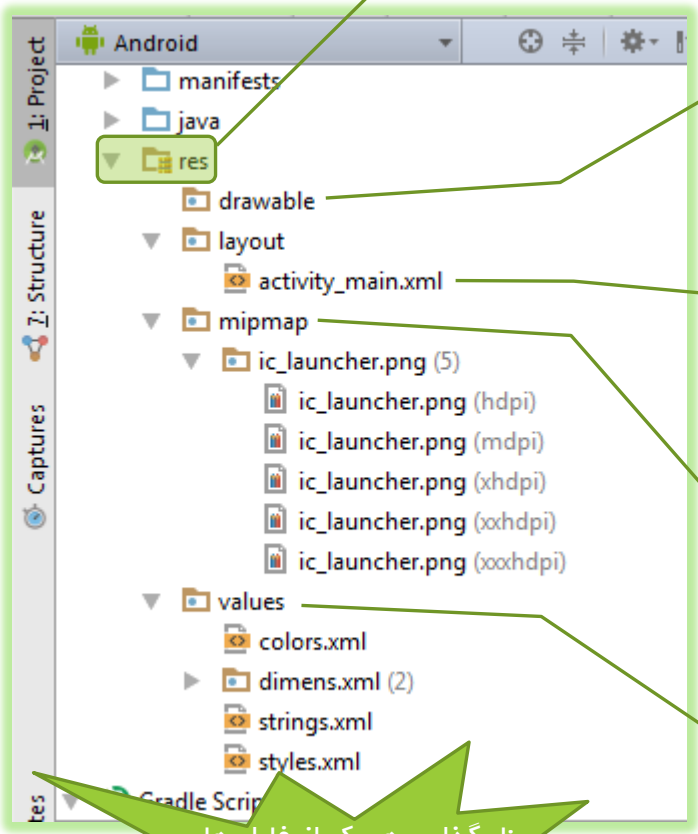
کدهای جاوا برنامه در این قسمت نوشته می شود. فایل های جاوای اکتیویتی ها در این قسمت قرار می گیرد. همچنین فولدر های مربوط به تست واحد نیز در این قسمت قرار دارد.

این پوشه شامل همه منابعی است که می توانیم در برنامه استفاده کنیم. کلیه عناصر موجود در یک برنامه مثل فایل های تصویری، صوتی، انیمیشن، آیکن و ... در این پوشه قرار می گیرد. این پوشه خود دارای زیر شاخه هایی می باشد که هریک از موارد فوق الذکر باید در زیر شاخه مخصوص به خود قرار گیرند.

فایل های مربوط به گریدل در این قسمت قرار می گیرند.

این پوشه شامل همه منابعی است که می توانیم در برنامه استفاده کنیم. کلیه عناصر موجود در یک برنامه مثل فایل های تصویری، صوتی، انیمیشن، آیکن و ... در این پوشه قرار می گیرد. این پوشه خود دارای زیر شاخه هایی می باشد که هریک از موارد فوق الذکر باید در زیر شاخه مخصوص به خود قرار گیرند.

بخش ساختار پروژه – زیر شاخه res



تمامی عکس ها باید در این پوشه قرار گیرد. همچنین تصاویر وکتوری SVG تبدیل شده به فرمت xml نیز در این پوشه قرار می گیرد.

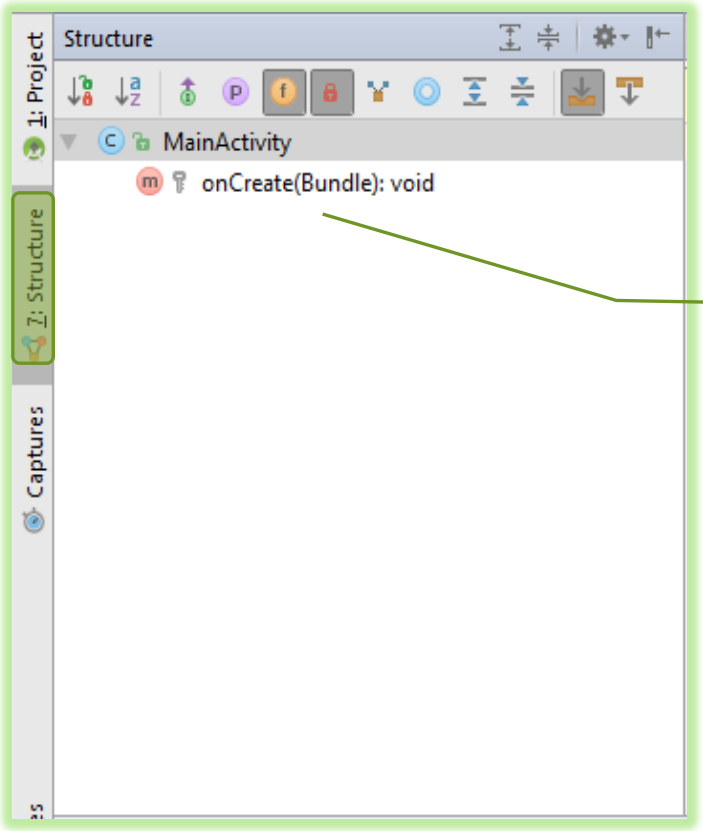
فایل لایه UI مربوط به هر اکتیویتی در این قسمت قرار می گیرد. بعد از ایجاد اکتیویتی، فایل مربوط به لایه آن به صورت خودکار در این بخش قرار می گیرد.

آیکن‌های برنامه با نام ic_launcher در این قسمت و در زیر شاخه های مربوط به هر رزولوشن صفحه نمایش قرار می‌گیرند. هر برنامه حداقل شامل ۵ تصویر آیکن با ابعاد ۱۹۲، ۱۴۴، ۹۶، ۷۲ و ۴۸ و با فرمت PNG است.

شامل فایل های xml برای تعریف رشته ها، رنگ ها و ... در صورت نیاز.

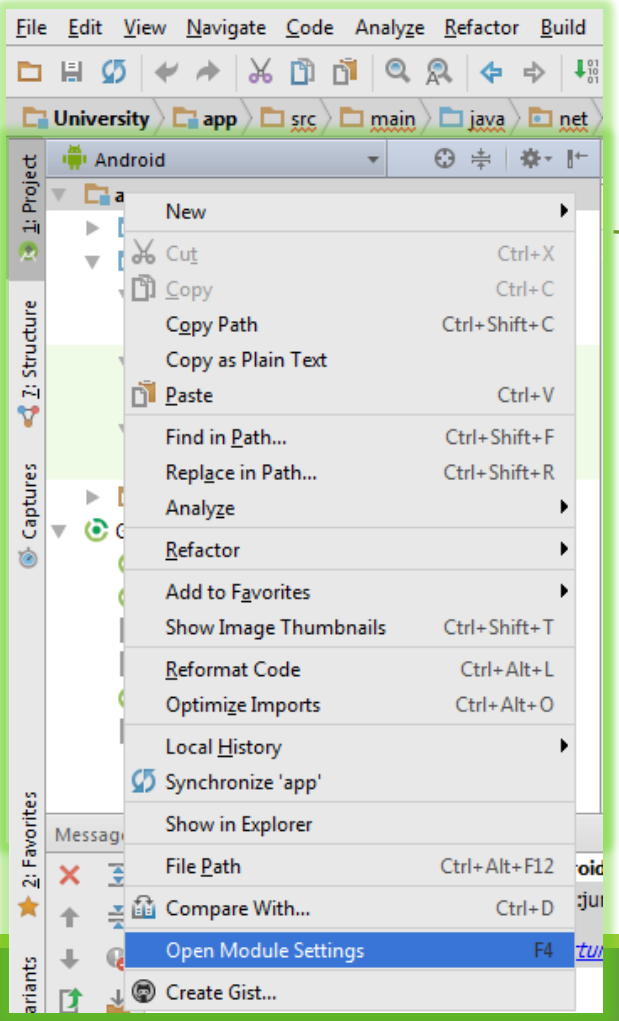
نام‌گذاری هریک از فایل های پوشه res تابع قوانین نام‌گذاری لایه است.

بخش ساختار پروژه – قسمت Structure



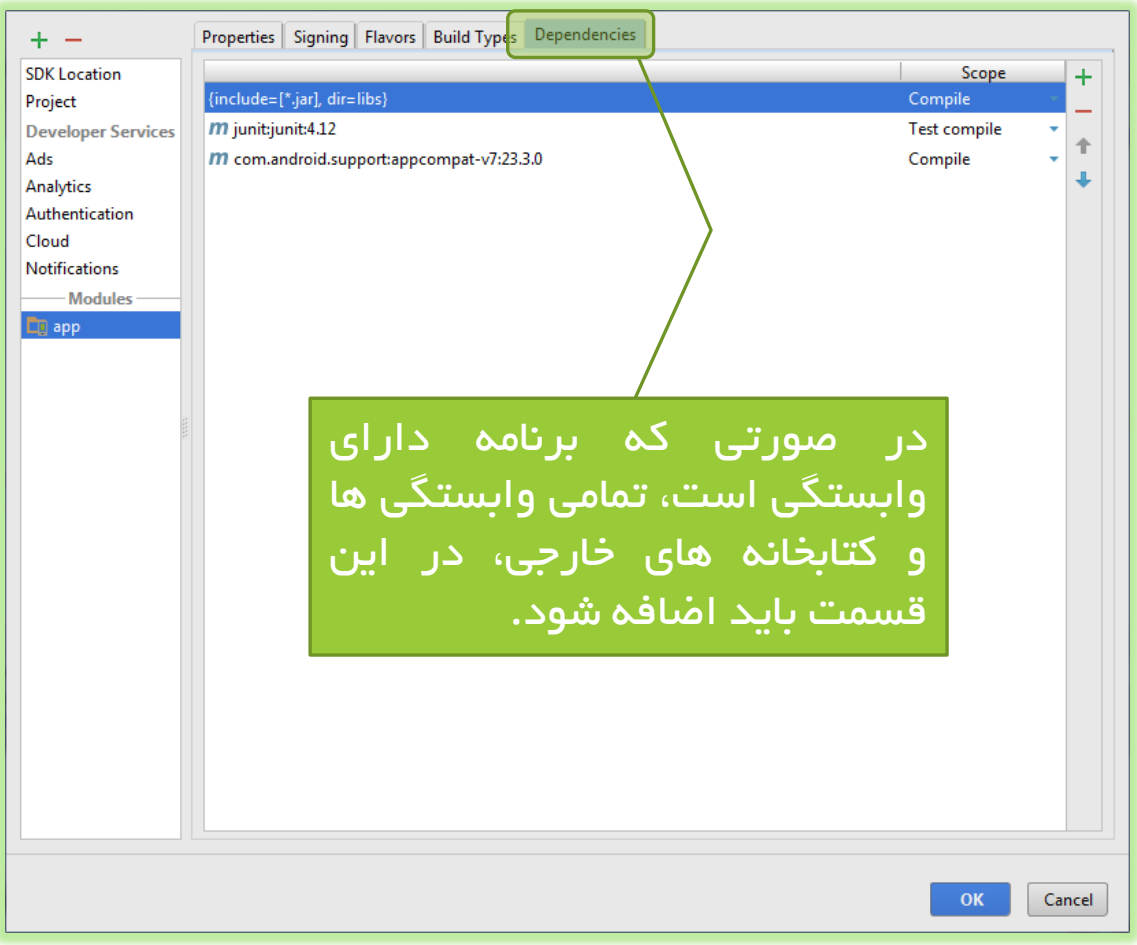
مشاهده تمام کلاس ها و زیر کلاس ها در این بخش امکان پذیر است.

بخش تنظیمات مربوط به پروژه
module settings

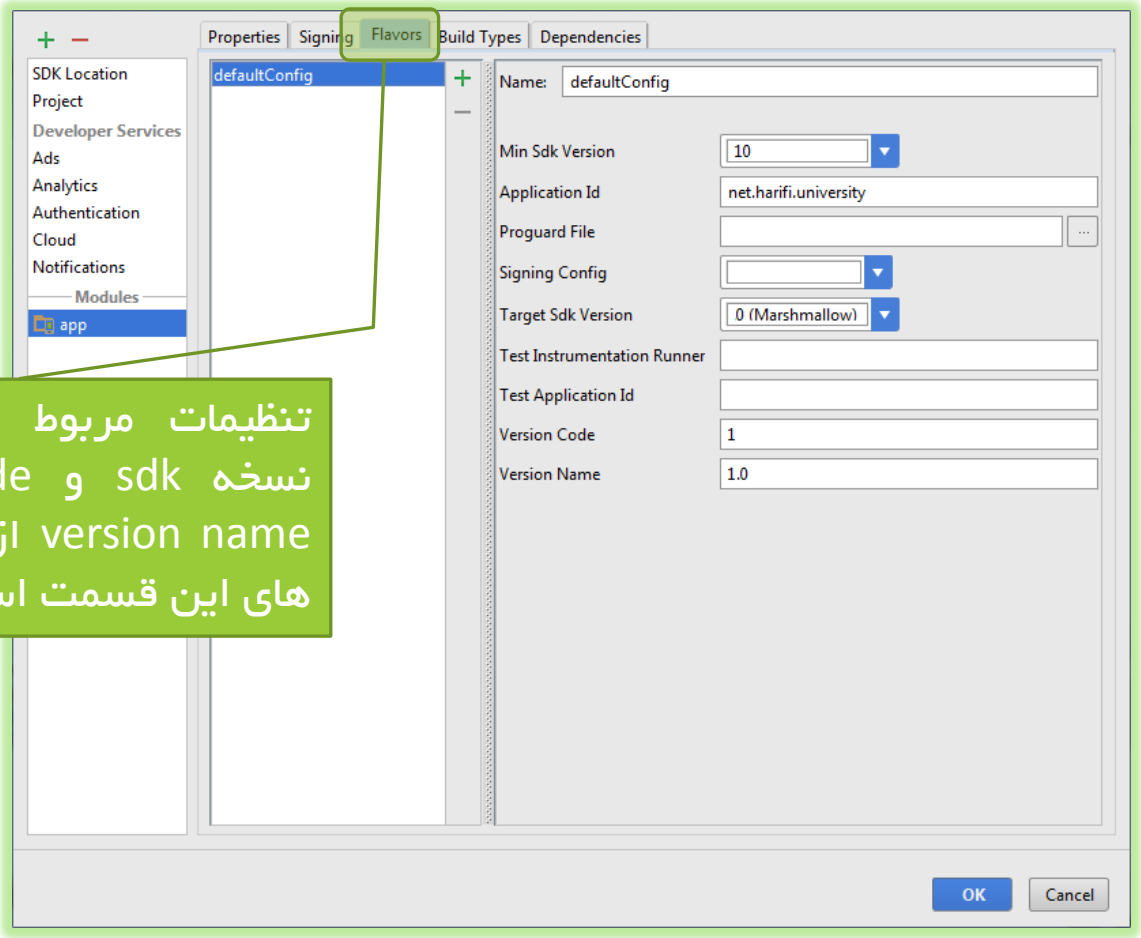


راست کلیک روی پروژه و
انتخاب گزینه
Open module settings

بخش تنظیمات مربوط به پروژه
module settings



بخش تنظیمات مربوط به پروژه
module settings

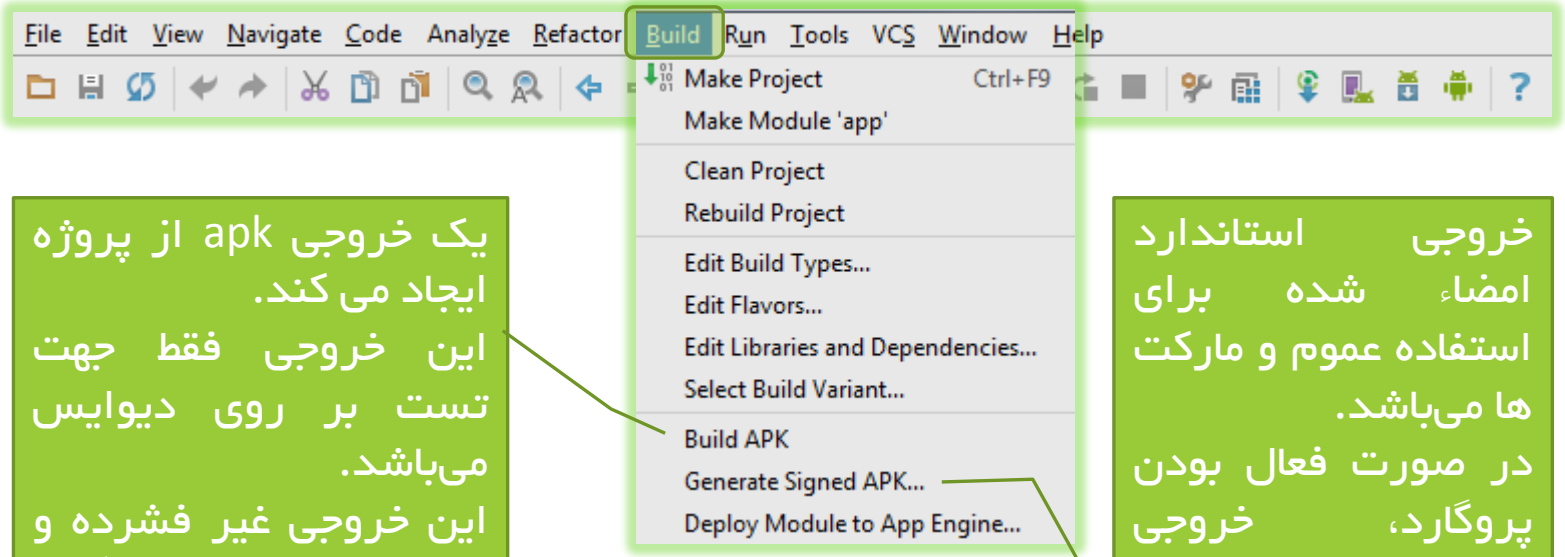


تنظیمات مربوط به پایین ترین نسخه sdk و version code و version name از مهمترین بخش های این قسمت است.

نکاتی در مورد
version code و version name

- version code
  - فقط یک عدد صحیح می تواند باشد.
  - برای تشخیص چندمین آپدیت برنامه در مارکت ها از آن استفاده می شود.
  - برای هربار به روز رسانی، عدد نسخه جدید باید بیشتر از عدد نسخه قبل باشد.
- version name
  - می تواند عدد یا حروف یا کلمه و غیره باشد.
  - جهت اطلاع کاربر نهایی برای تشخیص نسخه فعلی از نسخه قبلی استفاده می‌شود.

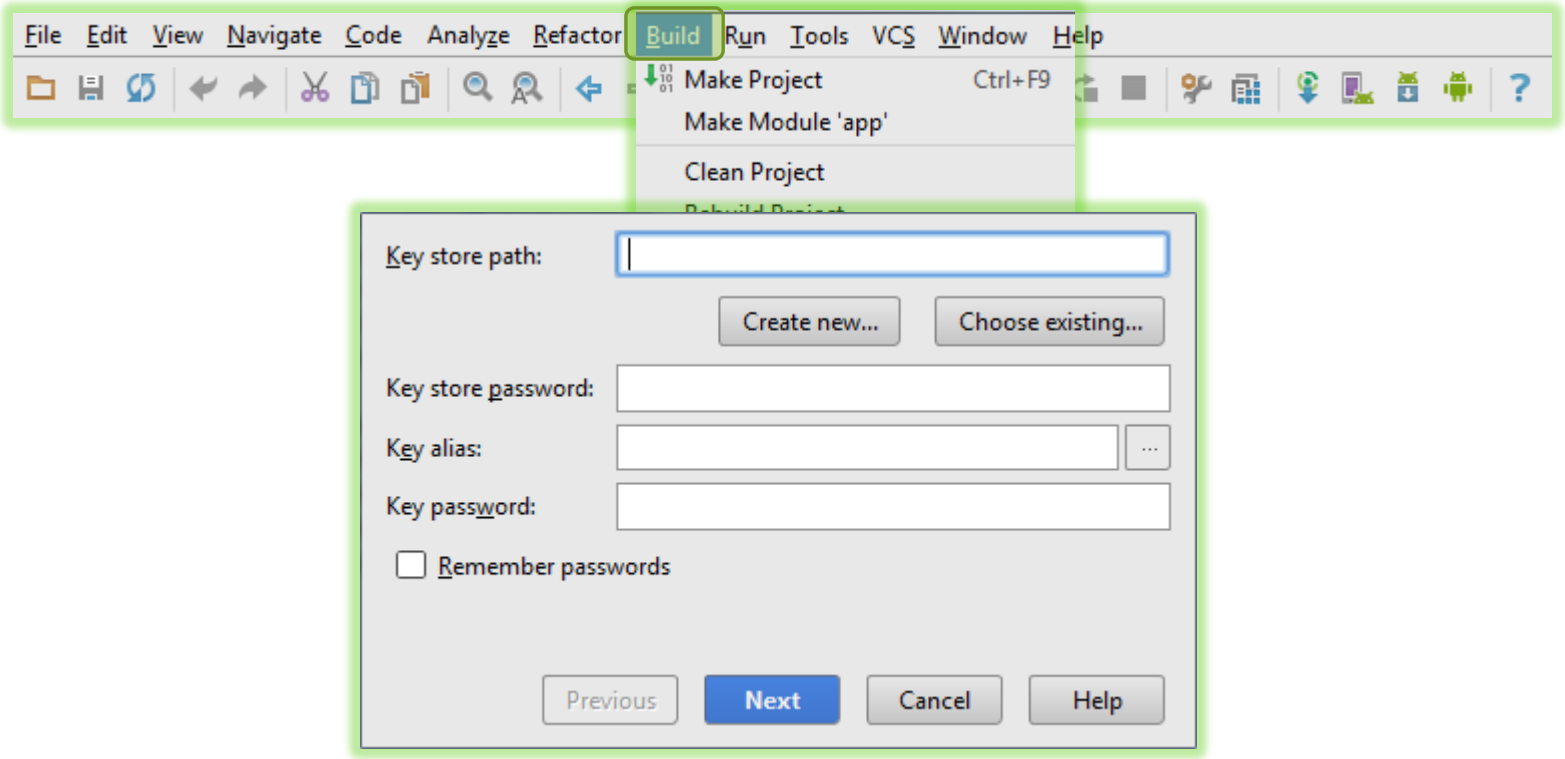
منوی build در اندروید استودیو



یک خروجی apk از پروژه ایجاد می کند.
این خروجی فقط جهت تست بر روی دیوایس می‌باشد.
این خروجی غیر فشرده و غیر بهینه بوده و هیچگونه محافظ کدی ندارد.
قابلیت دیکامپایل کلاس ها در این نوع خروجی وجود دارد (عدم امنیت کد).

خروجی استاندارد امضاء شده برای استفاده عموم و مارکت ها می‌باشد.
در صورت فعال بودن پروگارد، خروجی محافظت شده است.
کلاس های غیر ضروری در خروجی حذف شده و کد فشرده می شود.
نیاز به ایجاد کلید رمز دارد.

Generate Signed APK...



نکته: در صورتی که کلید ایجاد شود، تا زمانی که برنامه به روز رسانی می شود باید از همان کلید استفاده شود. در صورتی که کلید تغییر کند، برنامه در مارکت و دستگاه کاربر به عنوان برنامه جدید شناخته می شود در نتیجه به روز رسانی اعمال نخواهد شد.

محتوای فایل پروگارد

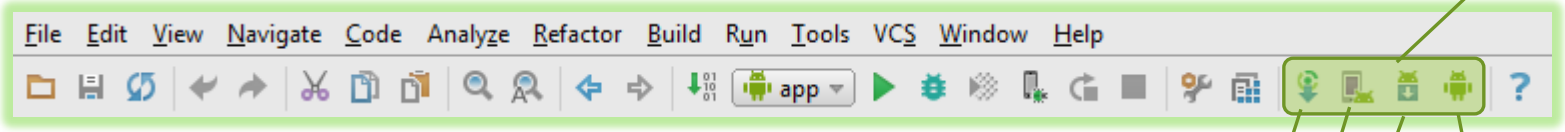
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.3"

    defaultConfig {
        applicationId "net.harifi.university"
        minSdkVersion 10
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:appcompat-v7:23.3.0'
}
```

محتوای فایل مانیفست

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.harifi.university">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="University"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

بخش های مهم در نوار ابزار

قسمت های پرکاربرد



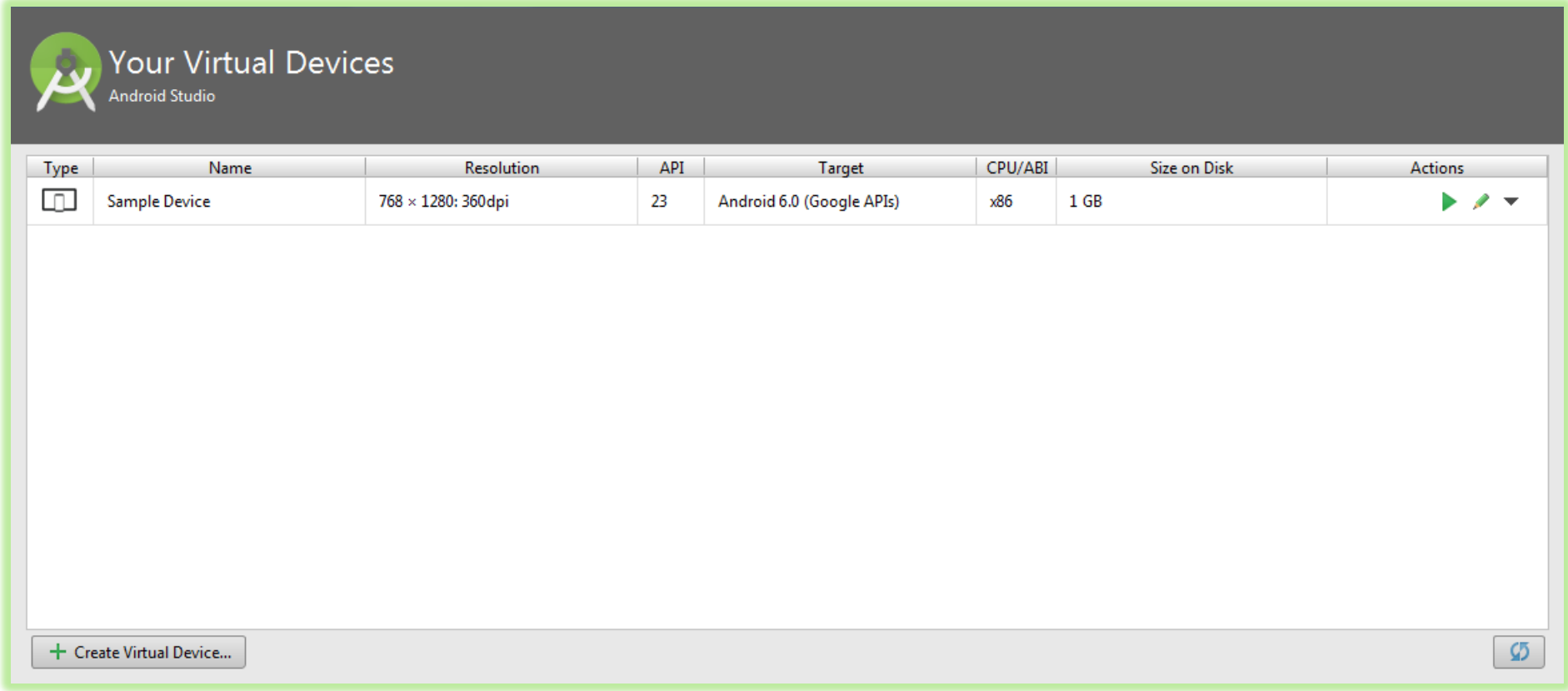
Sync Project with Gradle Files
به هنگام سازی فایل های پروژه با گریدل

AVD Manager
مدیریت دیوایس شبیه ساز

SDK Manager
مدیریت مربوط به sdk را انجام می دهد.
قابلیت آپدیت فایل های sdk در این قسمت قرار دارد.

Android Device Monitor
دسترسی به بخش عملکرد و کارایی برنامه از نظر حافظه، پردازنده، پردازنده گرافیک و شبکه را فراهم می کند.

AVD Manager

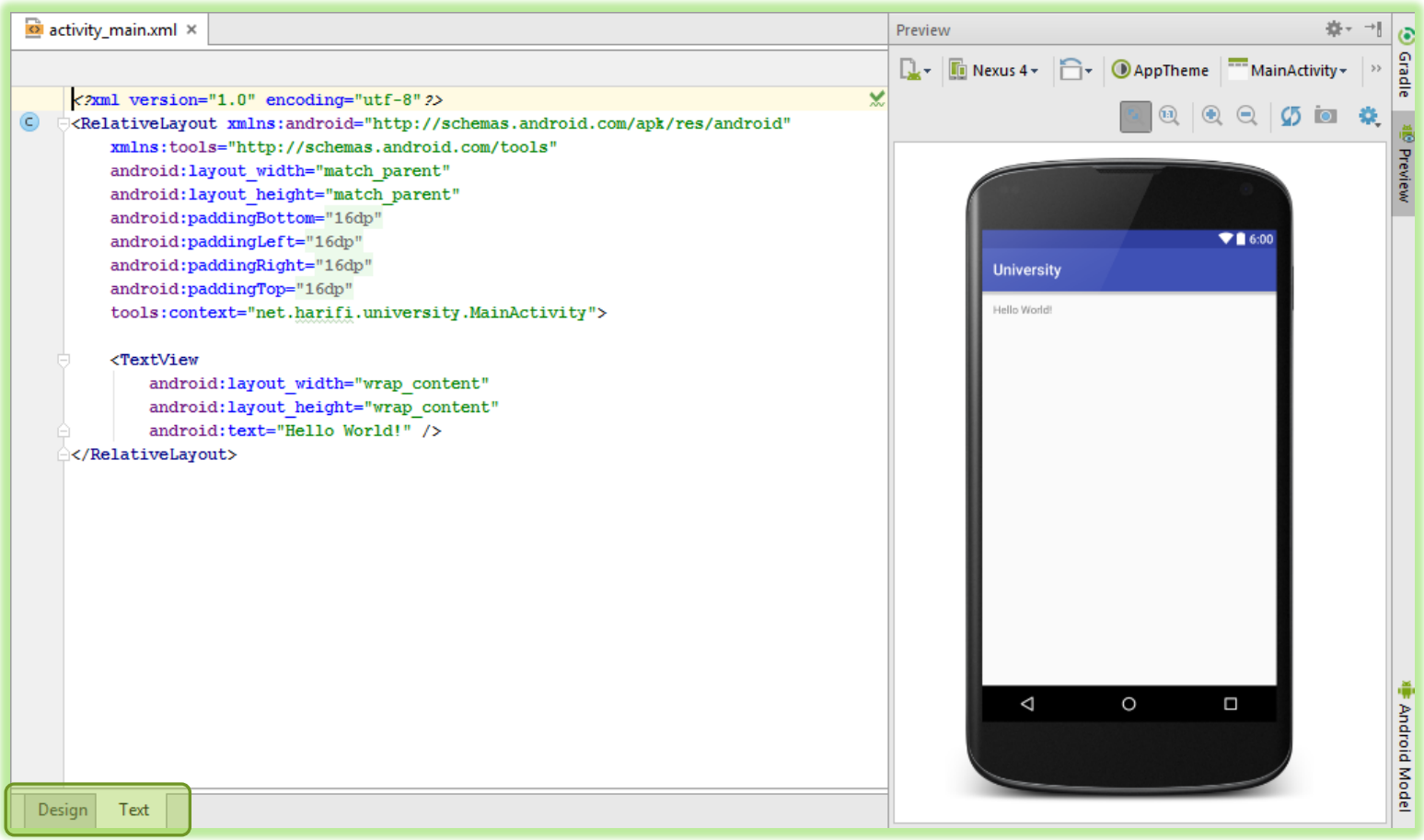

Your Virtual Devices
Android Studio
Type
Name
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Sample Device
768 × 1280: 360dpi
23
Android 6.0 (Google APIs)
x86
1 GB
Create Virtual Device...

بررسی فایل جاوای برنامه

MainActivity.java

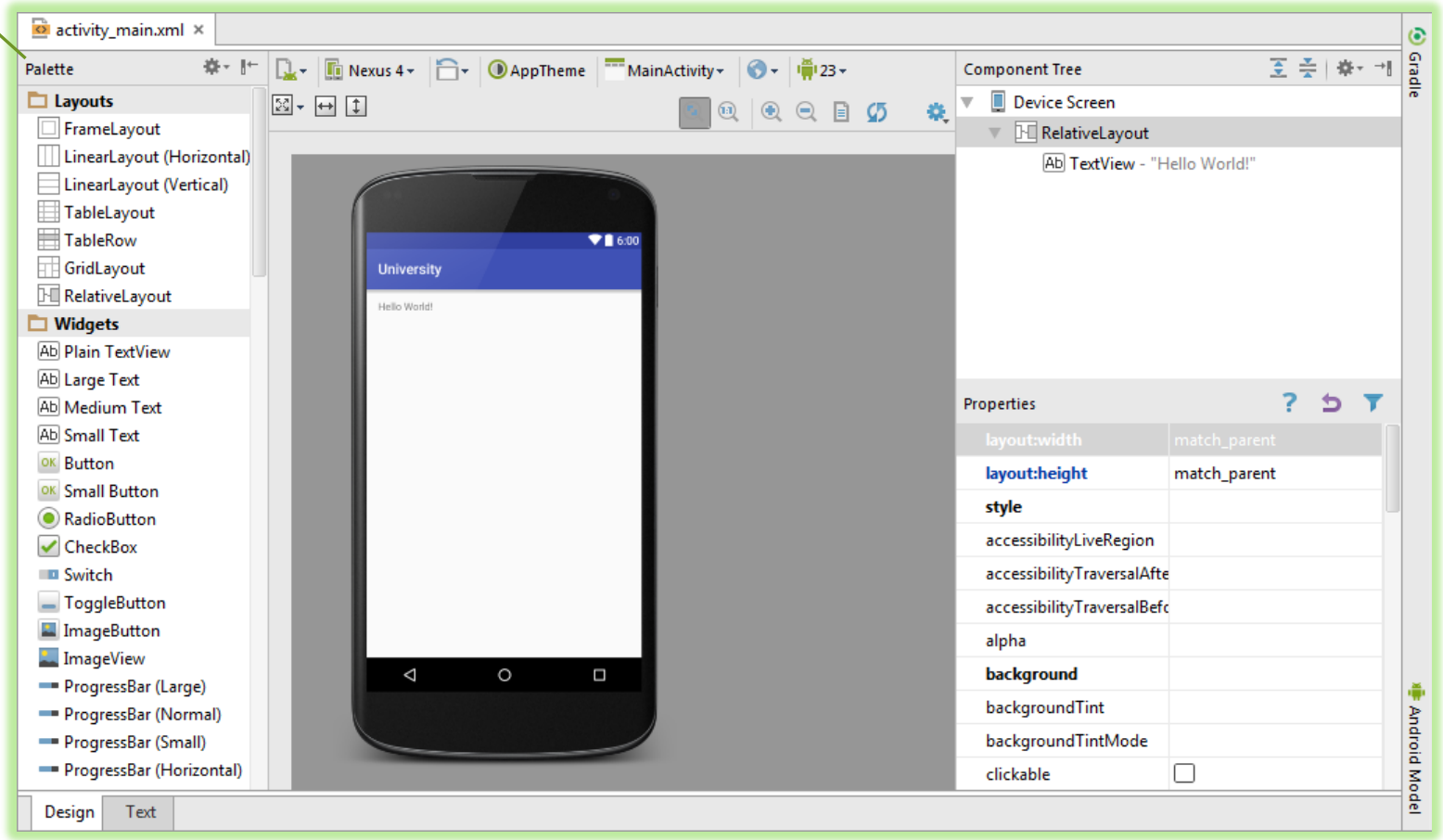
```java
package net.harifi.university;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

بررسی فایل لایه UI برنامه



Palette

بررسی فایل لایه UI برنامه



کار در کلاس

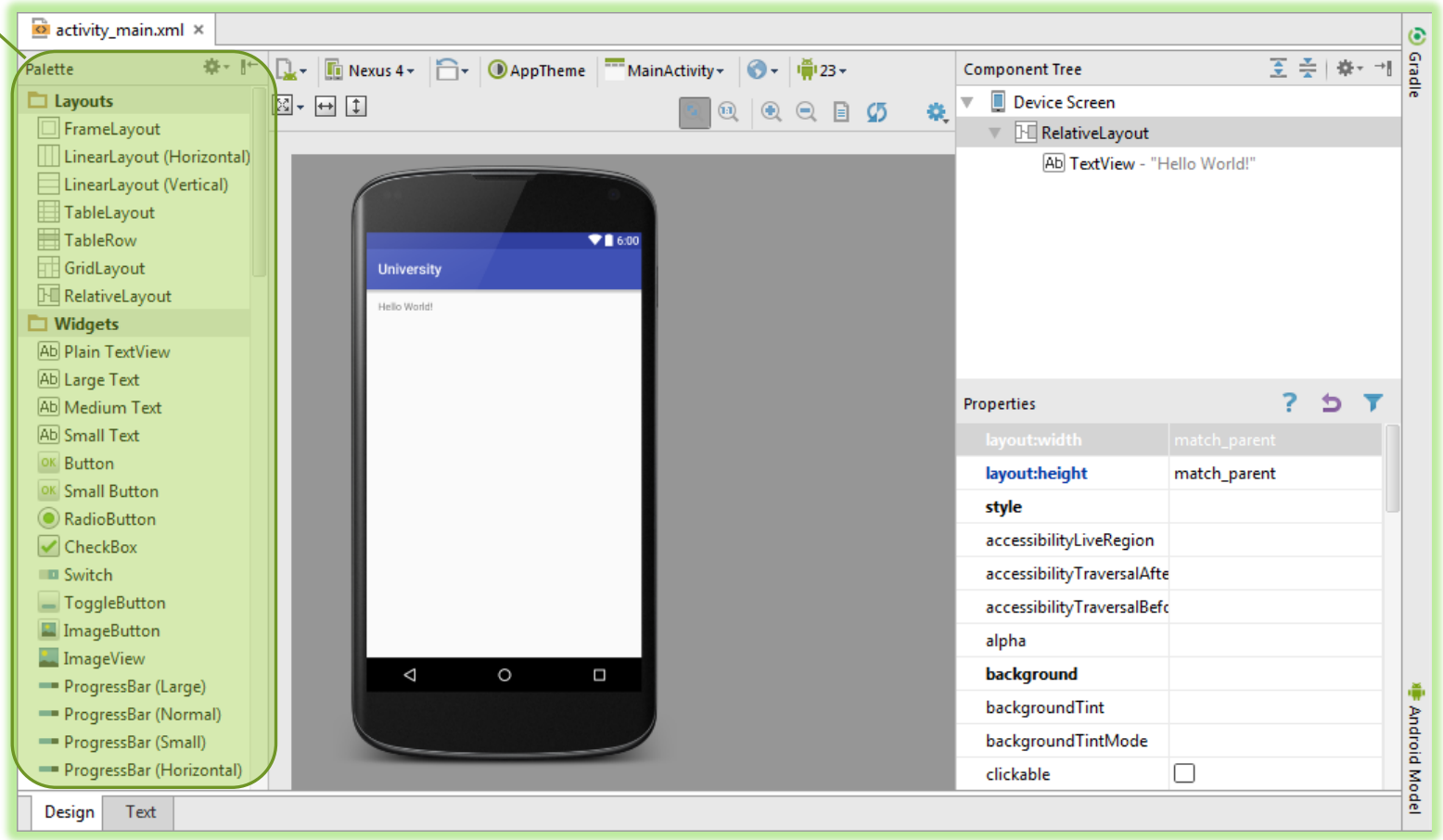
مراحل ایجاد پروژه در
اندروید استودیو را به
صورت عملی انجام داده و
قسمت های مختلف توصیف
شده در این جلسه از کلاس
را مشاهده کنید.

شروع کار با اندروید استودیو

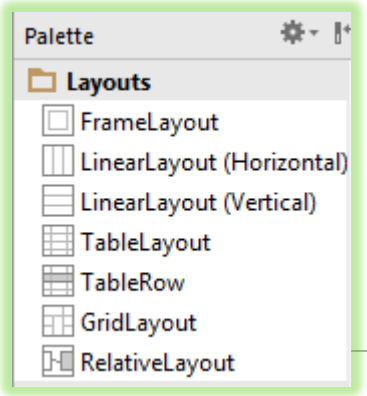
- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها

Palette

بررسی Palette ها

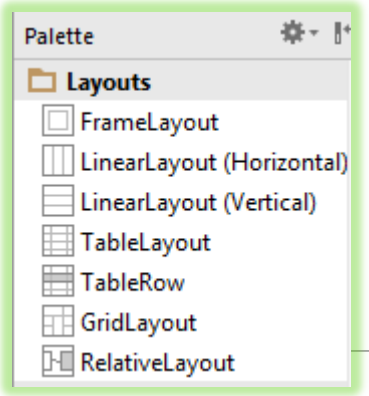


دسته بندی Palette ها در اندروید استودیو



بررسی Palette ها – Layouts

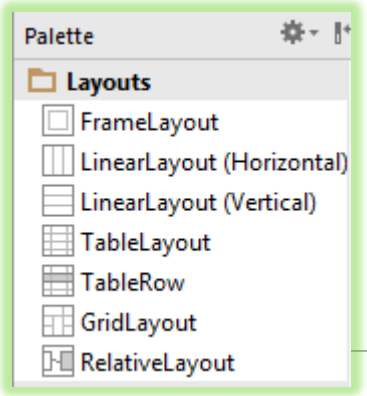
- FrameLayout - یک حفره روی صفحه ایجاد می کند که می توان از آن برای نمایش یک view استفاده کرد.
- LinearLayout - یک ViewGroup است که تمامی ویجت ها و لایه هایی که داخل آن هستند را به صورت عمودی و یا افقی تراز می کند.
- TableLayout - یک View است که ویجت ها و لایه های داخل خود را به صورت سطری و ستونی نشان می دهد.
- RelativeLayout - یک ViewGroup است که تمامی ویجت ها و لایه هایی که داخل آن هستند را به صورت وابسته به هم نشان می دهد.



بررسی Palette ها – Layouts

- برخی صفت های یک Layout

صفت	توضیح
android:id	یک نام منحصر به فرد برای هر View
android:layout_width	پهنا یک لایه را مشخص می کند
android:layout_height	ارتفاع یک لایه را مشخص می کند
android:layout_marginTop/Bottom	فاصله اضافی از بالا/پایین یک لایه
android:layout_marginLeft/Right	فاصله اضافی از سمت چپ/راست یک لایه
android:layout_gravity	مشخص می کند که فرزند لایه در چه موقعیتی قرار بگیرد
android:paddingTop/Bottom	فاصله از بالا/پایین برای پر کردن لایه
android:paddingLeft/Right	فاصله از سمت چپ/راست برای پر کردن لایه

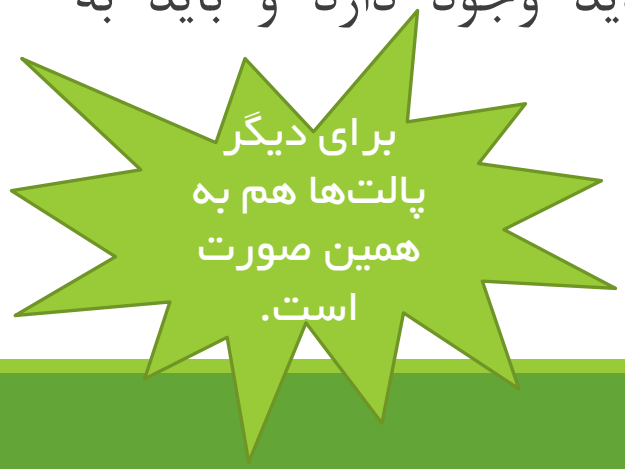


بررسی Palette ها – Layouts

- نحوه ID دادن در Layout ها

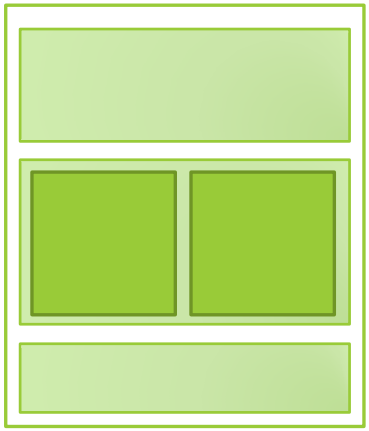
```
android:id="@+id/my_Layout"
```

- نماد (@) که در ابتدای رشته مربوط به نام گذاری قرار دارد، به تحلیلگر فایل XML اطلاع می دهد که این نوشته به عنوان ID در نظر گرفته شود.
- نماد (+) مشخص می کند که یک ID جدید وجود دارد و باید به resource ID ها اضافه شود.



بررسی Palette ها – Layouts
LinearLayout

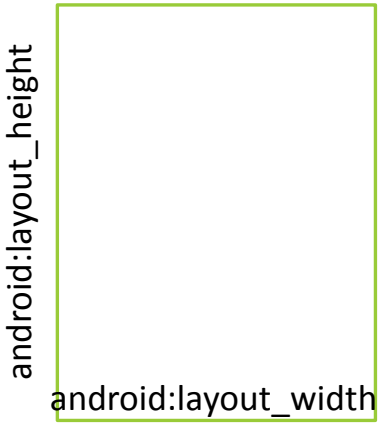
- در یک اکتیویتی می توان چندین لایه داشت. می توان ترکیبی از لایه ها را نیز داشت.
- چیدمان لایه ها به هر تعداد و با هر شکل می توانند با هم ترکیب شوند.
- مثلاً می توان داخل یک LinearLayout عمودی، تعدادی LinearLayout افقی داشت.



Layouts – ها Palette بررسی
LinearLayout

- ویژگی های چیدمان خطی (افقی/عمودی)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
...
</LinearLayout>
```



android:layout_height

android:layout_width

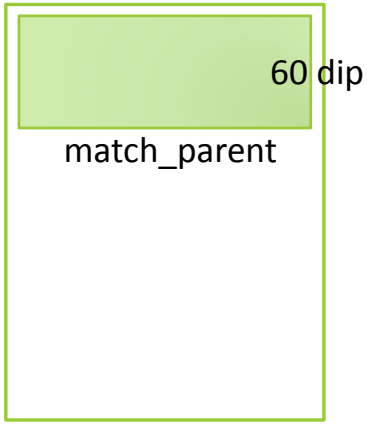
بررسی Layouts – ها Palette LinearLayout

- ویژگی های چیدمان خطی (افقی/عمودی)
  - fill_parent یا match_parent - از این مقدار برای وقتی استفاده می شود که می خواهیم LinearLayout به اندازه جایی که والد آن اجازه داده است بزرگ شود. اگر یک Layout والد نداشته باشد، این مقدار برابر با عرض یا ارتفاع صفحه نمایش خواهد شد.
  - wrap_content - از این مقدار وقتی استفاده می شود که می خواهیم یک Layout به اندازه ای بزرگ شود که تمام Layoutها و Viewهای درونی اش، داخل آن جا شوند.
  - یک عدد صحیح با واحد dp یا dip – برای مشخص کردن طول یا عرض مشخص با واحد dp.

dip مخفف Density Independent Pixel است به معنی پیکسل مستقل از تراکم صفحه نمایش.
در یک دستگاه با تراکم پیکسلی (رزولوشن) پایین، یک dip برابر یک پیکسل است اما در یک دستگاه با تراکم پیکسلی خیلی بالا، یک dip سه یا چهار پیکسل خواهد بود.

بررسی Palette ها – Layouts
LinearLayout

- مثال: ساخت لایه ای با ویژگی های زیر:




```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="60dip"
   android:orientation="horizontal" >
</LinearLayout>
```

بررسی Layouts – ها Palette
مزایا و معایب LinearLayout

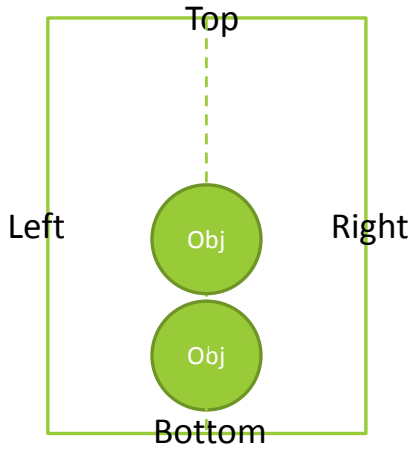
- مزایا:
  - سادگی فراوان: چیدمان خطی ساده است. عناصر داخل آن به ترتیبی که در فایل XML ظاهر می شوند، در Layout هم قرار می گیرند؛ چیدمان خطی ساختار ساده ای دارد و به سادگی می توان آن در ذهن تصور کرد.
  - قابلیت تغییر ساده: به علت سادگی فراوان این چیدمان، تغییر دادن آن هم بسیار ساده است.
- معایب:
  - ممکن نبودن ساخت برخی چیدمان ها: سادگی زیاد چیدمان خطی و محدودیت امکانات آن باعث می شود که نتوان برخی چیدمان ها که در برنامه های مختلف اندروید دیده می شود، صرفاً با اتکا به چیدمان خطی و ویژگی های آن بسازیم.
  - حجم زیاد کد برای ساخت چیدمان ها: اگر بخواهیم یک رابط کاربری پیچیده و پر از View های مختلف که به شدت در هم آمیخته شده اند، با LinearLayout بسازیم، باید Layout های زیادی را به صورت تو در تو به کار ببریم که باعث حجم بالای کد می شود.

کار در کلاس

چند LinearLayout ایجاد کنید و آن ها را با هم تراز کنید و درون آن ها ویجت هایی را قرار دهید.

بررسی Layouts – های Palette RelativeLayout

- نوعی چیدمان است که جای عناصر در آن به دقت، نسبت به والد (Parent) یا خواهرانش (Sibling) مشخص می شود.
- هر RelativeLayout دارای قسمتهایی است که فرزندانش می توانند جای خود را بر اساس آن ها مشخص کنند.



مثال درج آبجتی دیگر نسبت به آبجکت اولیه بر اساس خواهر:

```
android:layout_above="@+id/anotherView"
android:layout_below="@+id/anotherView"
android:layout_toRightOf="@+id/anotherView"
android:layout_toLeftOf="@+id/anotherView"
```

مثال درج آبجت بر اساس والد:

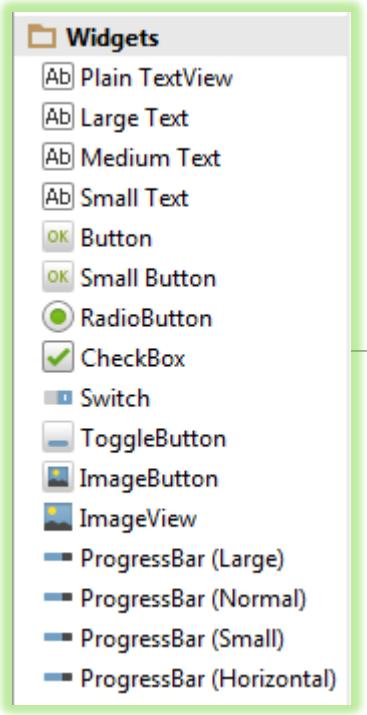
```
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
```

بررسی Layouts – ها Palette
مزایا و معایب RelativeLayout

- مزایا:
  - امکانات فراوان: امکانات خیلی بیشتری نسبت به LinearLayout دارد. این امکانات بیشتر، عملاً دو مزیت زیر را به ارمغان می‌آورد.
    - امکان ساخت برخی چیدمان ها که در LinearLayout نشدنی است: امکانات فراوان RelativeLayout باعث می شود که برخی چیدمان ها که آن ها را نمی توان با LinearLayout ساخت، با اندکی ذوق و سلیقه و کمی تجربه، به سادگی با RelativeLayout ساخت.
    - حجم کم کد برای ساخت چیدمان های پیچیده: چیزی که بسیار مشخص است این است که برای ساخت یک چیدمان مشابه، اگر از RelativeLayout استفاده کنیم، حجم کد تولید شده بسیار کمتر از وقتی است که از LinearLayout استفاده می کنیم.
- معایب:
  - پیچیده بودن کد XML چیدمان: همانطور که بیان شد، کد تولید شده برای RelativeLayout کمتر از کد LinearLayout است ولی این کمتر کد، نتیجه معکوسی در سادگی کد دارد. هر چه که امکانات بیشتر می شود، حجم کد کمتر ولی پیچیدگی آن بیشتر می شود.
  - سخت بودن تغییر چیدمان طراحی شده: بیان شد که با RelativeLayout می توان چیدمان هایی ساخت که با LinearLayout یا سخت است یا ممکن نیست. از سوی دیگر چیدمانی که با RelativeLayout ساخته می شود را نمی‌توان به سادگی تغییر داد. فرض کنید یک View در Layout خود دارید که چند View دیگر، به شدت به آن وابسته‌اند. اگر بخواهید آن View را حذف کنید، کل چیدمان شما به هم می ریزد.

کار در کلاس

چند RelativeLayout ایجاد کنید و آن ها را با هم تراز کنید و درون آن ها ویجت هایی را قرار دهید.



بررسی Widgets — ها Palette
معرفی برخی ویجت های پرکاربرد
TextView

- برای نشان دادن متن در UI استفاده می شود.

```
<TextView
   android:id="@+id/textView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Some Text Here" />
```

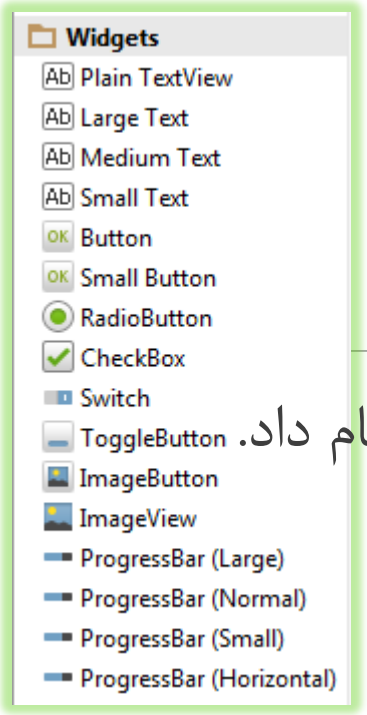
- برخی ویژگی های تکست ویو:
  - android:text – متنی که قرار است TextView نمایش دهد. در حال حاضر متن مستقیماً به عنوان مقدار این ویژگی نوشته می شود. (روش درج متن در فایل xml)

بررسی Palette Widgets — ها
معرفی برخی ویجت های پرکاربرد
TextView

- برخی ویژگی های تکست ویو:
- android:textColor - رنگ متن را مشخص می کند. رنگ را می توانیم به چند روش معرفی کنیم که ساده ترین آن استفاده از روش کد گذاری ARGB است. در این روش هر رنگ، ترکیبی از سه رنگ اصلی قرمز، سبز و آبی و میزان آلفا (شفافیت) است. هر رنگ با یک عدد شانزده تایی (هگزادسیمال) معرفی می شود و مقادیر مجاز از ۰۰ (مقدار صفر در سیستم ده دهی) تا FF (مقدار ۲۵۵ در سیستم ده دهی) است. در این روش مقدار شفافیت یا آلفا اختیاری است. برای نمونه تعریف رنگ سفید بدون شفافیت در این روش #FFFFFF است و تعریف رنگ سبز با میزان شفافیت ۵۰ درصد #7F00FF00 است.

بررسی Palette Widgets — ها
معرفی برخی ویجت های پرکاربرد
TextView

- برخی ویژگی های تکست ویو:
  - android:textSize - اندازه قلم یا فونت را مشخص می کند. واحد آن در اندروید sp است. sp شبیه به dip است با این ویژگی خاص که اگر کاربر اندازه فونت گوشی خود را تغییر دهد، این مقدار هم به تناسب تغییر می کند.
  - android:textStyle - حالت فونت TextViewرا تغییر می دهید. مقادیر مجاز bold و italic و bolditalic هستند.
  - android:gravity - متن داخل TextView به صورت پیش فرض چپ به راست. اگر بخواهید متن نمایش داده شده در TextView از سمت راست نمایش داده شود، می توانید gravity آن را right انتخاب کنید. عملکرد این ویژگی شبیه عملکرد Gravity در LinearLayout است.



بررسی Widgets – ها Palette
معرفی برخی ویجت های پرکاربرد
Button

- عناصری که با کلیک کردن بر روی آن هر عملی را می توان انجام داد.

```
<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Some Text Here" />
```

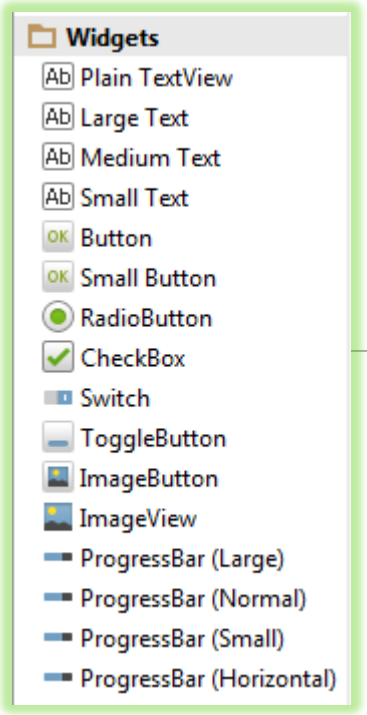
- ویژگی هایی شبیه به تکست ویو دارد.
- برخی ویژگی های دکمه:
  - android:background – با استفاده از این ویژگی، می توان یک رنگ، عکس، کشیدنی (drawable) را به عنوان پس زمینه دکمه تعریف کنید.

بررسی Widgets – Palette ها
معرفی برخی ویجت های پرکاربرد
Button

- برخی ویژگی های دکمه:
  - android:enabled – دکمه ها به صورت پیش فرض کلیک شدنی هستند. ولی می توان با فرستادن مقدار false به این ویژگی دکمه را غیر فعال کرد.
  - android:onClick – اگر بخواهیم یک تابع تعریف کنید تا هر زمان که روی دکمه کلیک شد آن تابع صدا زده شود، نام تابع در این ویژگی نوشته می شود.
  - android:soundEffectsEnabled – به صورت پیش فرض با کلیک کردن بر روی دکمه ها صدایی پخش می شود که حس کلیک کردن بر روی دکمه را به کاربر القا می کند. با استفاده از این ویژگی می توان این صدا را فعال یا غیر فعال کرد.

بررسی Palette Widgets – ها
معرفی برخی ویجت های پرکاربرد
Button

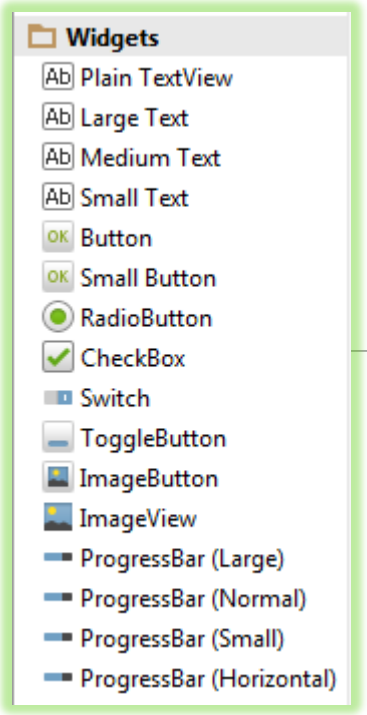
- برخی ویژگی های دکمه:
  - android:drawableTop – به غیر از تغییر پس زمینه (Background) یک دکمه، می‌توان یک تصویر را در دکمه در کنار متن نشان داد. اگر بخواهیم این تصویر بالای متن دکمه قرار بگیرد، می توان عکس یا کشیدنی (drawable) مربوطه را از طریق این ویژگی معرفی کرد.
  - android:drawableBottom – مشابه قبلی است با این تفاوت که عکس را در پایین نوشته نشان می دهد.
  - android:drawableLeft – مشابه قبلی است با این تفاوت که عکس را در سمت چپ نوشته نشان می دهد.
  - android:drawableRight – مشابه قبلی است با این تفاوت که عکس را در سمت راست نوشته نشان می دهد.



بررسی Widgets – ها Palette
معرفی برخی ویجت های پرکاربرد
CheckBox

```
<CheckBox
   android:id="@+id/checkBox1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="I agree." />
```

- چک باکس یک دکمه است با یک متن و یک جعبه که کاربر می تواند را تیک بزند یا نزند.
- ویژگی های چک باکس شبیه TextView و Button است. اگر مقدار ویژگی android:checked برابر true باشد، چک باکس تیک خورده و در غیر این صورت عادی (بدون تیک) نشان داده می شود.

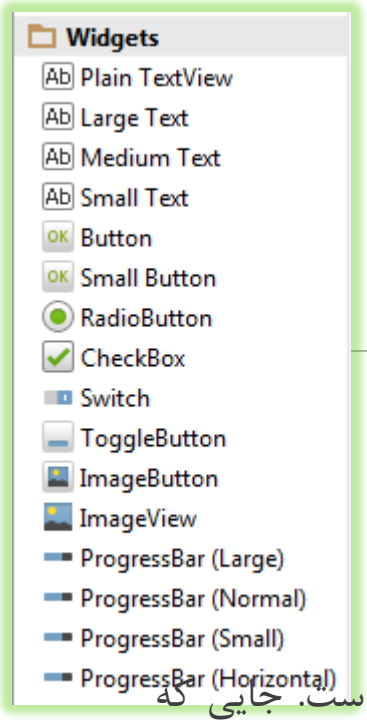


بررسی Widgets – ها Palette
معرفی برخی ویجت های پرکاربرد
RadioButton

```
<RadioButton
   android:id="@+id/radioButton1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="RadioButton" />
```

- دکمه رادیویی یا RadioButton شبیه چک باکس است. تنها تفاوت در این است که چندین دکمه رادیویی را می توان با هم در یک گروه قرار داد. در این حالت از میان دکمه های رادیویی گروه شده با هم، در هر زمان فقط یکی تیک خورده می‌شود. (برای گروه بندی از RadioGroup استفاده می شود)

- ویژگی های دکمه رادیویی شبیه TextView و Button و CheckBox است.



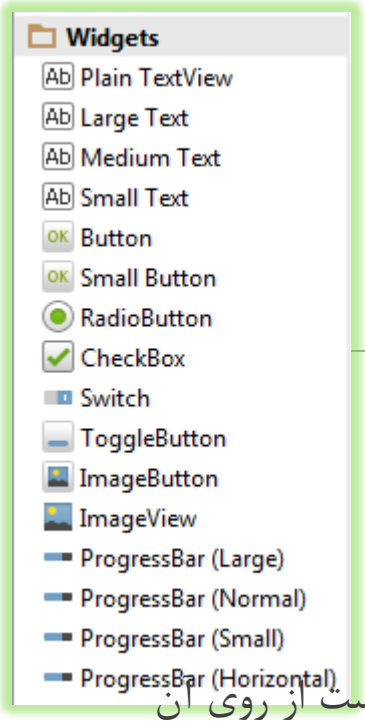
بررسی Widgets – های Palette
معرفی برخی ویجت های پرکاربرد
Switch

```
<Switch
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="false"
   android:text="WiFi" />
```

- کارکرد سوییچ شبیه کارکرد چک باکس است. بیشترین استفاده آن در تنظیمات برنامه ها است. جایی که می توان یک ویژگی را فعال یا غیرفعال کرد.

- مهمترین ویژگی های سوییچ:
  - android:text – متنی که در کنار سوییچ نشان داده می شود. در مثال فوق مشخص می باشد.
  - android:textOn – متنی که در حالتی که android:checked برابر true باشد، بر روی سوییچ نمایش داده می شود.
  - android:textOff – متنی که در حالتی که android:checked برابر false باشد، بر روی سوییچ نمایش داده می شود.

تذکر: Switch مختص اندرویدهای ۴ به بعد است و در نسخه های قبلی اندروید وجود ندارد. بنابراین فقط در برنامه هایی می توانید از آن استفاده کنید که minSdkVersion آن ها ۱۴ یا بیشتر باشد.

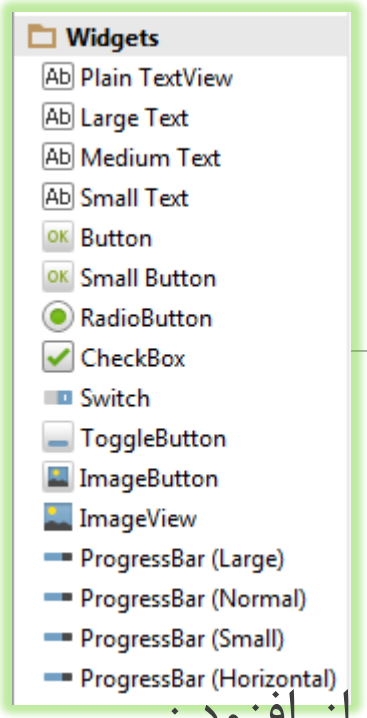


بررسی Widgets – های Palette
معرفی برخی ویجت های پرکاربرد
ToggleButton

```
<ToggleButton
   android:id="@+id/toggleButton1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="ToggleButton" />
```

- این نوع دکمه، دو حالت دارد. فشرده و عادی. دکمه معمولی (Button) به محض این که دست از روی آن برداشته می شود، از حالت فشرده به حالت عادی بر می گردد. اما ToggleButton وقتی که دست از روی آن برداشته می شود، حالت خود را از دست نمی دهد. (شکل ToggleButton در API های مختلف متفاوت است)

- ویژگی ها شبیه به TextView و Button است. برخی دیگر از ویژگی ها:
  - android:checked – اگر true باشد، وضعیت دکمه در حالت فشرده و در غیر این صورت در حالت عادی خواهد بود.
  - android:textOn – متن دکمه برای وقتی که در حالت فشرده است.
  - android:textOff – متن دکمه برای وقتی که در حالت عادی است.



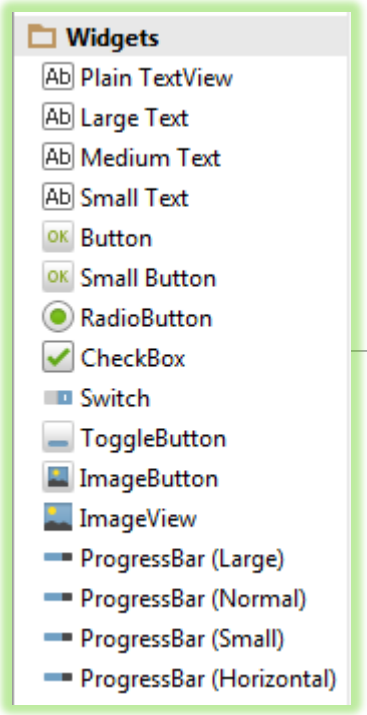
بررسی Widgets – های Palette
معرفی برخی ویجت های پرکاربرد
ImageView

```
<ImageView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/ic_launcher" />
```

- برای نمایش عکس در UI از این View استفاده می شود. بعد از افزودن یک ImageView به صفحه، کد بالا ایجاد می شود.
- ویژگی های مهم:
  - android:src – ارجاع به عکسی که قرار است نمایش داده شود.

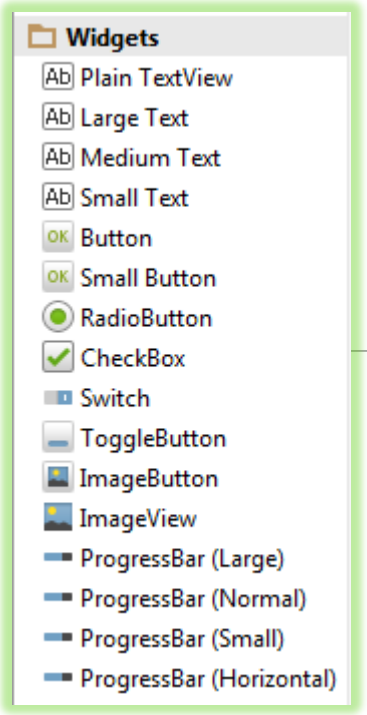
بررسی Widgets – Palette ها
معرفی برخی ویجت های پرکاربرد
ImageView

- ویژگی های مهم:
  - android:scaleType - این ویژگی مشخص می کند که عکسی که قرار است در ImageView نمایش داده شود، در صورت نیاز به تغییر اندازه یا جابجایی، چگونه باید رفتار کند. از مقادیر مجاز این ویژگی به برخی اشاره می کنیم:
    - center - عکس را در مرکز ImageView قرار می دهد ولی اندازه آن را تغییر نمی دهد (Scale نمی کند).
    - centerCrop - اگر عکس بزرگتر از ابعاد ImageView باشد، عکس را در مرکز قرار می دهد و قسمت های اضافه آن را می بُرد.
    - centerInside - اگر ابعاد عکس بزرگتر از ImageView باشد، با حفظ نسبت عرض و ارتفاع، عکس را تغییر اندازه می دهد تا به صورت کامل در ImageView قرار بگیرد. اگر اندازه عرض و ارتفاع عکس کمتر از ImageView باشد، تغییر اندازه رخ نمی دهد. در نهایت عکس در وسط ImageView قرار می گیرد.
    - fitCenter - شبیه قبلی است با این تفاوت که اگر اندازه عکس کمتر از ImageView باشد، با حفظ نسبت عرض و ارتفاع، عکس را بزرگتر می کند تا هم اندازه ImageView شود.
    - fitStart - شبیه fitCenter عمل می کند فقط در نهایت عکس را در بالا سمت چپ ImageView قرار می دهد.
    - fitEnd - شبیه fitCenter عمل می کند فقط در نهایت عکس را در پایین سمت راست ImageView قرار می دهد.
    - fitXY - بدون حفظ نسبت عرض و ارتفاع عکس را تغییر اندازه می دهد تا به صورت کامل در ImageView قرار بگیرد.



بررسی Widgets – ها Palette
معرفی برخی ویجت های پرکاربرد
ImageButton

- تمامی ویژگی های ImageView و Button را با هم دارد.
- تفاوت آن با ImageView این است که تصویر در آن به صورت دکمه در خواهد آمد و کاربر می تواند برای روی آن کلیک کند.



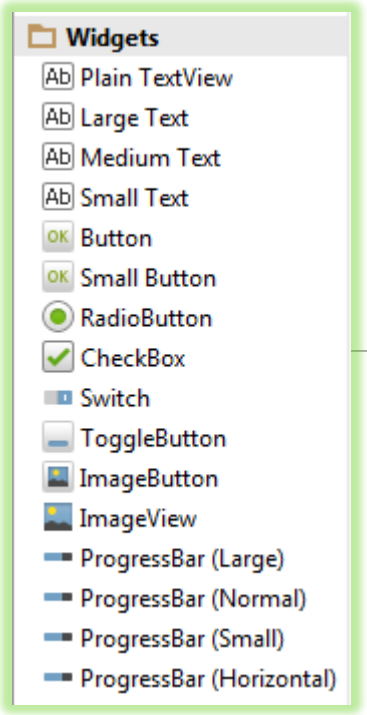
بررسی Widgets – ها Palette
معرفی برخی ویجت های پرکاربرد
ProgressBar

```
<ProgressBar
   android:id="@+id/progressBar1"
   style="?android:attr/progressBarStyleHorizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:progress="45" />
```

- ProgressBar نواری است که میزان پیشرفت کاری را نشان می دهد. مثال: فرض کنید می خواهیم اطلاعاتی از وب دریافت کرده و به کاربر نمایش دهیم. اینکه دریافت اطلاعات در چه مرحله ای است را با ProgressBar نشان می دهیم.

- ProgressBar حالات مختلف دارد. مانند خطی و یا دایره ای و ... (مثال فوق از نوع خط افق است.)

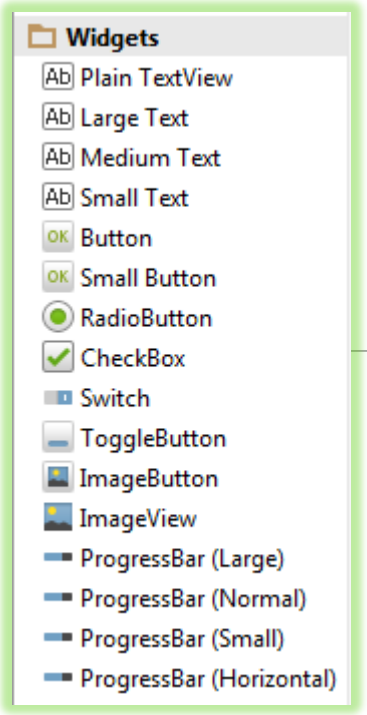
```
style="?android:attr/progressBarStyleLarge"
```



بررسی Widgets — ها Palette
معرفی برخی ویجت های پرکاربرد
RatingBar

```
<RatingBar
    android:id="@+id/ratingBar1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
    android:rating="1.5"/>
```

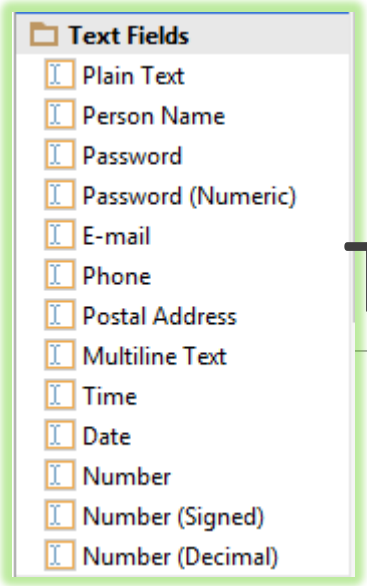
- از RatingBar برای امتیاز دادن به چیزی استفاده می شود. مثلاً فرض کنید از کاربران می خواهید به مطلب شما امتیاز بدهند. این امتیاز در قالب ستاره است. برای این کار از RatingBar استفاده می شود.
- برخی ویژگی های مهم RatingBar:
  - android:numStars - تعداد ستاره های RatingBar را می توانید با استفاده از این ویژگی تعیین کنید. مقدار پیش فرض ۵ است.
  - android:rating - در صورت اضافه کردن این ویژگی، امتیاز روی ستاره ها نشان داده می شود. مثلاً اگر تعداد ستاره ها ۵ باشد و ما مقدار این ویژگی را ۱.۵ اعلام کنیم، ۱.۵ ستاره به رنگ آبی در می آیند و امتیاز را نشان می دهند.



بررسی Widgets – Palette ها
معرفی برخی ویجت های پرکاربرد
WebView

کاربرد عملی وب ویو
در جلسات آینده
ارائه خواهد شد.

- وب ویو یکی از مهم ترین و کاربردی ترین Viewهای اندروید است.
- این View نسخه ای کوچک شده و بهینه شده از یک مرورگر اینترنتی است.
- می توان در یک WebView هر نوع محتوای HTML را نمایش دهید یا حتی آدرس یک صفحه اینترنتی را به آن داد تا محتوای آن صفحه را نمایش دهد.

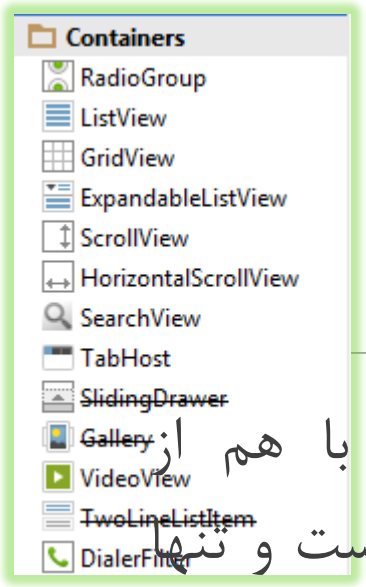


بررسی Palette ها – Text Fields

- تکست فیلدها، جایی هستند که کاربر متنی را وارد می کند.
- Viewیی که در این حالت به UI برنامه افزوده می شود، EditText است.
- وقتی که کاربر بر روی یک ادیت تکست کلیک می کند، صفحه کلید نمایش داده می‌شود و کاربر می تواند متن مورد نظر خود را وارد کند.
- همه EditText هایی که در اینجا معرفی شده اند، مشابه همدیگر هستند.
- فقط در یک ویژگی inputType با هم تفاوت دارند. این ویژگی مشخص می کند که برنامه باید انتظار چه نوع متنی را داشته باشد و به تناسب آن صفحه کلید را نمایش می‌دهد. مثلاً اگر مقدار inputType برابر phone باشد، صفحه کلید نمایش داده شده، کلیدهای عددی ۰ تا ۹ را به همراه کلیدهای * و # نمایش می دهد.

کار در کلاس

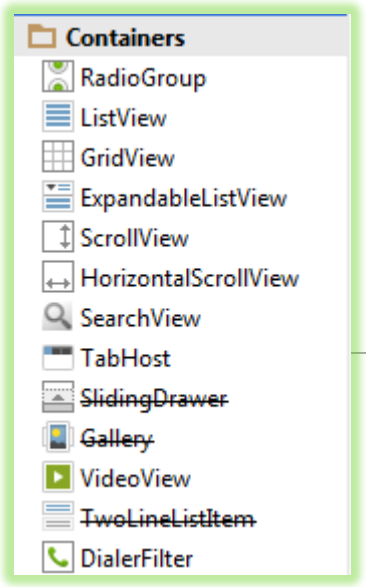
چند Widget و تکست فیلد ایجاد کنید و ویژگی های آن ها را بررسی کرده و مقایسه کنید.



بررسی Containers – های Palette
معرفی برخی کانتینرهای پرکاربرد
RadioGroup

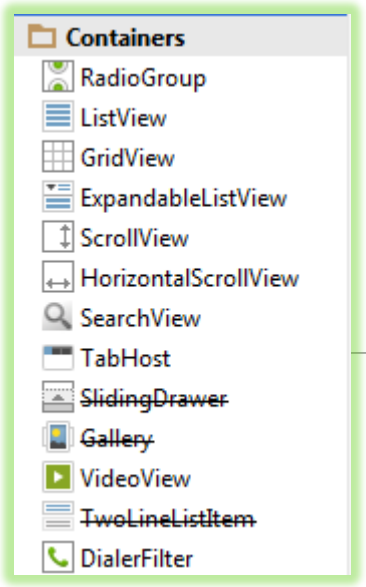
- برای گروه کردن چند دکمه رادیویی یا RadioButton با هم از RadioGroup استفاده می کنیم. RadioGroup قابل دیدن نیست و تنها خاصیتی که دارد، گروه کردن RadioButton ها است. مثال:

```
<RadioGroup
   android:id="@+id/radioGroup1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" >

   <RadioButton
      android:id="@+id/radio0"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:checked="true"
      android:text="Option One" />

    <RadioButton
      android:id="@+id/radio1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Option Two" />

   <RadioButton
      android:id="@+id/radio2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Option Three" />

</RadioGroup>
```

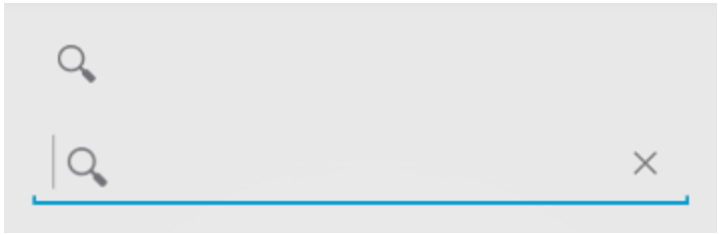


بررسی Containers – های Palette
معرفی برخی کانتینرهای پرکاربرد
ListView و ExpandableListView

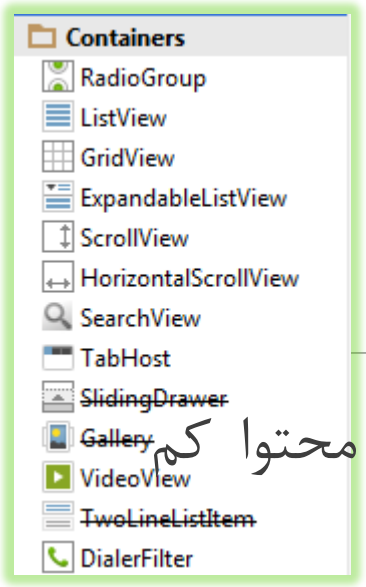
- ListView
  - برای نمایش یک لیست از اطلاعات از ListView استفاده می شود.
  - این لیست می تواند آرایه ای از String باشد یا لیستی از یک شیء خاص.
  - برای استفاده از لیست ویو لازم است که کد جاوا نوشته شود.
  - معمولاً از ListView برای نمایش اطلاعاتی که از پایگاه داده به دست می آید استفاده می‌شود.
- ExpandableListView
  - بسیار شبیه ListView است. با این تفاوت که هر کدام از عناصر آن، مثل یک ListView مستقل هستند. برای مثال با استفاده از کد نویسی، هر آیتم این لیست می تواند نام یک کشور باشد و با کلیک کردن بر روی نام هر کشور، آیتم باز شده و لیست شهرهای آن کشور نمایش داده شود.



بررسی Containers – های Palette
معرفی برخی کانتینرهای پرکاربرد
SearchView



- نوع خاصی از EditText است که برای جستجو به کار می رود.
- این View ابتدا بسته (collapsed) است و با کلیک کردن بر روی آن باز (expand) می شود.



بررسی Containers – ها Palette
معرفی برخی کانتینرهای پرکاربرد
ScrollView

- در صورت استفاده از ScrollView هر گاه جا برای نمایش محتوا کم باشد، کاربر می تواند اسکرول کند تا بتواند همه محتوا را ببیند.
- فرزند اسکرول ویو فقط می تواند یک آیتم یا یک لایه باشد.

```
<ScrollView
   android:layout_width="match_parent"
   android:layout_height="wrap_content" >
</ScrollView>
```

بررسی Containers – ها Palette
معرفی برخی کانتینرهای پرکاربرد
ScrollView

- مثال:

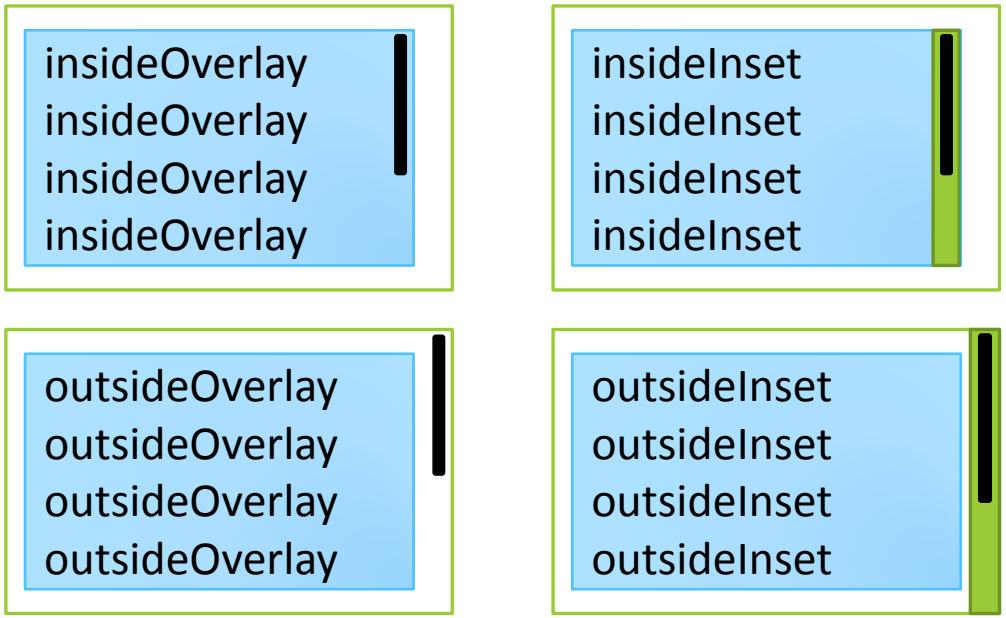
```
<ScrollView
   android:layout_width="match_parent"
   android:layout_height="wrap_content" >

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical" >

            <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="TextView 1" />

            <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="TextView 2" />
   </LinearLayout>
</ScrollView>
```

بررسی Containers – ها Palette
معرفی برخی کانتینرهای پرکاربرد
ScrollView

- مهم ترین ویژگی های ScrollView
  - android:fillViewport – این ویژگی به اسکرول ویو اجازه می دهد محتوای خود را بکشد تا با اندازه جایی که دارد بزرگ شود.
  - android:scrollbars – اگر مقدار این ویژگی را none تعریف کنیم، هیچگاه اسکرول بار نمایش داده نمی شود.
  - android:fadeScrollbars – اسکرول بار بعد از مدتی ناپدید می شود. اگر مقدار این ویژگی را false بگذارید، اسکرول بار همیشه نمایش داده خواهد شد.
  - android:scrollbarFadeDuration – مدت زمان ناپدید شدن اسکرول بار به صورت میلی ثانیه را می توان با این ویژگی تعیین کرد.

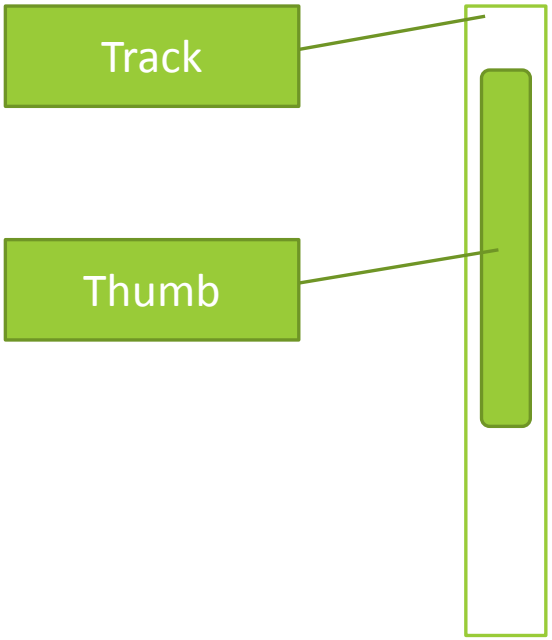
بررسی Containers – ها Palette
معرفی برخی کانتینرهای پرکاربرد
ScrollView

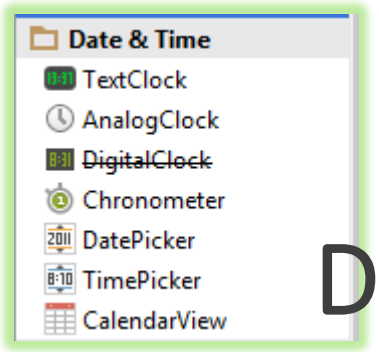
- مهم ترین ویژگی های ScrollView
  - android:scrollbarStyle - مکان اسکرول بار را تعیین می کند. مقادیر این ویژگی می تواند موارد زیر باشد:
    - insideOverlay
    - insideInset
    - outsideOverlay
    - outsideInset



بررسی Containers – ها Palette
معرفی برخی کانتینرهای پرکاربرد
ScrollView

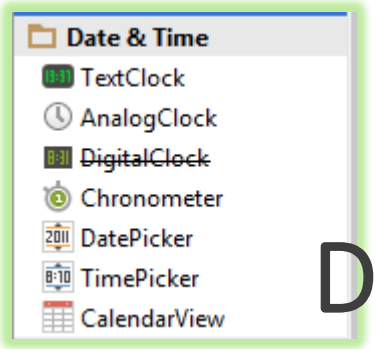
- مهم ترین ویژگی های ScrollView
  - android:scrollbarSize – اندازه اسکرول بار عمودی یا افقی را تعیین می کند.
    - android:scrollbarThumbHorizontal
    - android:scrollbarThumbVertical
    - android:scrollbarTrackHorizontal
    - android:scrollbarTrackVertical





بررسی Palette های Date & Time –

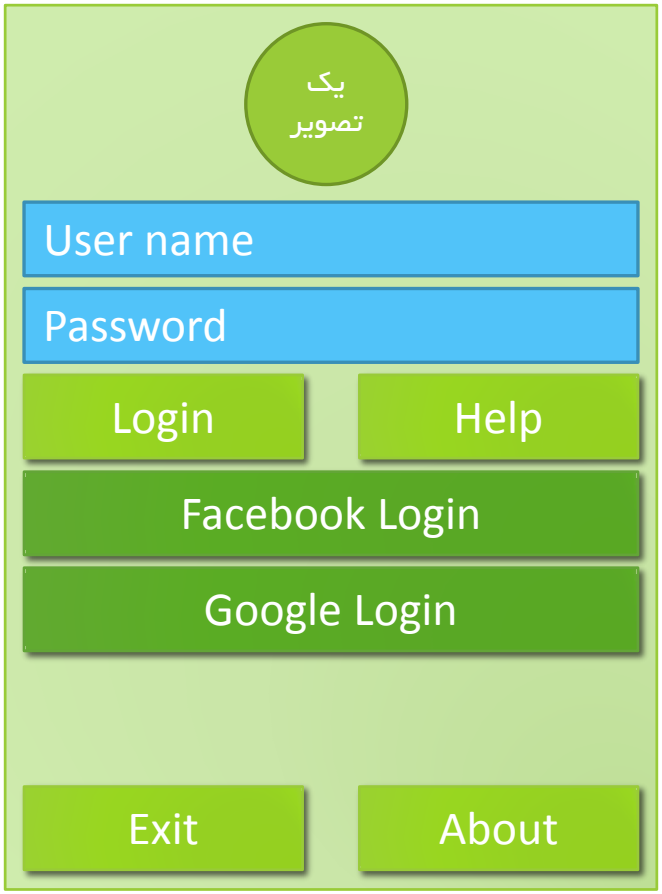
- در پالت Date and Time ویوهای مرتبط با تاریخ و ساعت قرار دارند.
- این View ها علیرغم این که به شدت کاربردی و مورد نیاز هستند، به علت تفاوت ساعت و تقویم ایرانی، معمولاً قابل استفاده در برنامه های ایرانی و فارسی نیستند.
- TimePicker – از این View برای انتخاب زمان استفاده می شود. می‌توان زمان را به فرمت ۲۴ ساعته یا ۱۲ ساعته نمایش داد.
- DatePicker – از این View برای انتخاب تاریخ استفاده می شود. این View قابلیت این که تاریخ را بر مبنای تقویم کشورها و مناطق مختلف نمایش دهد، ندارد. این View در اندرویدهای با API بالاتر از ۱۱، به صورت پیش فرض یک CalendareView هم در کنار خود دارد.



بررسی Date & Time – ها Palette

- CalendarView - مثل DatePicker، از این View هم برای انتخاب یک تاریخ استفاده می شود. این ویو نیز قابلیت نمایش تقویم غیر انگلیسی را ندارد.
- Chronometer - از این View برای نمایش گذر زمان استفاده می شود. ویژگی های این View بسیار شبیه TextView است.
- AnalogClock - از این View برای نمایش یک ساعت آنالوگ استفاده می شود. این View همواره زمان فعلی سیستم را نمایش می دهد.
- DigitalClock - از این View برای نمایش ساعت فعلی سیستم به صورت یک ساعت دیجیتال استفاده می شود.
- TextClock – ساعت و یا تاریخ فعلی را به صورت رشته (String) نمایش می دهد.

کار در کلاس



فرم روبرو را با استفاده از
RelativeLayout یا
LinearLayout ایجاد نمایید.

شروع کار با اندروید استودیو

- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها
- تأثیرگذاری روی ویجت ها با کد نویسی جاوا

کد نویسی جاوا – ویجت TextView

- فرض کنیم در فایل XML تکست ویو با نام textView1 تعریف شده است. حال می خواهیم رشته ای را به تکست ویو نسبت دهیم تا در صورت اجرای برنامه رشته مورد نظر نمایش داده شود. کد جاوا به شرح زیر است:

- مرحله اول: تعریف رشته

```
String textView1_text = "This is my textView1 Text";
```

- مرحله دوم: تعریف تکست ویو و ارتباط ویوی آن با XML

```
TextView textView1 = (TextView) findViewById (R.id.textView1);
```

- مرحله سوم: ست کردن رشته به تکست ویو

```
textView1.setText(textView1_text);
```

کد نویسی جاوا – ویجت TextView

- نکته: کد جاوا تکست ویو در کلاس onCreate() اکتیویتی نوشته می‌شود.
- برخی متدهای تکست ویو در جاوا
  - رنگ متن/رشته

```
textView1.setTextColor(Color.RED);
// or
textView1.setTextColor(Color.parseColor("#FF0000"));
// or
textView1.setTextColor(Color.rgb(255,0,0));
```

  - فاصله بین خطوط

```
textView1. setLineSpacing(1.0f, 1.0f);          //single float, multi float
```

کد نویسی جاوا – ویجت TextView

- برخی متدهای تکست ویو در جاوا
- دریافت متن و انتساب آن به یک تکست ویوی دیگر

```
String textView2_text;
TextView textView1 = (TextView) findViewById(R.id.textView1);
textView2_text = textView1.getText().toString();
```

- تغییر اندازه فونت

```
TextView textView1 = (TextView) findViewById(R.id.textView1);
textView1.setTextSize(TypedValue.COMPLEX_UNIT_SP, 30);
```

کد نویسی جاوا – ویجت TextView

- برخی متدهای تکست ویو در جاوا
- راست چین، چپ چین، وسط چین

```
TextView textView1 = (TextView) findViewById(R.id.textView1);
textView1. setGravity(Gravity.CENTER_HORIZONTAL);
```

- انتقال به خط جدید در متن نمایش داده شده

```
String newline = "This is my \n new Line";
```

کد نویسی جاوا – ویجت TextView

- برخی متدهای تکست ویو در جاوا
- فاصله بین متن و لبه های تکست ویو

```
// setPadding(left, top, right, bottom)
textView1. setPadding(20, 15, 10, 0);
```

- چک کردن برابری متن یک تکست ویو به متن دیگر یا عبارت دلخواه (به عنوان شرط در ساختار شرطی یا متغیر boolean استفاده می شود)

```
// textView1.equals("myText")

if (textView1.equals("myText") {
...
}
```

به جای رشته، متغیر رشته نیز می تواند قرار گیرد.

کد نویسی جاوا – ویجت TextView

- برخی متدهای تکست ویو در جاوا
- چک کردن موجود بودن رشته در رشته ای دیگر (در شرط یا متغیر boolean قرار می گیرد)

```
Boolean b = textView1.contains("SasanHarifi");
```

- حذف چند کاراکتر از ابتدای یک رشته (در مثال زیر ۴ کاراکتر حذف می شود)

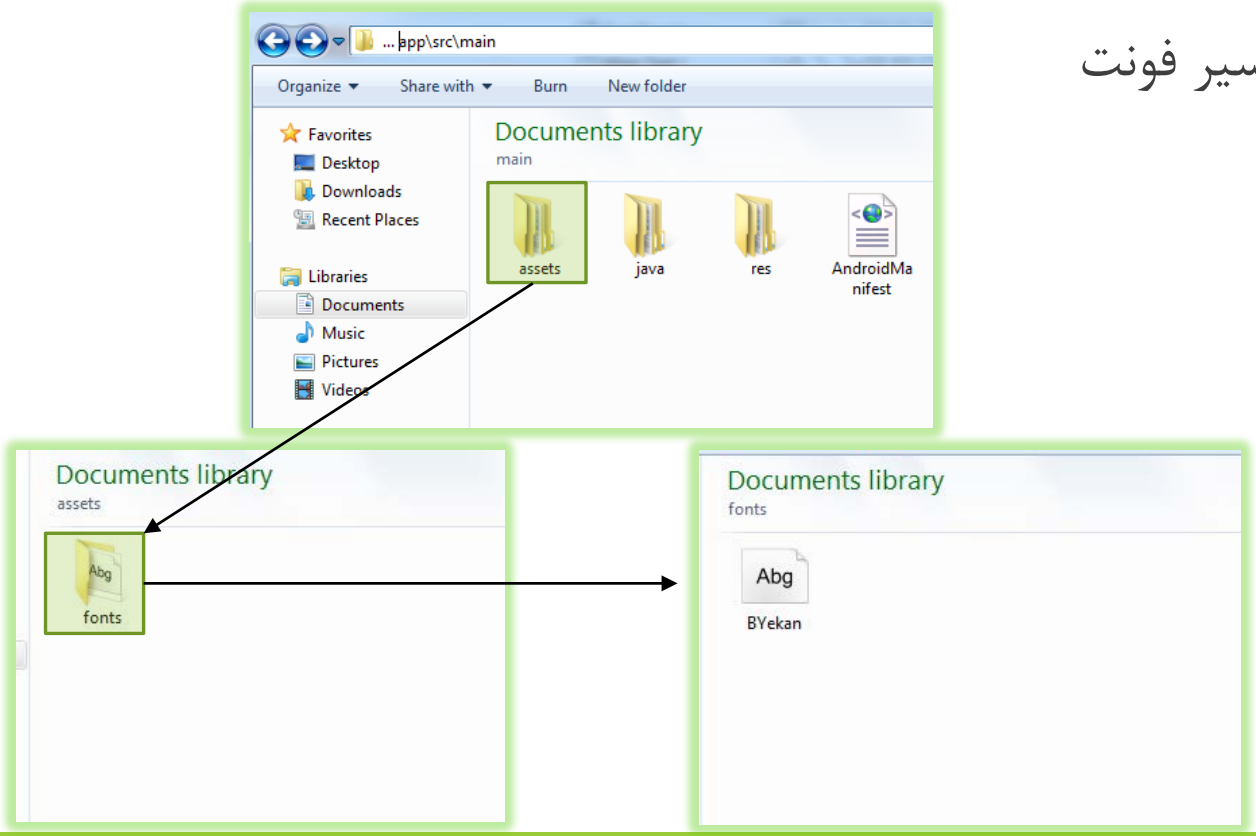
```
String textView2_text;
TextView textView1 = (TextView) findViewById(R.id.textView1);
textView2_text = textView1.getText().toString();
textView2_text = textView2_text .substring(4, textView2_text.length());
```

کد نویسی جاوا – ویجت TextView

- استفاده از فونت دلخواه در تکست ویو
  - برنامه برای استفاده از فونت دلخواه نیاز به فونت هایی با پسوند ttf دارد.
  - برای آدرس دهی فونت دلخواه در داخل برنامه نیاز به ایجاد فولدر assets داریم.
  - فولدر assets به معنی دارایی های خارج از برنامه می باشد که در کنار فایل مانیفست، فولدر java و فولدر res در مسیر src\main ساخته می شود.
  - در داخل فولدر assets فولدر دیگری به نام fonts ایجاد می شود که فونت مورد نظر در آن کپی می شود. (ایجاد فولدر فونت برای خوانایی پروژه است)

کد نویسی جاوا – ویجت TextView

- استفاده از فونت دلخواه در تکست ویو
- مسیر فونت



کد نویسی جاوا – ویجت TextView

- استفاده از فونت دلخواه در تکست ویو
- تعریف مسیر فونت در اکتیویتی

```
Typeface myFont = Typeface.createFromAsset(getAssets(), "fonts/BYekan.ttf");
```

- مثال

```
String textView3_text = "فونت دلخواه متن";
TextView textView3 = (TextView) findViewById(R.id.textView3);
textView3.setText(textView3_text);

Typeface myFont = Typeface.createFromAsset(getAssets(), "fonts/BYekan.ttf");

textView3.setTypeface(myFont);
```

کد نویسی جاوا – ویجت Button

- فرض کنیم در فایل XML دکمه (ImageButton) با نام button1 تعریف شده است. کد جاوا برای اینکه هنگام کلیک بر روی دکمه عملیاتی اتفاق بی افتد به شرح زیر است: کد جاوا دکمه در کلاس onCreate() اکتیویتی نوشته می‌شود.

- مرحله اول: تعریف دکمه و ارتباط ویوی آن با XML

```
ImageButton button1 = (ImageButton) findViewById (R.id.button1);
```

- مرحله دوم: تعریف لیسنر برای دکمه مورد نظر

```
button1.setOnClickListener (new View.OnClickListener() {
        public void onClick(View arg0){
            // my code or my action
        }
    });
```

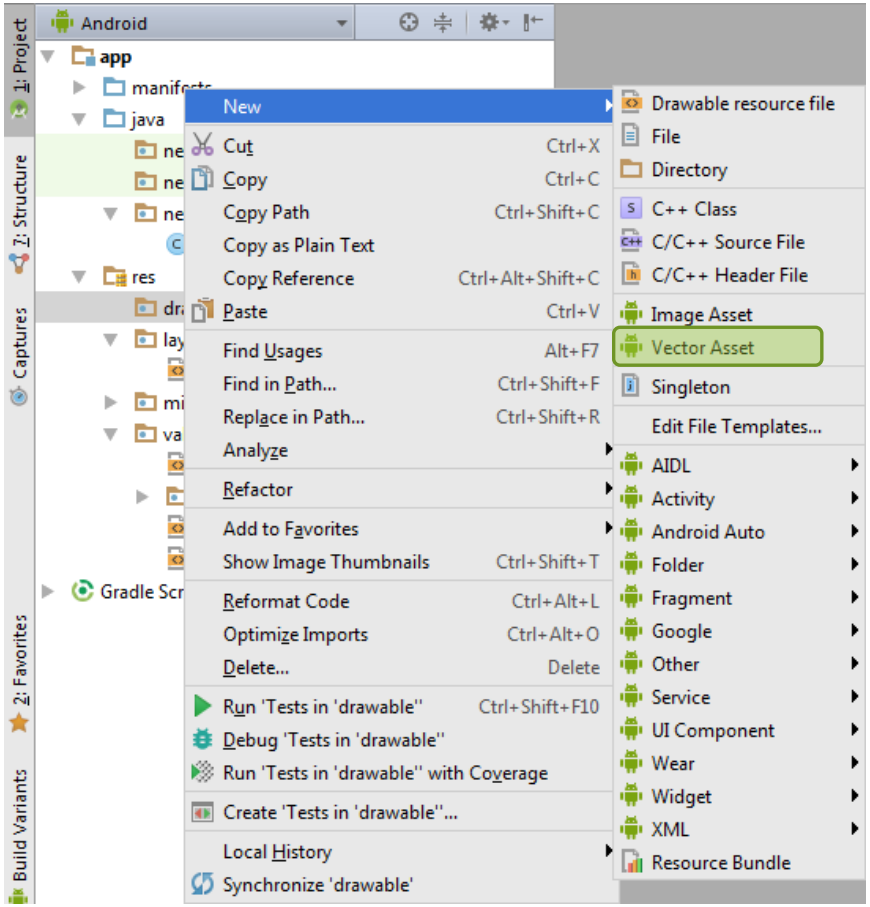
کد نویسی جاوا – ویجت Button

- لیسنر (Listeners) یکی از سه مفهوم اساسی مدیریت رویداد (Event) می‌باشد.
- واژه لیسنر به معنی شنونده و Event Listeners به معنی شنونده رویدادها می‌باشد.
- هنگامی که کاربر روی یک ویو (مثلا دکمه) کلیک می کند، این رویداد توسط سیستم عامل شنود می شود.
- توسط Event Listener Registeration رویداد ثبت می شود.
- توسط Event Handlers کنترل می شود.

کد نویسی جاوا – ویجت Button

- ایجاد دکمه های وکتوری
- مرحله اول: یافتن تصویر یک دکمه با فرمت svg مورد قبول برای اندروید استودیو.
- مرحله دوم: ایمپورت کردن دکمه در اندروید استودیو و تبدیل آن به xml
- کافیست روی دایرکتوری drawable راست کلیک و از منوی new گزینه vector asset را انتخاب کنیم.

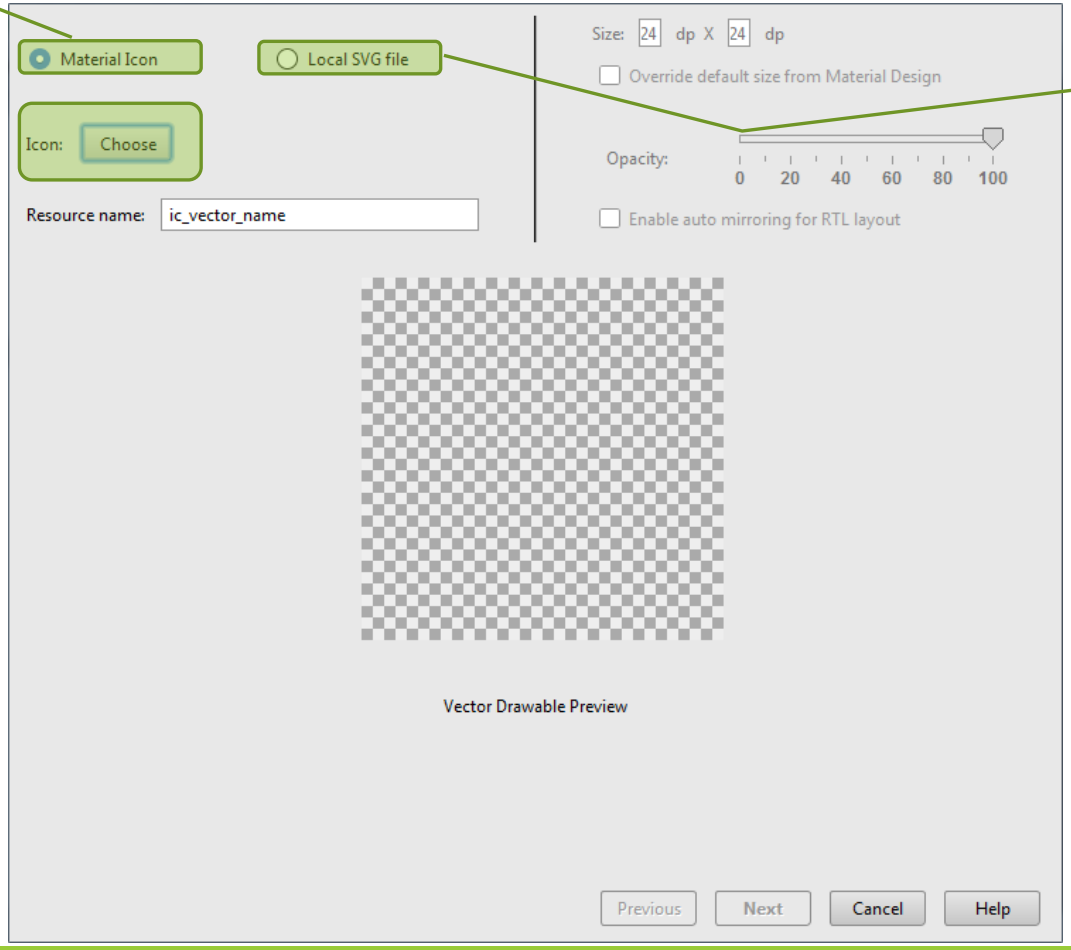
کد نویسی جاوا – ویجت Button



- ایجاد دکمه های وکتوری

کد نویسی جاوا – ویجت Button

Svg های پیش فرض و آماده



انتخاب svg قابل قبول از کامپیوتر

کد نویسی جاوا – ویجت Button
ساخت دکمه های داینامیک

- دکمه داینامیک دکمه ای است که هنگام کلیک بر روی آن دچار تغییراتی مثل تغییر رنگ یا اندازه و ... می شود.
- برای ساخت دکمه داینامیک نیاز به حداقل دو تصویر دکمه داریم.
- مراحل ایجاد:
  - تصویر اول را به عنوان تصویر نرمال در نظر می گیریم.
  - تصویر دوم را به عنوان تصویری که هنگام کلیک برای دکمه باید نمایش داده شود در نظر می‌گیریم.
  - یک فایل xml در فولدر drawable ایجاد می کنیم و selector برای دکمه ها تعریف می کنیم.
  - بعد از تعریف selector نام فایل selector را در src دکمه در UI اکتیویتی قرار می دهیم.

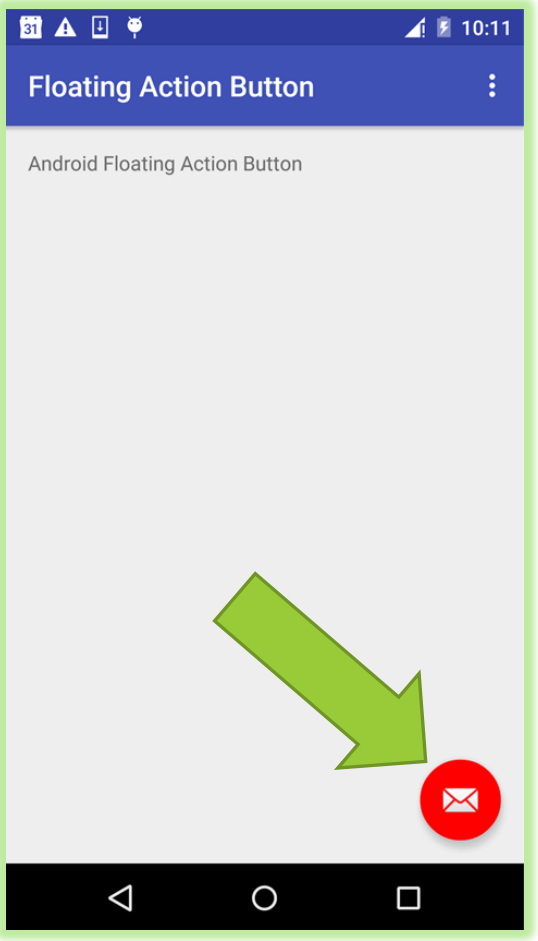
```
android:src="@drawable/selector_file"
```

کد نویسی جاوا – ویجت Button
ساخت دکمه های داینامیک

- محتویات فایل selector با پسوند xml در فولدر drawable

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item
        android:state_focused="true"
        android:state_pressed="false"
        android:drawable="@drawable/pressed_image" />
   <item
        android:state_focused="true"
        android:state_pressed="true"
        android:drawable="@drawable/pressed_image" />
   <item
        android:state_focused="false"
        android:state_pressed="true"
        android:drawable="@drawable/pressed_image" />
   <item
        android:drawable="@drawable/normal_image" />
</selector>
```

کد نویسی جاوا – ویجت Button
ساخت دکمه شناور



- دکمه ای است که به صورت شناور بر روی اکتیویتی قرار می گیرد.
- از کتابخانه com.android.support:design استفاده می کند.
- برای ایجاد دکمه شناور باید کتابخانه فوق به گریدل اضافه شود.
- مثال اضافه کردن کتابخانه به گریدل

```
dependencies {
   compile 'com.android.support:design:23.3.0'
}
```

کد نویسی جاوا – ویجت Button
ساخت دکمه شناور

- سپس در فایل UI اکتیویتی کد مربوط به دکمه شناور را ایجاد می کنیم.

```
<android.support.design.widget.FloatingActionButton
      android:id="@+id/floating_button_id"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="bottom|end"
      android:src="@drawable/floating_button_image"
      android:layout_weight="1"
     android:scaleType="center"
      app:backgroundTint="@android:color/transparent"
       />
```

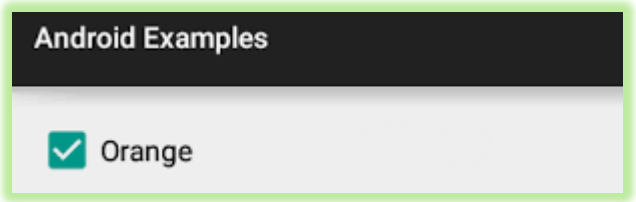
کد نویسی جاوا – ویجت Button
ساخت دکمه شناور

- در نهایت مشابه دکمه معمولی در فایل جاوای اکتیویتی ارتباط ویوی آن با اکتیویتی را برقرار می کنیم. (کد در کلاس onCreate() نوشته می شود)

```
FloatingActionButton fl_button = (FloatingActionButton) findViewById (R.id.floating_button_id);


fl_button.setOnClickListener(new View.OnClickListener(){
        public void onClick(View arg0){

            ...
           // my action code
            ...

       }
    });
```

کار در کلاس

نمونه ای از تکست ویو و
دکمه ایجاد کرده و موارد
توصیف شده را بر روی آن ها
اعمال کنید.



کد نویسی جاوا – ویجت CheckBox

- فرض کنیم در فایل xml یک چک باکس با نام checkBox1 تعریف شده است. کد جاوا به شرح زیر است: کد جاوا در کلاس onCreate() اکتیویتی نوشته می‌شود.

- مرحله اول: تعریف یک رشته برای چک باکس

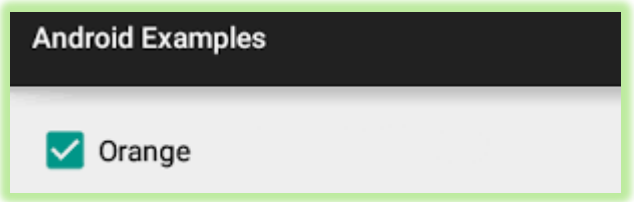
```
String checkbox_title = "Orange";
```

- مرحله دوم: تعریف چک باکس و ارتباط به ویوی آن

```
CheckBox my_checkBox = (CheckBox) findViewById(R.id.checkbox_view);
```

- مرحله سوم: ست کردن رشته به چک باکس

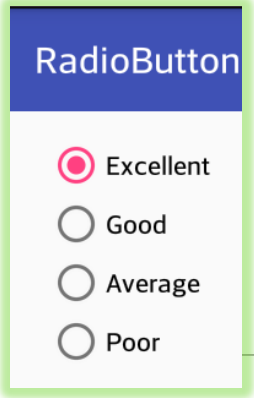
```
my_checkBox.setText(checkbox_title);
```



کد نویسی جاوا – ویجت CheckBox

- برای چک کردن اینکه آیا چک باکس تیک خورده یا خیر یا در صورتی که تیک بخورد عملیاتی صورت گیرد، مشابه دکمه، باید یک لیسنر برای آن ایجاد شود.

```
my_checkBox.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           if (((CheckBox) v).isChecked()) {
              // my action code if checkbox is checked
           }
           else {
             // my action code if checkbox is unchecked
            }
        }
     });
```



کد نویسی جاوا – ویجت RadioButton

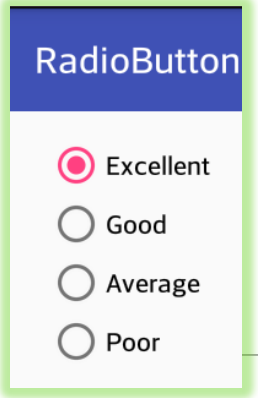
- فرض کنیم در فایل xml یک رادیو گروپ با نام radioGroup1 تعریف شده است. کد جاوا به شرح زیر است: کد جاوا در کلاس onCreate() اکتیویتی نوشته می‌شود.

- مرحله اول: تعریف رادیو گروپ و ارتباط ویوی آن

```
RadioGroup radio_group = (RadioGroup) findViewById(R.id.radioGroup1);
```

- مرحله دوم: تعریف و ست کردن رشته برای بچه های آن (دکمه های رادیویی که قبلا در UI درون رادیو گروپ ایجاد شده اند).

```
((RadioButton) radio_group.getChildAt(0)).setText("Excellent");
((RadioButton) radio_group.getChildAt(1)).setText("Good");
((RadioButton) radio_group.getChildAt(2)).setText("Average");
((RadioButton) radio_group.getChildAt(3)).setText("Poor");
```



کد نویسی جاوا – ویجت RadioButton

- مرحله سوم: ایجاد یک لیسنر برای رادیو گروپ

```
radio_group_button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                // تعریف یک متغیر برای دریافت اینکه کدام دکمه انتخاب می شود
               int selectedId = radio_group.getCheckedRadioButtonId();

               // یافتن دکمه انتخاب شده با آی دی دریافت شده
              radioButton = (RadioButton) findViewById(selectedId);

               // دریافت شماره بچه انتخاب شده
              int child = radio_group.indexOfChild(findViewById(radio_group.getCheckedRadioButtonId()));
        }
    });
```

کد نویسی جاوا – EditText

- فرض کنیم در فایل xml یک تکست فیلد با نام editText1 تعریف شده است. کد جاوا به شرح زیر است: کد جاوا در کلاس onCreate() اکتیویتی نوشته می‌شود.
- تعریف ادیت تکست و ارتباط ویوی آن

```
EditText my_editText = (EditText) findViewById (R.id.mozu_editText1);
```

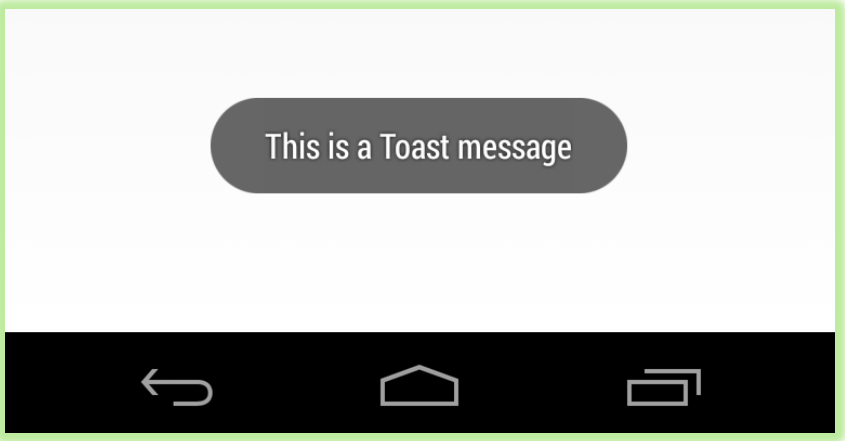
- دریافت اطلاعاتی که کاربر در آن تایپ کرده است

```
String txt;
txt = my_editText .getEditableText().toString();
```

کد نویسی جاوا – نمایش Toast

- یک Toast ویویی است که یک پیغام کوچک را به کاربر نمایش می دهد. برای ایجاد Toast از کلاس Toast استفاده می شود.
- روش سریع ایجاد Toast

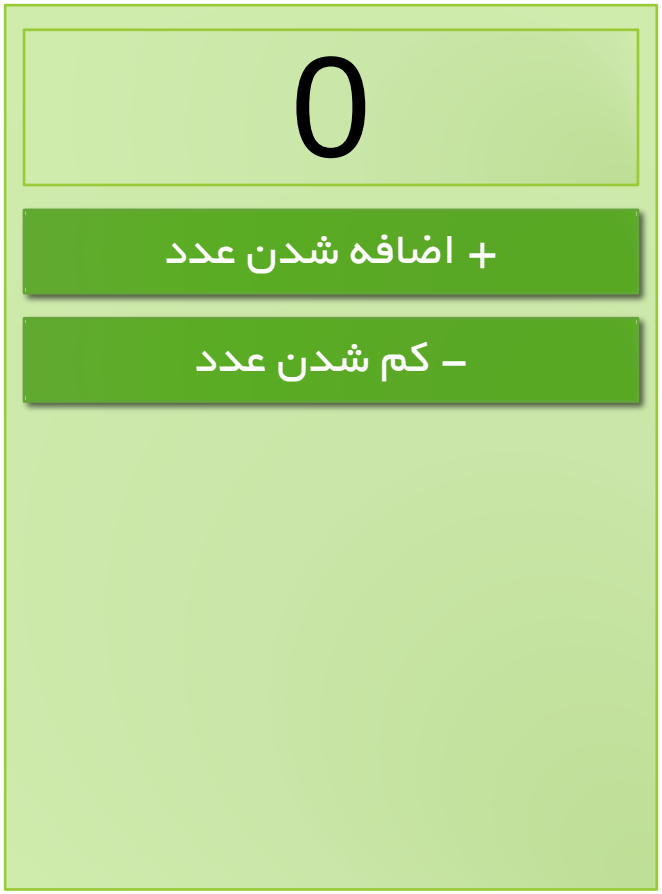
```
Toast.makeText(this, "This is a Toast message", Toast.LENGTH_LONG).show();
```



کار در کلاس

نمونه ای از چک باکس و دکمه رادیویی و ادیت تکست ایجاد کرده و موارد توصیف شده را بر روی آن ها اعمال کنید.

کار در کلاس – پروژه



فرم روبرو را با یک تکست ویو و دو دکمه ایجاد کنید که با کلیک بر روی دکمه عملیات کم کردن و زیاد کردن عدد تکست ویو انجام شده و نمایش داده شود.

کار در کلاس – کد های پروژه

```
int Count = 0;
String Counter = "";

    final TextView myText = (TextView) findViewById(R.id.textView);
    Button myButton1 = (Button) findViewById(R.id.button);
    Button myButton2 = (Button) findViewById(R.id.button2);

    myButton1.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0){
            Count++;
            Counter = String.valueOf(Count);
            myText.setText(Counter);
        }
    });

    myButton2.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0){
            Count--;
            Counter = String.valueOf(Count);
            myText.setText(Counter);
        }
    });
```

شروع کار با اندروید استودیو

- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها
- تأثیرگذاری روی ویجت ها با کد نویسی جاوا
- آشنایی با Intent و Splash Screen در قالب انجام پروژه

آشنایی با اینتنت (Intent)

- در لغت به معنی قصد و نیت برای انجام کار است.
- در اندروید به واسطه اینتنت اعلام می کنیم که قصد انجام چه کاری را داریم.
- اینتنت واسطی است بین کامپوننت های مختلف.
- ساده ترین و کاربردی ترین کار اینتنت انتقال از یک اکتیویتی به اکتیویتی دیگر است.

آشنایی با اینتنت (Intent)

- کد جاوا دستور انتقال از یک اکتیویتی به اکتیویتی دیگر به شرح زیر است. کد جاوا در کلاس onCreate() اکتیویتی نوشته می‌شود.

```
Intent intent_name = new Intent (current_activity.this, other_activity.class);
startActivity (intent_name );
```

کار در کلاس – پروژه اول

دو اکتیویتی ایجاد کنید و آن ها را با استفاده از اینتنت به یکدیگر متصل کنید.

آشنایی با Splash Screen

- انتقال خودکار از یک اکتیویتی به اکتیویتی دیگر بعد از مدت زمان مشخص که معمولا در ابتدای یک برنامه استفاده می شود را Splash Screen می گوییم.
- این کار توسط کلاس کنترل کننده (Handler) و زیر کلاس تأخیر (postDelayed) انجام می شود. کد جاوای آن به شرح زیر است.

```
new Handler().postDelayed (new Runnable() {
    @Override
    public void run() {
        // my intent code
    }
}, 3000);
```

آشنایی با Splash Screen

- در صورتی که بخواهیم ابتدای برنامه یک اسپلش اسکرین نمایش دهیم و بعد از پنج ثانیه به اکتیویتی اصلی برنامه برسیم، باید دو اکتیویتی ایجاد کرده و کد زیر را در کلاس onCreate() اکتیویتی اول ایجاد کنیم.

```
new Handler().postDelayed (new Runnable() {
    @Override
    public void run() {
        Intent intent_name = new Intent (current_activity.this, other_activity.class);
        startActivity (intent_name );
    }
}, 5000);
```

کار در کلاس – پروژه دوم

دو اکتیویتی ایجاد کنید و آن ها را با استفاده از اسپلش اسکرین به یکدیگر متصل کنید. مدت زمان نمایش اکتیویتی اول را ۲ ثانیه در نظر بگیرید.

شروع کار با اندروید استودیو

- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها
- تأثیرگذاری روی ویجت ها با کد نویسی جاوا
- آشنایی با Intent و Splash Screen در قالب انجام پروژه
- وب ویو (WebView)

وب ویو (webView)

- همانطور که قبلا بیان شد، وب ویو یک مرورگر کوچک که می توان از آن در برنامه ها استفاده کرد.
- جهت نمایش صفحات وب، یا صفحه ای با فرمت HTML کاربرد دارد.
- فرض کنید در فایل UI برنامه یک وب ویو با نام webView1 تعریف شده است. کد جاوا جهت استفاده از آن به شرح زیر است.
- مرحله اول: تعریف متغیر از نوع وب ویو و ارتباط با xml

```
WebView webInfo;
webInfo = (WebView) findViewById (R.id.webView1);
```

وب ویو (webView)

- مرحله دوم: تعریف یک متغیر رشته با محتوای html برای نمایش در وب ویو

```
String my_html = "<html><body> this is my page </body></html>";
```

- مرحله سوم: لود رشته در وب ویو

```
webInfo.loadUrl("page.php");
//    http://www.google.com/
```

```
webInfo.loadDataWithBaseURL(null, my_html, "text/html", "UTF-8", null);
```

- آدرس URL برای استفاده. اگر null باشد صفحه خالی نمایش داده می شود.
- صفحه html به صورت یک رشته.
- نوع نمایش که به صورت پیش فرض text/html است.
- نوع کد گذاری برای صفحه
- نمایش URL به عنوان صفحه history

وب ویو (webView)

- برای استفاده از فونت یا تصویر یا CSS دلخواه، فایل آن ها باید در پوشه assets قرار گیرد. و سپس به صورت زیر آدرس دهی شود:

```
String my_html = "<html><body>
                    this is my page
                  <img src=\"file:///android_asset/img/myImage.png\">
                  </body></html>";
```

برای استفاده از وب ویو و نمایش آنلاین صفحات نیازاست که مجوز دسترسی به اینترنت وجود داشته باشد. این مجوز در فایل مانیفست اضافه می شود:

```
<uses-permission android:name="android.permission.INTERNET" />
```

برای مطالعه و دریافت اطلاعات بیشتر در مورد وب ویو به سایت توسعه دهندگان گوگل مراجعه کنید:

https://developer.android.com/reference/android/webkit/WebView.html

برای مطالعه و دریافت اطلاعات بیشتر تنظیمات وب ویو به لینک زیر مراجعه کنید:
https://developer.android.com/reference/android/webkit/WebSettings.html

برخی تنظیمات وب ویو (webView)

- برای فعال یا غیر فعال سازی برخی تنظیمات از کلاس getSettings() وب ویو استفاده می شود.
- برخی تنظیمات:
  - فعال یا غیر فعال سازی کد جاوا اسکریپت در صفحه مورد نظر:

```
webInfo.getSettings().setJavaScriptEnabled(true);   // or false
```

  - مرور صفحه در حالت کلی یا جاسازی محتویات به اندازه عرض وب ویو:

```
webInfo.getSettings().setLoadWithOverviewMode(true);  // or false
```

  - وب ویو از مکانیزم زوم صفحه استفاده کند یا خیر:

```
webInfo.getSettings().setBuiltInZoomControls(true);  // or false
```

برای مطالعه و دریافت اطلاعات بیشتر تنظیمات وب ویو به لینک زیر مراجعه کنید:
https://developer.android.com/reference/android/webkit/WebSettings.html

برخی تنظیمات وب ویو (webView)

- برخی تنظیمات:
  - تغییر اندازه فونت وب ویو با استفاده از تنظیمات وب ویو: به صورت پیش فرض ۱۶ تعیین شده است. می تواند عددی بین ۱ تا ۷۲ باشد.

```
webInfo.getSettings().setDefaultFontSize(50);
```

  - باز شدن لینک های صفحه مورد نظر در خود وب ویو به صورت مستقیم:

```
webInfo.setWebViewClient(new WebViewClient());
```

برخی تنظیمات وب ویو (webView)

- برخی تنظیمات:
  - دسترسی به صفحات قبلی لینک شده به وب ویو با استفاده از دکمه back دستگاه. دکمه back به صورت پیش فرض باعث خروج از اکتیویتی می شود.

```
@Override
   public boolean onKeyDown (int keyCode, KeyEvent event) {
      if(event.getAction() == KeyEvent.ACTION_DOWN){
         switch(keyCode)
         {
         case KeyEvent.KEYCODE_BACK:
            if(webView.canGoBack()){
                  webView.goBack();
            } else{
               finish();
            }
            return true;
         }
      }
      return super.onKeyDown(keyCode, event);
   }
```

کار در کلاس

یک وب ویو ایجاد کنید و اطلاعات دلخواه در آن نمایش دهید.

شروع کار با اندروید استودیو

- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها
- تأثیرگذاری روی ویجت ها با کد نویسی جاوا
- آشنایی با Intent و Splash Screen در قالب انجام پروژه
- وب ویو (WebView)
- نمایش نوتیفیکیشن (Notification)

نمایش نوتیفیکیشن (Notification)

- برای ایجاد نوتیفیکیشن از کلاس NotificationCompat.Builder استفاده می شود.
- شیوه استفاده از کلاس NotificationCompat.Builder به صورت زیر است:

```
NotificationCompat.Builder mBuilder = (NotificationCompat.Builder) new
NotificationCompat.Builder(this) … ;
```

تنظیمات مربوط به نوتیفیکیشن

نمایش نوتیفیکیشن (Notification)

- برخی تنظیمات نوتیفیکیشن
- نمایش آیکن مربوط به نوتیفیکیشن که در استاتوس بار نمایش داده می شود.

```
.setSmallIcon(R.drawable.myIcon)
```

- تعیین نوع لرزش دستگاه هنگام نمایش نوتیفیکیشن.

```
.setVibrate(new long[] { 300, 300, 300, 300, 300 })  // mili sec
```

- تعیین نور مربوط به LED در برخی دستگاه ها.

```
.setLights (0xff367bd2, 3000, 3000)  // mili sec
```

نمایش نوتیفیکیشن (Notification)

- برخی تنظیمات نوتیفیکیشن
  - نمایش آیکنی که هنگام مشاهده نوتیفیکیشن کنار موضوع نمایش داده می‌شود.

```
.setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.logo))
```

  - نمایش عنوان نوتیفیکیشن

```
.setContentTitle("my title")
```

  - نمایش پیام مورد نظر در نوتیفیکیشن

```
.setContentText("my description")
```

نمایش نوتیفیکیشن (Notification)

- برخی تنظیمات نوتیفیکیشن
- نمایش متن طولانی و نوتیفیکیشنی چند سطری

```
.setStyle(new NotificationCompat.BigTextStyle().bigText("\n"))
```

- حذف نوتیفیکیشن از استاتوس بار با لمس کاربر

```
.setAutoCancel(true)
```

- باز شدن اکتیویتی دلخواه بعد از لمس کاربر بر روی نوتیفیکیشن

```
mBuilder.setContentIntent(PendingIntent.getActivity(this, 0, new Intent(this,
myActivity.class), PendingIntent.FLAG_UPDATE_CURRENT));

mBuilder.setOngoing(true);   //غیر قابل حذف
```

نمایش نوتیفیکیشن (Notification)

- حال برای نمایش نوتیفیکیشن، بعد از اینکه تنظیمات دلخواه صورت گرفت، باید از کلاس NotificationManager استفاده کنیم. کد مربوط به NotificationManager به شرح زیر است:

```
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(001, mBuilder.build());
```

یک شماره دلخواه برای عدم تداخل نوتیفیکیشن های مختلف باهمدیگر

نوتیفیکیشنی که ایجاد و تنظیم کردیم

کار در کلاس

چند دکمه ایجاد کنید که با کلیک بر روی هر کدام نوتیفیکیشن مربوط به همان دکمه نمایش داده شود.

شروع کار با اندروید استودیو

- مفاهیم مقدماتی اندروید استودیو
- ایجاد پروژه در اندروید استودیو و مفاهیم آن
- آشنایی با Palette ها
- تأثیرگذاری روی ویجت ها با کد نویسی جاوا
- آشنایی با Intent و Splash Screen در قالب انجام پروژه
- وب ویو (WebView)
- نمایش نوتیفیکیشن (Notification)
- ذخیره و بازیابی اطلاعات در فایل

ذخیره اطلاعات متنی در فایل

- ذخیره اطلاعات متنی در فایلی در محل ذخیره سازی داخلی دستگاه (InternalStorage) با استفاده از کلاس FileOutputStream صورت می گیرد.
- ذخیره سازی در حافظه داخلی دستگاه نیاز به مجوز ندارد.
- فرض کنید می خواهیم اطلاعاتی که کاربر در یک تکست فیلدی (EditText) به نام editText1 وارد می کند را ذخیره سازی کنیم. به روش زیر عمل می‌کنیم.
- مرحله اول: تعریف ادیت تکست

```
EditText edit_text = (EditText)findViewById(R.id.editText1);
```

ذخیره اطلاعات متنی در فایل

- مرحله دوم: ذخیره سازی با استفاده از کد جاوا و کلاس FileOutputStream

```
try {
        FileOutputStream fileout = openFileOutput("mytextfile.txt", MODE_PRIVATE);
        OutputStreamWriter outputWriter = new OutputStreamWriter (fileout);
        outputWriter.write(textmsg.edit_text().toString());
        outputWriter.close();

        //display file saved message
        Toast.makeText(getBaseContext(), "File saved", Toast.LENGTH_SHORT).show();
}
catch (Exception e) {
        e.printStackTrace();
}
```

بازیابی اطلاعات متنی از فایل

- برای بازیابی نیاز به یک اندازه بلوک برای خواندن اطلاعات داریم (تعداد کاراکترهایی که خوانده می شود).

```
static final int READ_BLOCK_SIZE = 100;
```

- این بلوک از نوع int و قبل از کلاس onCreate() تعریف می شود.

- برای بازیابی اطلاعات بازهم از کلاس FileInputStream استفاده می شود با این تفاوت که تمامی اطلاعات بافر می شود، سپس با استفاده از یک حلقه مثل حلقه while خوانده می شود و در یک متغیر رشته ریخته می شود.

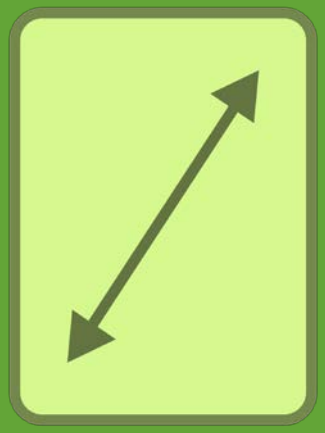
بازیابی اطلاعات متنی از فایل

- قطعه کد زیر بازیابی اطلاعات را انجام می دهد:

```
try {
        FileInputStream fileIn = openFileInput("mytextfile.txt");
        InputStreamReader InputRead= new InputStreamReader(fileIn);
        char[] inputBuffer = new char[READ_BLOCK_SIZE];
        String s = "";
        int charRead;
        while ((charRead = InputRead.read(inputBuffer)) > 0) {
                String readstring = String.copyValueOf(inputBuffer,0,charRead); // char to string conversion
                s += readstring;
        }
 InputRead.close();
 edit_text.setText(s);
}
catch (Exception e) {
        e.printStackTrace();
}
```

کار در کلاس

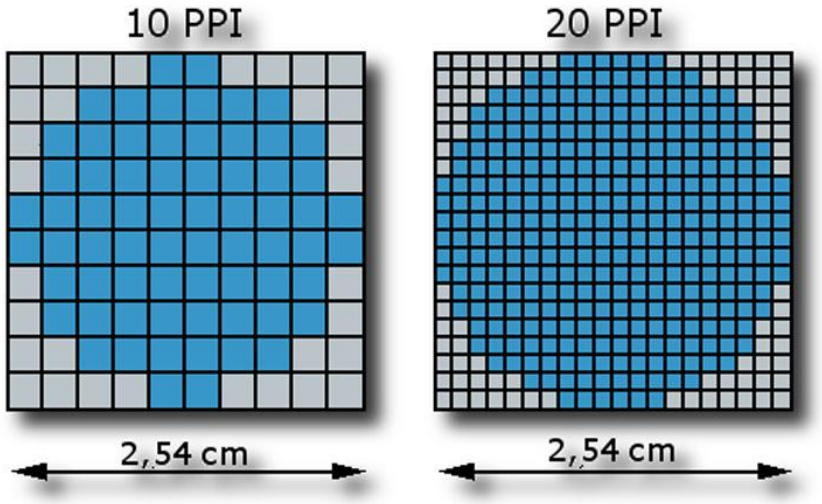
یک نمونه کار ذخیره و بازیابی اطلاعات را با استفاده از ادیت تکست انجام دهید.



رزولوشن صفحه در برنامه های اندروید

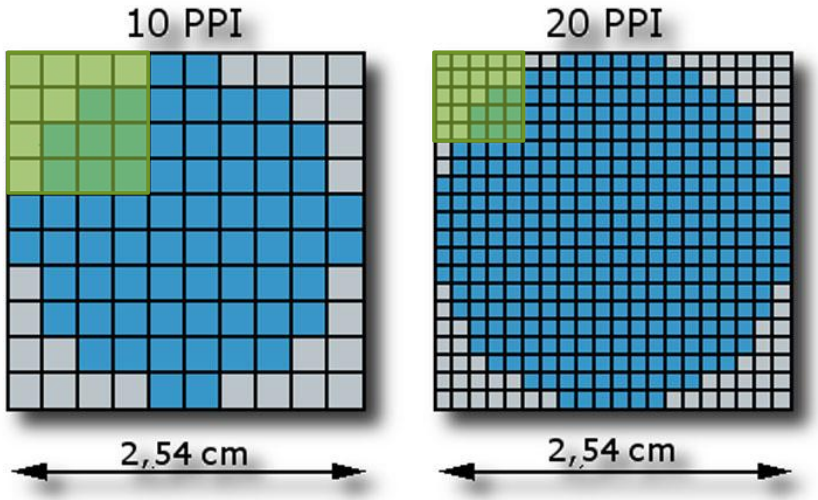
رزولوشن صفحه در برنامه ها

- دستگاه های مختلف اندروید دارای صفحه نمایش مختلفی هستند.
- صفحه نمایش می تواند تراکم پیکسلی کم یا زیاد داشته باشد.
- هرچه تراکم پیکسلی بالاتر باشد، کیفیت نمایش تصاویر بالاتر می رود.



رزولوشن صفحه در برنامه ها

- فرض کنید تصویری به ابعاد ۵ در ۵ پیکسل وجود دارد. تصویر در صفحه نمایش با تراکم پیکسلی بالا، کوچکتر نمایش داده می شود.



راه حل این مشکل استفاده از dip به جای پیکسل است. در این صورت سیستم خودش به تناسب صفحه، تصویر را بزرگ می کند. عیب این روش از بین رفتن کیفیت تصویر است.

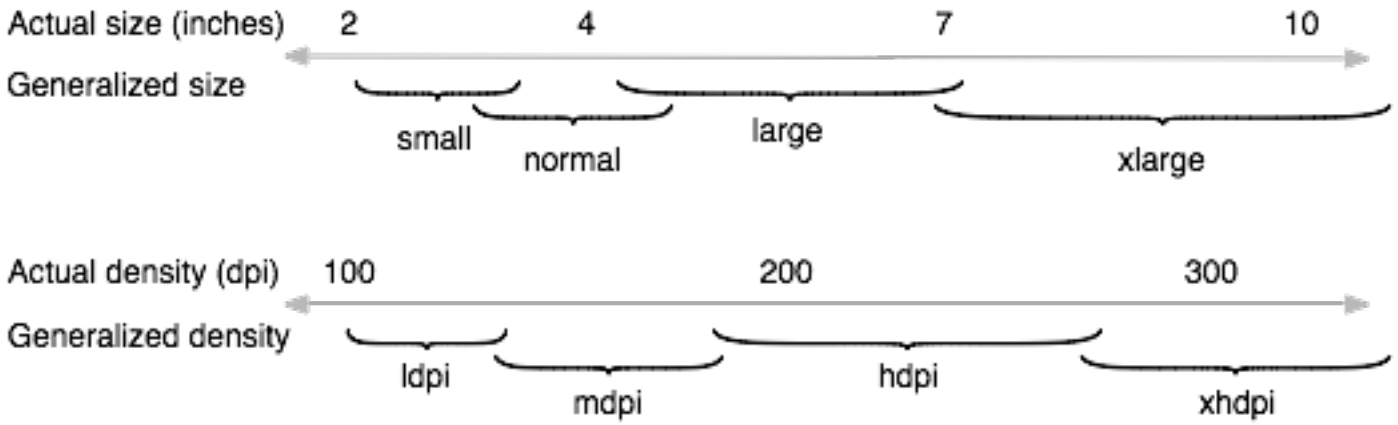
رزولوشن صفحه در برنامه ها

- دسته بندی تراکم های مختلف صفحه نمایش در اندروید

تراکم	توضیح
ldpi	مخفف low-dpi، به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها بسیار کم و در حدود ۱۲۰ dpi است.
mdpi	مخفف medium-dpi، به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها متوسط و در حدود ۱۶۰ dpi است.
hdpi	مخفف high-dpi، به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها بالا و در حدود ۲۴۰ dpi است.
xhdpi	مخفف extra-high-dpi، به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها بالا و در حدود ۳۲۰ dpi است.
xxhdpi	به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها بالا و در حدود ۴۸۰ dpi است.
xxxhdpi	به صفحه نمایش هایی اطلاق می شود که تراکم پیکسلی آنها بالا و در حدود ۶۴۰ dpi است.

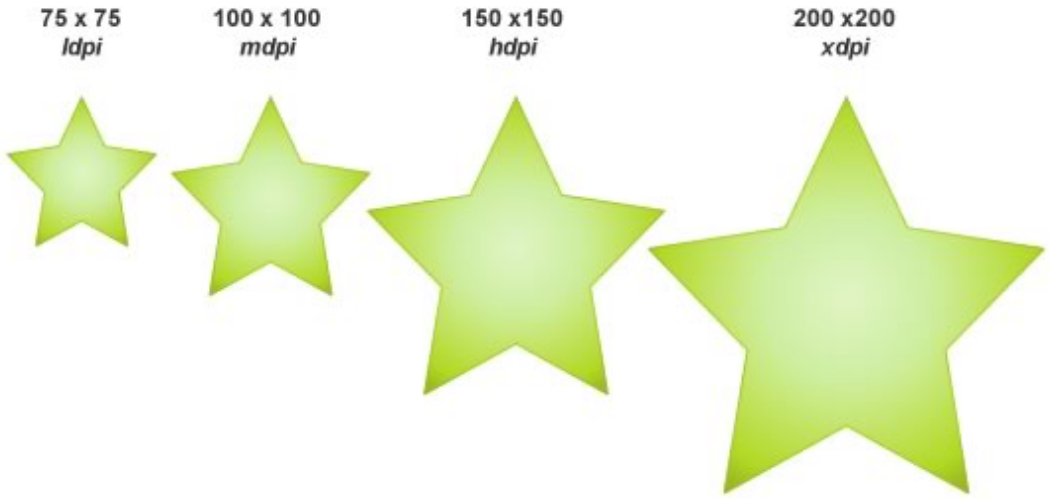
رزولوشن صفحه در برنامه ها

- درک تراکم های مختلف



رزولوشن صفحه در برنامه ها

- درک تراکم های مختلف



رزولوشن صفحه در برنامه ها

- درک تراکم های مختلف

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*	Extra high density (320), *xhdpi*
Small screen	**QVGA (240x320)**		480x640	
Normal screen	**WQVGA400 (240x 400)** **WQVGA (240x432)**	**HVGA (320x480)**	**WVGA (480x800)** **WVGA854 (480x854)** 600x1024	640x960
Large screen	**WVGA800 (480x800)** **WVGA854 (400x854)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024		
Extra large screen	1024x600	**WXGA (1280x800)** 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

صداگذاری در برنامه های اندروید

صداگذاری در اندروید

- در اندروید برای صدا گذاری از کلاس مدیا پلیر استفاده می شود.
- برای استفاده از صدا، ابتدا باید فایل صدا موجود باشد.
- فایل صدا باید در فولدری به نام raw در پوشه res قرار بگیرد.
- فرض کنید فایل صدایی به نام check.mp3 در مسیر res\raw موجود باشد. برای استفاده از صدا صورت زیر عمل می کنیم:
- مرحله اول: تعریف متغیر از جنس مدیاپلیر

```
MediaPlayer checkSound;
```

صداگذاری در اندروید

- مرحله دوم: ایجاد ارتباط متغیر با فایل صدای موجود در پوشه raw

```
checkSound = MediaPlayer.create (context, R.raw.check);
```

- مرحله سوم: استفاده از صدا با استارت کردن آن

```
checkSound.start();
```

برای توقف صدا از کد زیر استفاده می شود:

```
checkSound.stop();
```

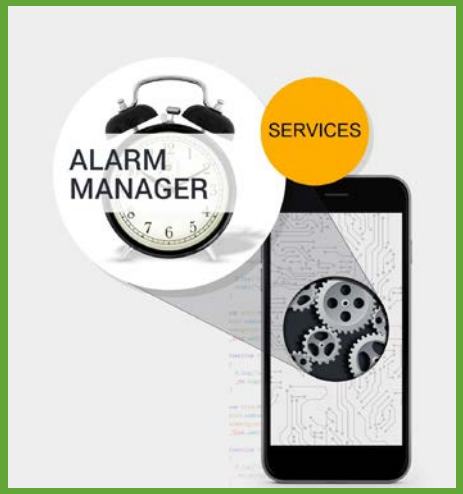
صداگذاری در اندروید

- شیوه صحیح استفاده از صدا به این صورت است که در صورت استارت صدا، ابتدا چک شود اینکه آیا صدا در حال اجرا هست یا خیر.
- در صورتی که صدا در حال اجرا باشد، ابتدا صدا متوقف شود، سپس حافظه آن آزاد شده و دوباره به آن تخصیص داده می شود. در نهایت دوباره استارت می شود.
- این شیوه باعث می شود تا روی هم اجرا شدن صدا به وجود نیاید.

صداگذاری در اندروید

- مثال از اجرای صحیح صدا هنگام کلیک بر روی یک دکمه

```
myButton.setOnClickListener(new View.OnClickListener(){
        public void onClick(View arg0){
              try {
                    if (checkSound.isPlaying()) {
                          checkSound.stop();
                          checkSound.release();
                          checkSound = MediaPlayer.create(context, R.raw.check);
                     }
               checkSound.start();
               }
              catch(Exception e) {
               e.printStackTrace();
              }
           // action of button
        }
      });
```



۵ آشنایی با سرویس و مدیریت آلارم

کامپوننت سرویس

- همانطور که قبلا در بخش توصیف کامپوننت ها بیان شد، یک سرویس کامپوننتی است که برای انجام عملیات طولانی مدت در پس زمینه بدون نیاز به تعامل با کاربر اجرا می شود و حتی اگر اپلیکیشن destroy شود، باز هم ادامه پیدا می کند.
- برای ایجاد سرویس، یک کلاس جاوا ایجاد می کنیم که از کلاس پایه Service یا یکی از زیرکلاس های موجود آن ارث بری می کند.
- هر سرویس باید با تگ <service> در مانیفست برنامه تعریف شود.

```
<service android:enabled="true" android:name=".MyService"/>
```

کامپونت سرویس

- شروع به کار سرویس با فراخوانی کلاس آن توسط یک اینتنت و دستور startService() انجام می‌شود.

```
Intent intent=new Intent (this,MyService.class);
startService(intent);
```

- مثالی از ایجاد یک سرویس ساده:

```
public class MyService extends Service {
   @Override
   public int onStartCommand(Intent intent, int flags, int startId) {
      // your code
      return Service.START_FLAG_REDELIVERY;
   }
   @Override
   public IBinder onBind(Intent intent) {
      return null;
   }
 }
```

کامپوننت سرویس

- توقف سرویس با استفاده از دستور stopService() صورت می گیرد.
- از متد onDestroy نیز که در کلاس سرویس ایجاد می شود نیز می توان برای توقف سرویس استفاده کرد. این متد به صورت زیر است:

```
@Override
public void onDestroy(){
	super.onDestroy();
}
```

- فراخوانی متد بالا به صورت زیر است:

```
onDestroy();
```

اعلان شروع سرویس هنگام بوت شدن دستگاه

- سرویس بعضی برنامه های به صورت است که، وقتی که دستگاه خاموش می شود، بعد از روشن شدن دستگاه نیاز است تا سرویس های برنامه شروع به کار کنند. برای این کار ایجاد یک کلاس از کامپوننت BroadcastReceiver داریم تا به همه اعلام کنیم که بعد از روشن شدن دستگاه، سرویسی باید شروع به کار کند.

- برای این کار ابتدا کلاس مربوطه را ایجاد می کنیم:

```
public class BootReceiver extends WakefulBroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
        context.startService(new Intent(context, myService.class));
        }
}
```

اعلان شروع سرویس هنگام بوت شدن دستگاه

- سپس باید اطلاعات کلاس را به فایل مانیفست معرفی کنیم. یعنی اعلام می کنیم که بعد از بوت شدن دستگاه کلاس BootReceiver که قبلا ایجاد کردیم را اجرا کن.

```
<receiver android:name=".BootReceiver">
        <intent-filter>
          <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
</receiver>
```

- همچنین اجرا بعد از بوت شدن نیاز به مجوز دارد که در مانیفست اضافه می‌شود:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

مدیریت آلارم (AlarmManager)

- گاهی اوقات می خواهیم که بعضی از عملیات ها در زمان دیرتری صورت گیرد، کلاس AlarmManager دسترسی را به سیستم های Alarm فراهم می‌کند، این کار باعث می شود بتوان عملیات مورد نظر را در زمانی که می‌خواهیم انجام دهیم.
- AlarmManager بخشی از CPU را اشغال می کند و تا زمانی که متد onRecieve() فعال است، آن بخش از CPU را اشغال است.
- همچنین از آلارم برای اجرای سرویس ها در زمان مشخص نیز می تواند استفاده شود.

مدیریت آلارم (AlarmManager)

- کد زیر مثالی است که در آن آلارم یک سرویس را هر ۳۰ ثانیه، اجرا می‌کند.

```
Intent intent = new Intent (this, myService.class);
PendingIntent pintent = PendingIntent.getService(this, 0, intent, 0);

AlarmManager alarm = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
alarm.setRepeating (AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(), 30 * 1000, pintent);
```

Android

THE END BEGINS…