

## Л5. Методи

**Завдання 1.** Який з алгоритмів вірно застосовує метод **Calculate the Price**

### Option A

```
/**
 * Calculates the price of the order based on the current
 * quantity.
 *
 * @return the price
 */
private int calculate price(int quantity {
    int price = quantity * 5;
    return price;
}
```

### Option B

```
/**
 * Calculates the price of the order based on the current
 * quantity.
 *
 * @return the price
 */
private calculatePrice(int quantity)
    int price = quantity * 5;
    return price;
```

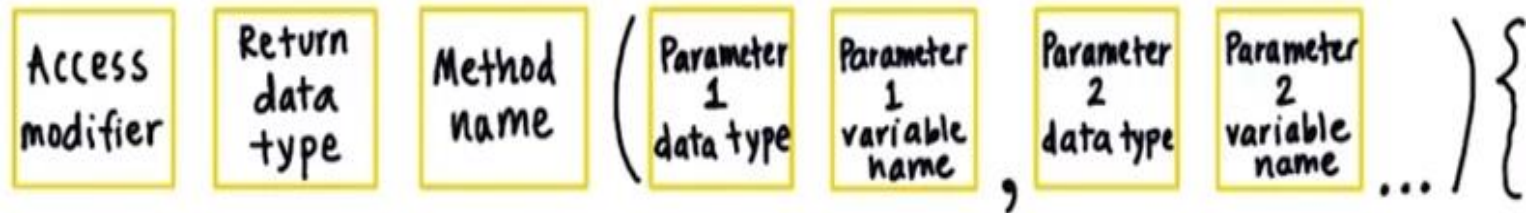
### Option C

```
/**
 * Calculates the price of the order based on the current
 * quantity.
 *
 * @return the price
 */
private int calculatePrice(int quantity) {
    int price = quantity * 5;
    return price;
}
```

## Л5. Методи

У Java визначення метода ще називається **сигнатура метода**. Сигнатура метода – це тільки його назва та тип параметрів. Більш детально з особливостями застосування методів можна ознайомитись в [документації по Java](#). Приклад оголошення методу

```
private int calculatePrice(int quantity) {}
```



//Here is an example of a typical method declaration:

```
public double calculateAnswer(double wingSpan, int numberOfEngines, double length, double grossTons)
{
//do the calculation here
}
```

## Л5. Методи

*Завдання 2. Дано частина коду. Потрібно визначити назву методу, вхідні параметри та тип даних, що повертається. Заповнити таблицю*

	Method Name	Input Parametrs	Return data type
Sample	createCustomGreeting	String firstName, String lastName	String
Method 1			
Method 2			
Method 3			

```
/**
 * Creates a custom greeting message based on the name.
 *
 * @return text greeting
 */
private String createCustomGreeting(String firstName, String lastName) {
    return "Welcome, " + firstName + " " + lastName + "!";
}
```

## Л5. Методи

	Method Name	Input Parametrs	Return data type
Sample	createCustomGreeting	String firstName, String lastName	String
Method 1			
Method 2			
Method 3			

### Method 1

```
private String createCalendarEventReminder(String eventName, String location, int minutesAway) {  
    String reminder = "You have an upcoming event in " + minutesAway + " minutes.";  
    reminder = reminder + " It is " + eventName + " held at " + location + ".";  
    return reminder;  
}
```

### Method 2

```
private int deductPoints(int pointsUsed) {  
    // Everyone starts with 100 points  
    int numberOfPoints = 100 - pointsUsed;  
    return numberOfPoints;  
}
```

### Method 3

```
private String findTotalTripLength(int distanceOfFirstTrip, int distanceOfSecondTrip, int  
distanceOfThirdTrip) {  
    int totalLength = 2;  
    totalLength = totalLength + distanceOfFirstTrip + distanceOfSecondTrip + distanceOfThirdTrip;  
    String message = "The total trip will be: " + totalLength + " miles.";  
    return message;  
}
```

## Л5. Методи

**Параметри** вказуються при створенні методу, їх приймає метод та обробляти всередині. При створенні методу оголошуємо та перераховуємо які параметри він буде приймати при виклику.

**Аргументи** передаються методу при його виклику, вхідні дані, які він прийме на обробку. При виклику методу передаємо йому фактичні значення – аргументи

Виконання методу завершується:

- коли виконали усі рядки в методі;
- коли досягнули оперетора повернення **return**, який є першим;
- метод завершиться якщо викликається виняток або помилка.

У кодї може зустрітись `return`. Це вказує, що метод завершується і будь-який наступний рядок у методі не буде виконуватись. Може також зустрітись код – **`return returnValue`**. Це є повернення вхідного параметру з методу.

```
// a method for computing the area
public int getArea() {
    return width*height;
}
```

Тип даних, що повертає даний метод  
`int width*height`.

## Л5. Методи

**Завдання 3.** Є код до 4 різних методів. В кожному з них є помилка. Потрібно знайти помилку в кожному з методів. Написати номер рядка та опис помилки.

Метод	Номер рядка	Опис помилки
Method 1		
Method 2		
Method 3		
Method 4		

### Method 1

```
/**
 * Get the email account name.
 *
 * @return the name of the account.
 */
private String getAccountName() {
    return "android@gmail.com";
    return "droid@gmail.com";
}
```

### Method 2

```
/**
 * Add $4 of tip onto the current bill.
 *
 * @return the total price of the bill (including tip).
 */
private addTip(int bill) {
    return bill + 4;
}
```

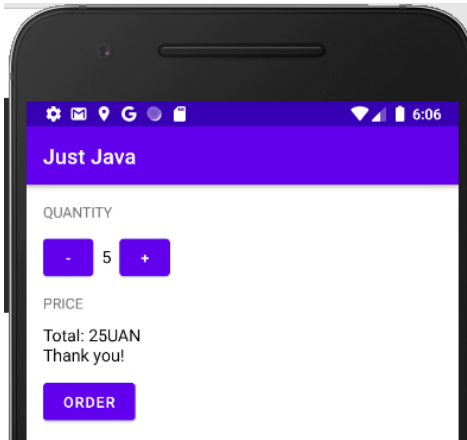
### Method 3

```
/**
 * Sets up the app for the current city.
 */
private nothing setup() {
    cityName = "London";
}
```

### Method 4

```
/**
 * Get the number of students in a class.
 *
 * @return the number of students.
 */
private int getStudentClassSize() {
    return "20";
}
```

## Л5. Just Java. Change Price to Order Summary



1. У файлі `activity_main.xml`:

- змінюємо текст `PRICE` на `ORDER SUMMARY`;
- `TextView` зі значення ціни задаємо ідентифікатор `@+id/order_summary_text_view`.

2. У методі `displayMessage` файлу `MainActivity.java`:

- змінюємо назву змінної `priceTextView` на `orderSummaryTextView`;
- змінюємо `R.id.price_text_view` на `R.id.order_summary_text_view`

3. У файлі `MainActivity.java`:

- видалити або закоментувати (`ctrl + /` (рядок) або `ctrl + shift + /` (фрагмент коду)) метод `displayPrice`;
- внести зміни або додати методи `submitOrder`, `calculatePrice`, `createOrderSummary`

## Л15. Just Java. Change Price to Order Summary

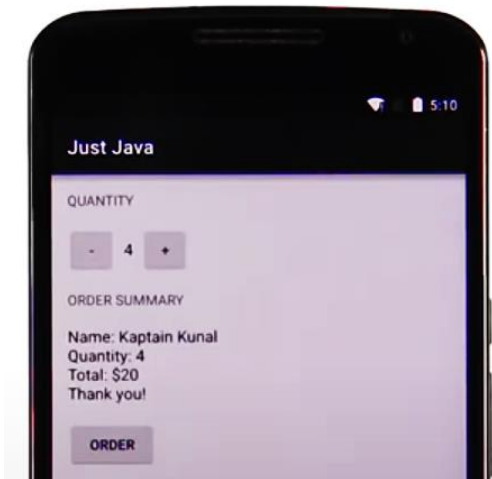
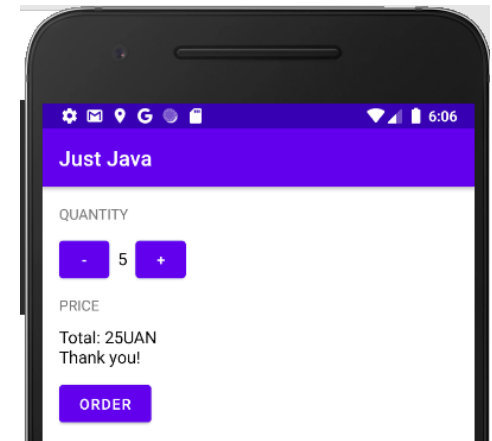
```
int numberOfCofees = 0;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
public void submitOrder(View view) {
    int price = calculatePrice();
    displayMessage(createOrderSummary(price));
}
private int calculatePrice() {
    int basePrice = 5;
    return numberOfCofees * basePrice;
}
private String createOrderSummary(int price) {
    String priceMessage = "Name: Mallyness\n";
    priceMessage += "Quantity: " + numberOfCofees + " \n";
    priceMessage += "Total: " + price + "$ \n";
    priceMessage += "Thank you!";
    return priceMessage;
}
private void displayQuantity(int number) {
    TextView quantityTextView = (TextView)
    findViewById(R.id.quantity_text_view);
    quantityTextView.setText("" + number);
}
```

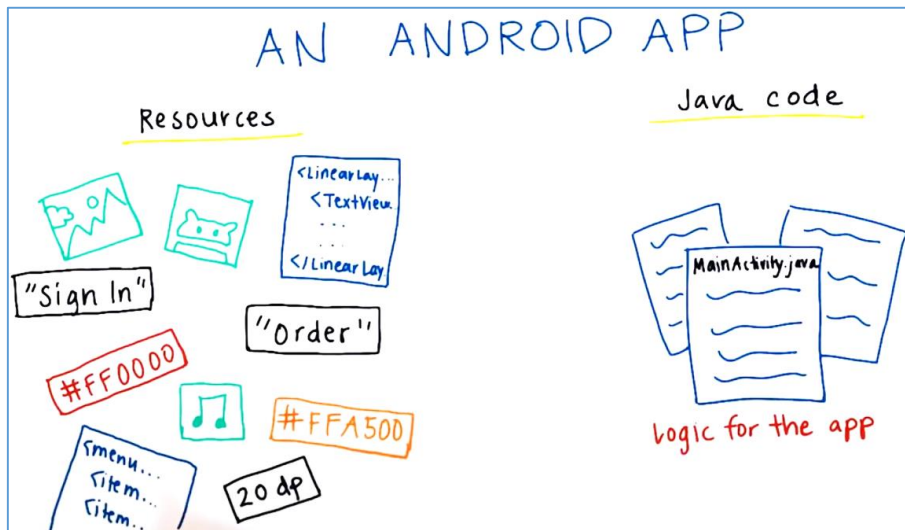


## 15. Just Java. Change Price to Order Summary

```
public void countPlus(View view) {
    numberOfCofees++;
    display(numberOfCofees);
}
public void countMinus(View view) {
    if(numberOfCofees>1) numberOfCofees--;
    display(numberOfCofees);
}
private void display(int number) {
    TextView quantityTextView = (TextView)
    findViewById(R.id.quantity_text_view);
    quantityTextView.setText("" + number);
}
private void displayMessage(String message) {
    TextView orderSummaryTextView = (TextView)
    findViewById(R.id.order_summary_text_view)
    orderSummaryTextView.setText(message);
}
```



## Л5. Ідентифікатори ресурсів. Об'єкти

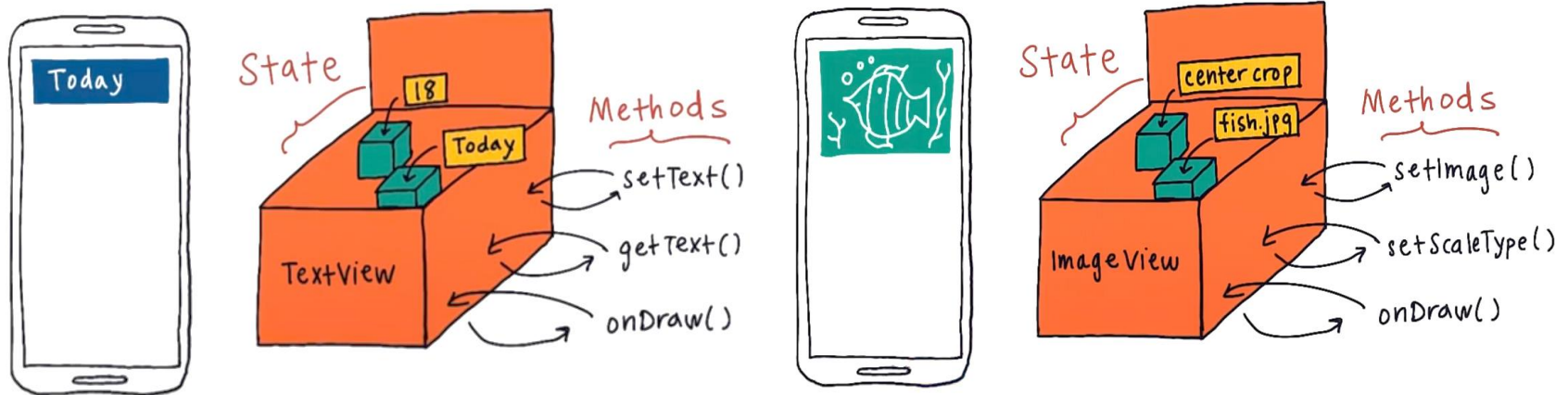


Ресурси Android (/res) можна поміщати у різні папки. Для звернення до графічних рисунків використовується формат **R.drawable**, для рядків **R.string.hello** де **R** це клас, **hello** це ім'я ресурсу. В кодї вказуємо ресурси: **R.тип.имя\_ресурса**, в розмітці вказуємо ресурси: **@тип/имя\_ресурса**.

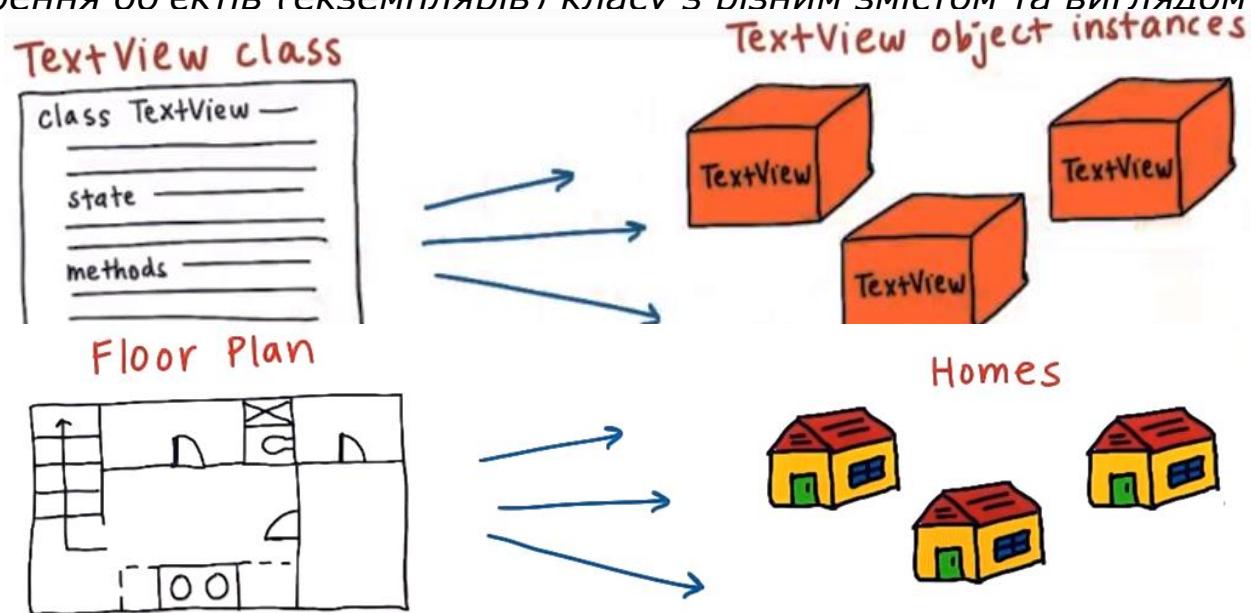
[Огляд різних ресурсів проектів](#)  
[Підтримка різних екранів](#)

Resource Type	In Java code	In XML
Image	R.drawable.photo	@drawable/photo
String "Hello"	R.string.hello	@string/hello
Layout XML file	R.layout.activity_main	@layout/activity_main
ID	R.id.pice_text_view	@id/price_text_view
Color #FF0000	R.color.red	@color/red

# Л5. Об'єкти та їх методи в Java



**TextView** це клас, який має поля стану та методи. Цей клас використовується для створення об'єктів (екземплярів) класу з різним змістом та виглядом



План дома не є самим домом. Кожний дом відрізняється але усі вони мають однаковий каркас (план).

## Л5. Класи в Java

Клас в Java визначається ключовим словом *class*, назвою *TextView*, деяких параметрів та `{}`.

[Детальніше про класи в Java](#)

[Спрощений код \*TextView\*](#)

[Спрощений код \*ImageView\*](#)

[Повний код \*TextView\*](#)

[Повний код \*ImageView\*](#)

```
public class ImageView extends View {  
  
    // Resource ID for the source image that should be displayed in the ImageView.  
    private int mImageId;           змінні  
                                     починаються з m  
                                     (Member Variable )  
  
    // Context of the app  
    private Context mContext;  
  
    /**  
     * Constructs a new ImageView.  
     */  
    public ImageView(Context context) {           конструктор класу  
        mImageId = 0;  
        mContext = context;  
    }  
  
    /**  
     * Sets the source image in the ImageView.  
     *  
     * @param imageId is the resource ID of the image to be displayed.  
     */  
    public void setImage(int imageId) {  
        mImageId = imageId;  
    }  
}
```

Щоб отримати доступ до вихідного коду, вбудованому в документацію будь-якого з класів Андроїд, встановіть це [розширення для Google Chrome](#)

## Л5. Класи в Java. Конструктори

Конструктори викликаються щоб створити екземпляр класу.

Object  
Data  
Type

Variable  
name

= new

Object  
Data  
Type

( Input  
Args ) ;

```
TextView priceTextView = new TextView(context);  
ImageView coffeeImageView = new ImageView(context);  
Button submitOrderButton = new Button(context);  
CatView sleepyCatView = new CatView("Tiger");
```

Contex об'єкт дозволяє отримати доступ до ресурсів та інших частин додатку

### CREATE OBJECT WITH FACTORY METHODS

Object  
Data  
Type

Variable  
Name

=

Object  
Data  
Type

•

Factory  
Method  
Name

( Input  
Arguments ) ;

```
MediaPlayer player = MediaPlayer.create(context, R.raw.song);  
Toast toastMessage = Toast.makeText(context, "Hi", duration);
```

## Л5. Методи

Завдання 4. Знайти помилку при створені коду об'єкта та виправити її.

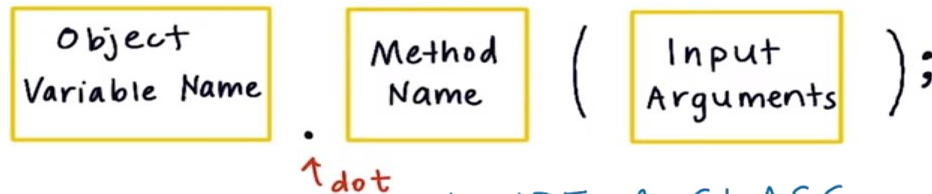
1	<i>TextView = new TextView (context);</i>
2	<i>Image view img = new ImageView(context);</i>
3	<i>ToggleButton button = create ToggleButton(context);</i>
4	<i>Toast toast = toast.makeText(context, text, duration);</i>

# Л5. Класи в Java. Конструктори

Розглянемо ще раз метод **displayMessage**

```
private void displayMessage(String message) {  
    TextView orderSummaryTextView = (TextView) findViewById(R.id.order_summary_text_view);  
    orderSummaryTextView.setText(message);  
}
```

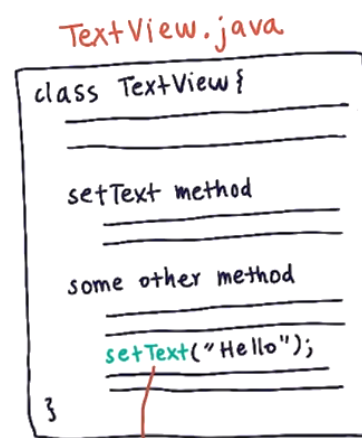
Він містить об'єкт **TextView** з ім'ям **orderSummaryTextView**. У наступному рядку викликаємо метод `setText` для даного об'єкту. Для запуску методу на об'єкті у нього буде такий формат



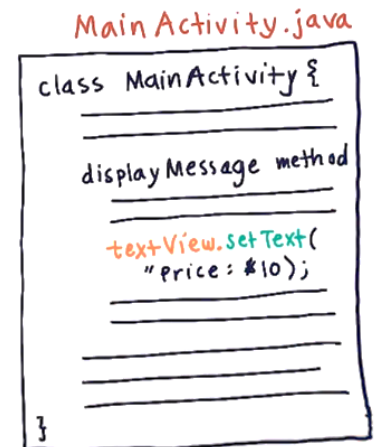
Якщо знаходимося в середині класу і викликаємо метод `setText` з іншого методу даного класу то при виклику методу крапка не ставиться. Якщо викликаємо метод класу зовні з іншого місця, тоді вказується через точку. Доступ є тільки до **public** змінних та методів.

INSIDE A CLASS

vs. OUTSIDE A CLASS



Same as `this.setText("Hello");`



```
titleTextView.setText("News");  
titleTextView.setTextSize(18);  
warningTextView.setTextColor(Color.RED);  
welcomeImageView.setImageResource(R.drawable.cloud);
```

## Л5. Inheriting Behavior (Успадковуємо поведінку)

У попередньо розглянутому кодї що означає `findViewById`?

```
private void displayMessage(String message) {  
    TextView orderSummaryTextView = (TextView) findViewById(R.id.order_summary_text_view);  
    orderSummaryTextView.setText(message);  
}
```

Якщо переглянути повний код, то ніде не знайдемо метод **`findViewById`**. Код **`MainActivity.java`** не великий, але можна побачити такий рядок з кодом **`extends AppCompatActivity`**:

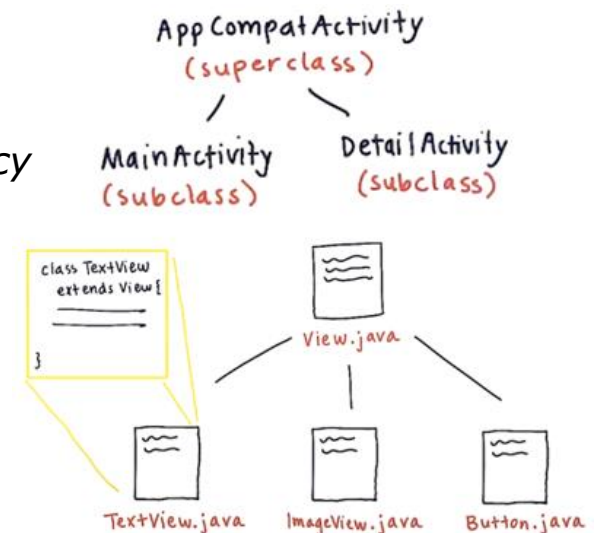
```
public class MainActivity extends AppCompatActivity {  
    int numberOfCofees = 0;
```

Це значить, що **`MainActivity`** це розширення класу **`AppCompatActivity`** і відповідно **`MainActivity`** успадковує (отримує доступ) до методів **`AppCompatActivity`**.

Класи `TextView`, `ImageView`, `Button` успадковують усі властивості та методи класу `View` але в них знаходяться методи та поля, що притаманні тільки ним.

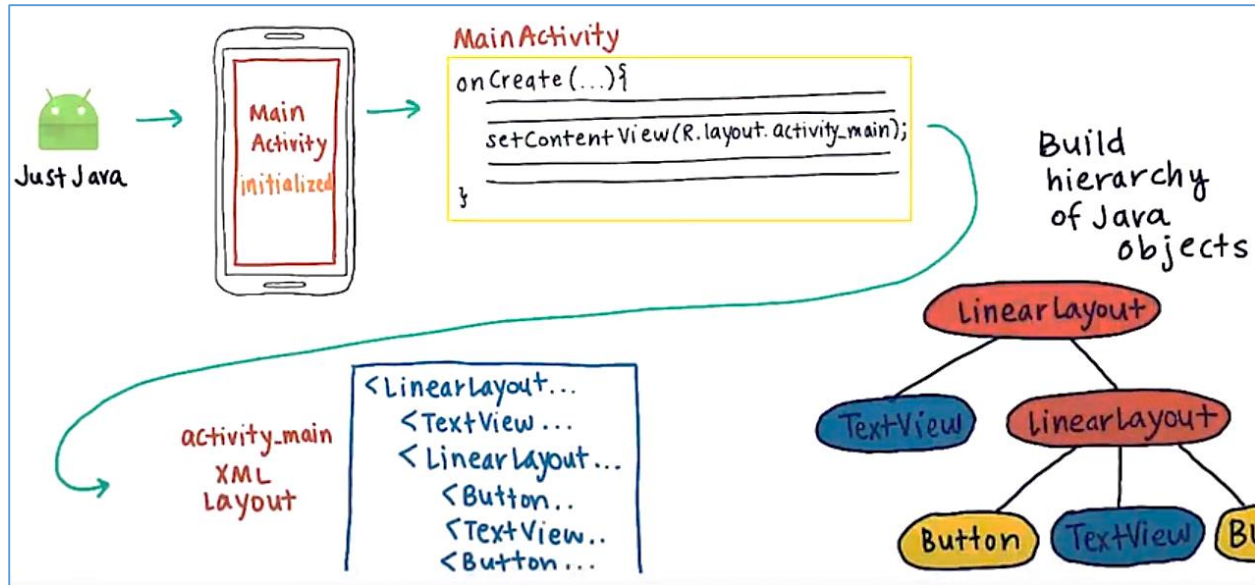
Діаграма ієрархії класів

```
java.lang.Object  
↳ android.view.View  
↳ android.widget.TextView  
↳ android.widget.EditText
```





## Л5. Приведення типів даних



Якщо натиснути на іконку додатку на пристрої відкривається основна функція **MainActivity** – **onCreate**. У методі викликається метод **setContentView** з макетом ресурсу **activity.main**. Так як хочемо мати більше 1 елементу для додатку, то вводимо макет ресурсу та прив'язуємо до **xml** файлу. Функція знаходить файл макету **xml** та зчитує кожний рядок та буде повну ієрархію об'єктів java. Під час створення кожного з елементів викликається конструктор **new TextView** або **new LinearLayout**, далі передаються всі дані атрибути в об'єкт java. В XML файлі для ідентифікації об'єктів їм присвоюємо ID. В класі **AppCompatActivity** є метод **findViewById (int id)**, який знаходить елемент за його ID

## Л5. Приведення типів даних

Результат отриманий з **findViewById** (*R.id.id\_name*) повертає *View* згідно документації. Однак в правій частині створюється елемент *TextView*, значить потрібно *View* привести до типу *TextView*. Це робиться через дужки з *TextView*

```
private void displayMessage(String message) {  
    TextView orderSummaryTextView = (TextView) findViewById(R.id.order_summary_text_view);  
    orderSummaryTextView.setText(message);  
}
```

Як використовувати метод *setText* для зміни *View*-елемента? Це робиться в два етапи:

Крок 1: Отримання об'єкта **View** за допомогою його ідентифікатора. У першому рядку коду отримуємо **TextView** і зберігаємо його в змінну під назвою **orderSummaryTextView** (текстове поле замовлення). Щоб отримати **View**, використовуємо метод **findViewById** (отримати *view* за ідентифікатором), це метод з класу **Activity**. У нього передаємо один аргумент - ідентифікатор **view**, цей аргумент надаємо в вигляді **R.id.IDOFVIEW**. В даному випадку ідентифікатор *View* заданий в XML - **order\_summary\_text\_view**. Далі відбувається приведення об'єктів, цим займається (**TextView**). Тобто повертається значення методу **findViewById** повинно бути типу **TextView**, а не просто *View*.

## Л5. Приведення типів даних

```
private void displayMessage(String message) {  
    TextView orderSummaryTextView = (TextView) findViewById(R.id.order_summary_text_view);  
    orderSummaryTextView.setText(message);  
}
```

Крок 2: Виклик методу на об'єкті *View*. Оскільки викликаємо метод на об'єкті, потрібно використовувати виклик через точку. Рядок **`orderSummaryTextView.setText`** (*message*) - все одно що інструкція візьми об'єкт *orderSummaryTextView*, якому можна задати текст з рядка, який передали в метод (в даному випадку передали змінну типу *String* під назвою *message*).

Методи, як **`setText`** (встановити текст) і **`setImageResource`** (встановити картинку) називають сетерами. Вони встановлюють одне зі значень *View* (наприклад, зберігається в ньому текст або картинку). Вони традиційно починаються зі слова **`set`** (встановити).

Є ще така категорія методів, як геттери. Їх єдине призначення - отримати одне зі значень *View*, наприклад, текст, який на даний момент встановлено. Вони традиційно починаються зі слова **`get`** (отримати).

## Л5. Методи

**Завдання 5.** Нище наведені фрагменти коду, які можуть містити помилку, що відносяться до невідповідності типів даних. Якщо код вірний, то пишть, що код без помилок (No error)

1	<pre>View nameTextView = findViewById(R.id.name); nameTextView.setText("Laura");</pre>
2	<pre>TextView description=(TextView) findViewById(R.id.description); String maxLines = description.getMaxLines();</pre>
3	<pre>ImageView iconImageView = findViewById(R.id.icon); iconImageView.setImageResource(R.drawable.logo);</pre>
4	<pre>View textView = findViewById(R.id.title); textView.setVisibility(View.GONE);</pre>