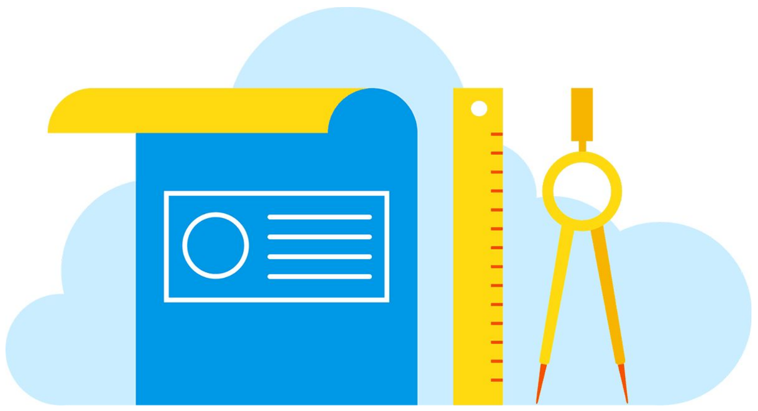


Android Devs Training– 1st Week

Training Manual

Trainer: Haris Hashim

Date: 23/07/2018

Table Of Contents

Overview & Purpose

Introduction to Android Studio and Java coding review

Topic Covered

1. Getting around in Android Studio IDE
2. Java coding refresher
3. Example of JAVA code in Android development

Objectives

1. Getting familiar with IDE
2. Prepare developer with basic coding concept before future lessons
3. Expose developers to quick tips and shortcut when typing codes using Android Studio

Materials Needed

1. Notebook or PC with Android Studio installed and fully working

Part 1: Setting up JAVA code

1. Open Android and create a new Android Studio Project.

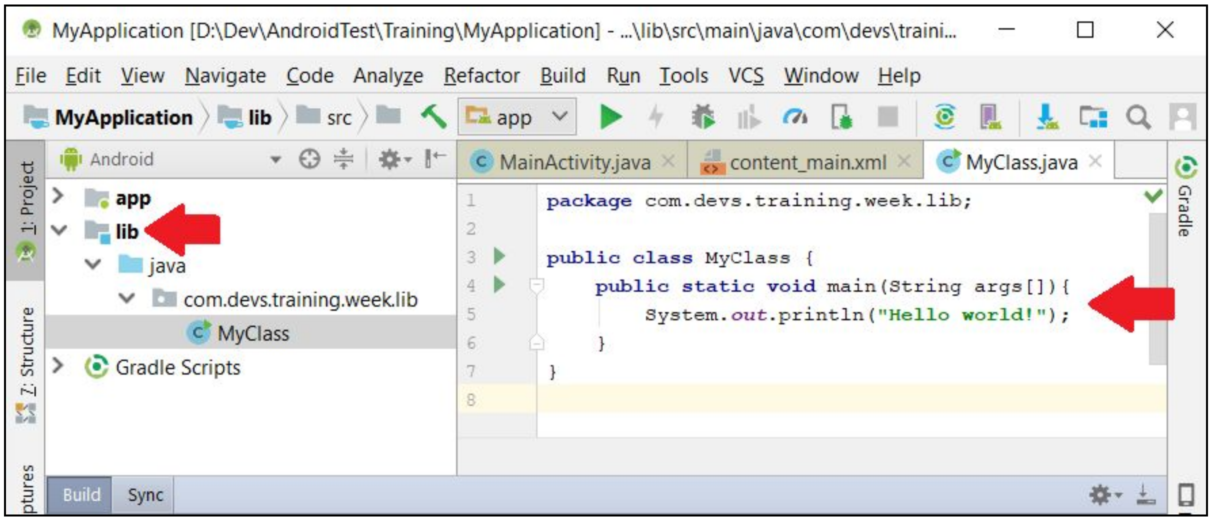
   Use default value in Create New Project dialogs.

2. In the newly created project add new module using main menu.

   File => New => New Module

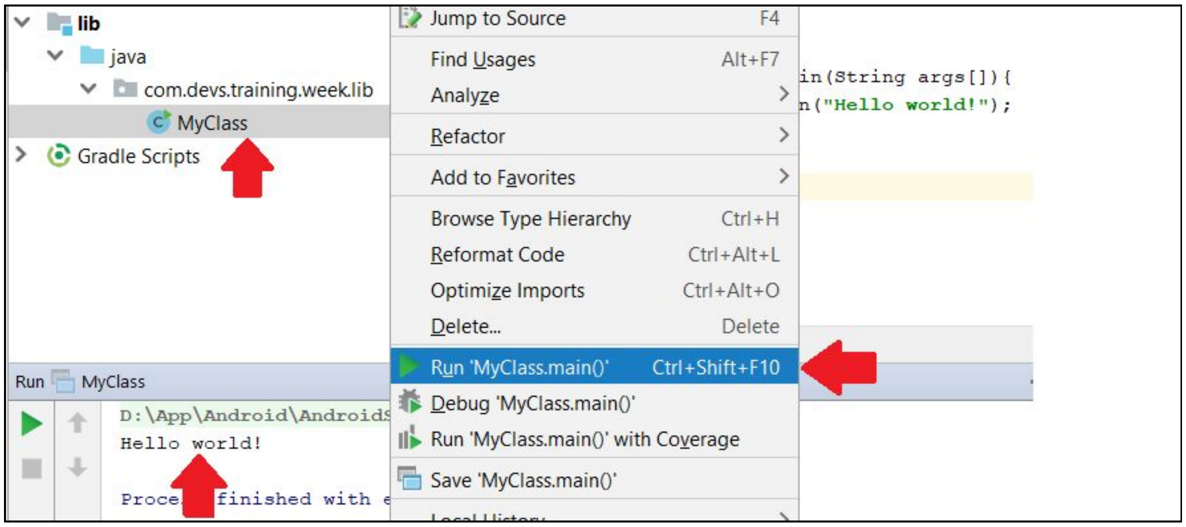
   Choose "Java Library" and use default value in Create New Module dialog

3. Expand lib folder and subfolders to find MyClass.java file and type bellow code.


MyApplication [D:\Dev\AndroidTest\Training\MyApplication] - ...\lib\src\main\java\com\devs\traini...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication lib src app
Android
MainActivity.java content_main.xml MyClass.java
app
lib
java
com.devs.training.week.lib
MyClass
Gradle Scripts
Build Sync

```
package com.devs.training.week.lib;

public class MyClass {
    public static void main(String args[]){
        System.out.println("Hello world!");
    }
}
```

4. Right click on MyClass file and run it! Note the "Hello world!" output location.



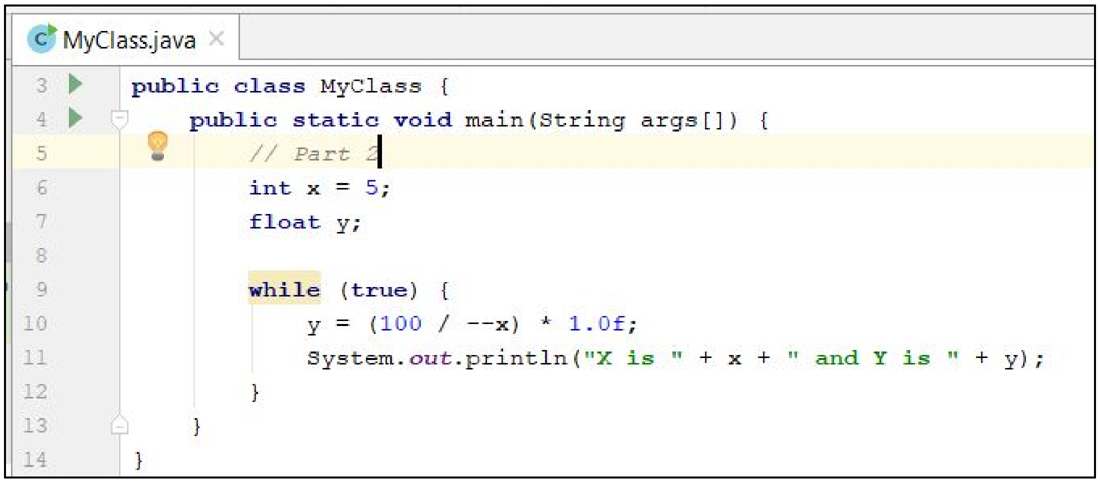
```
D:\App\Android\Androids
Hello world!

Proce  finished with e
```

Part 2: Coding and Debugging

Concepts & Demonstration

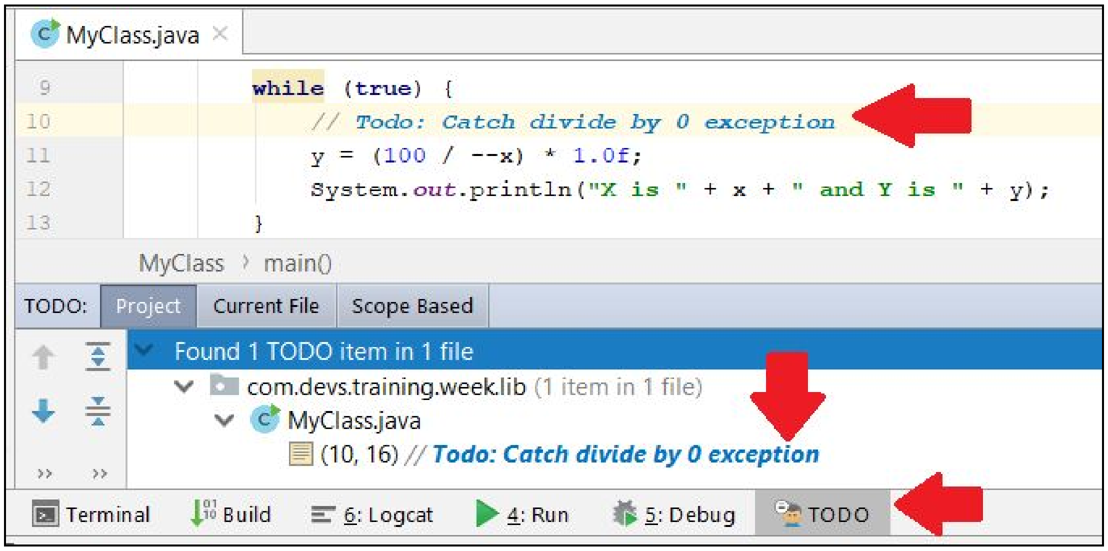
1. System.out and System.err
2. How to add Todo
3. Debugging using Android Studio
4. Tips to tidy up using Reformat Code (CTRL-ALT-L) - ASK ME!
5. Tips to split bottom tab to display Todo & Debug at the same time - ASK ME!

1. Continuing from Part 1. Modify MyClass to be as follows. Run and observe the output to see that there is a bug at line 10 of bellow image.



```
3   public class MyClass {
4       public static void main(String args[]) {
5           // Part 2
6           int x = 5;
7           float y;
8
9           while (true) {
10              y = (100 / --x) * 1.0f;
11              System.out.println("X is " + x + " and Y is " + y);
12          }
13      }
14  }
```

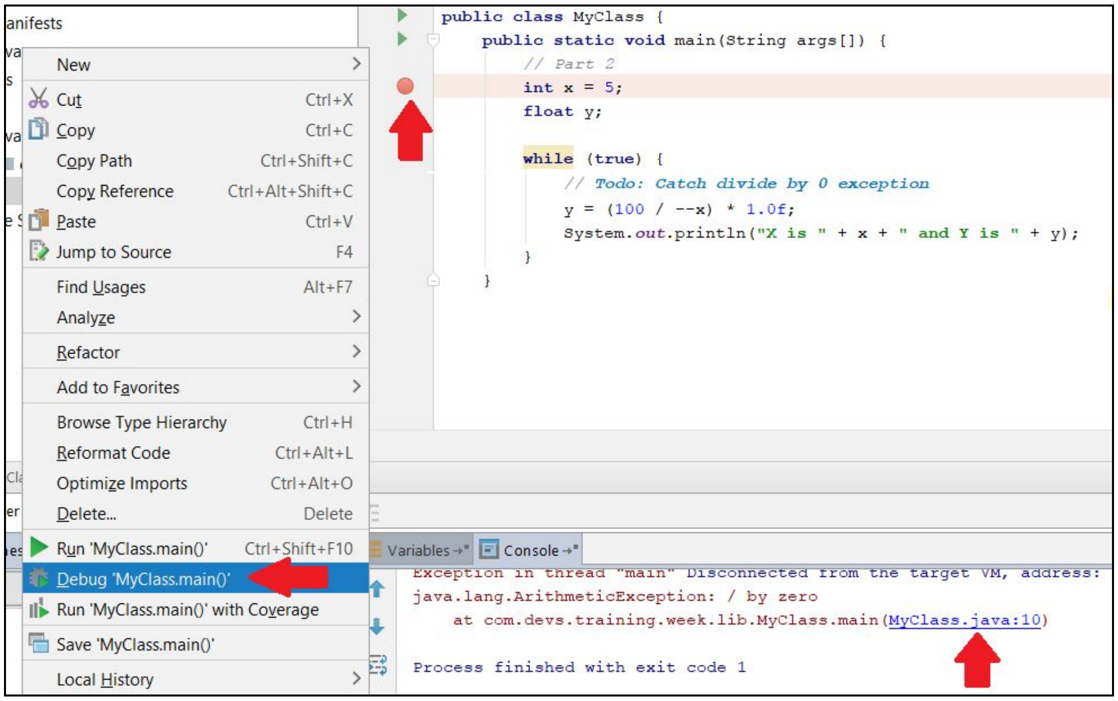
2. A cool tips for developers is to add Todo item to easily find and fix bugs later!



```
9           while (true) {
10              // Todo: Catch divide by 0 exception
11              y = (100 / --x) * 1.0f;
12              System.out.println("X is " + x + " and Y is " + y);
13          }
```

```
Found 1 TODO item in 1 file
  com.devs.training.week.lib (1 item in 1 file)
    MyClass.java
      (10, 16) // Todo: Catch divide by 0 exception
```

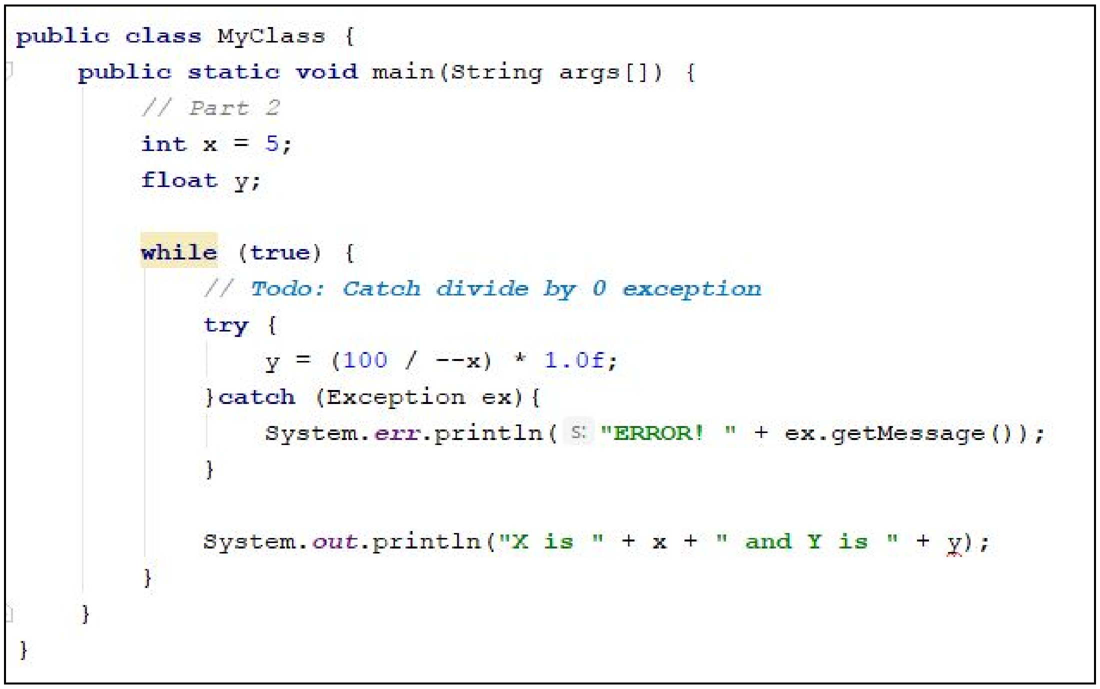
3. Set breakpoint and perform debugging to observe code flow and identify error.



```java
public class MyClass {
    public static void main(String args[]) {
        // Part 2
        int x = 5;
        float y;

        while (true) {
            // Todo: Catch divide by 0 exception
            y = (100 / --x) * 1.0f;
            System.out.println("X is " + x + " and Y is " + y);
        }
    }
```

```
Exception in thread "main" Disconnected from the target VM, address:
java.lang.ArithmeticException: / by zero
    at com.devs.training.week.lib.MyClass.main(MyClass.java:10)

Process finished with exit code 1
```

4. Modify the code to catch exception and run or debug again!



```java
public class MyClass {
    public static void main(String args[]) {
        // Part 2
        int x = 5;
        float y;

        while (true) {
            // Todo: Catch divide by 0 exception
            try {
                y = (100 / --x) * 1.0f;
            }catch (Exception ex){
                System.err.println( "ERROR! " + ex.getMessage());
            }

            System.out.println("X is " + x + " and Y is " + y);
        }
    }
}
```

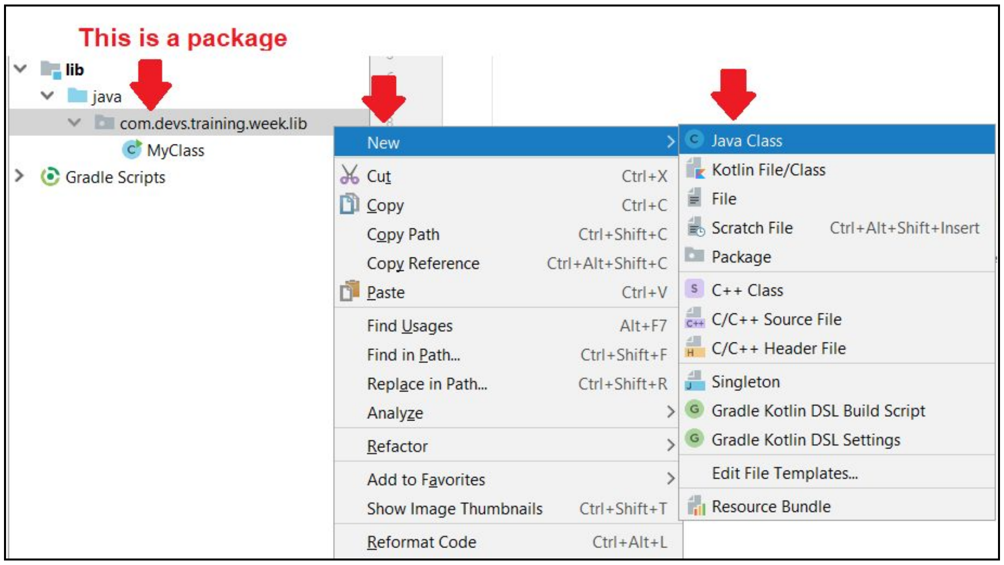
5. After the bug is fixed, remove the TODO comment.
6. We just introduced another bug in step 4! As exercise, find and fix it.

Part 3: Class and Exception Handling

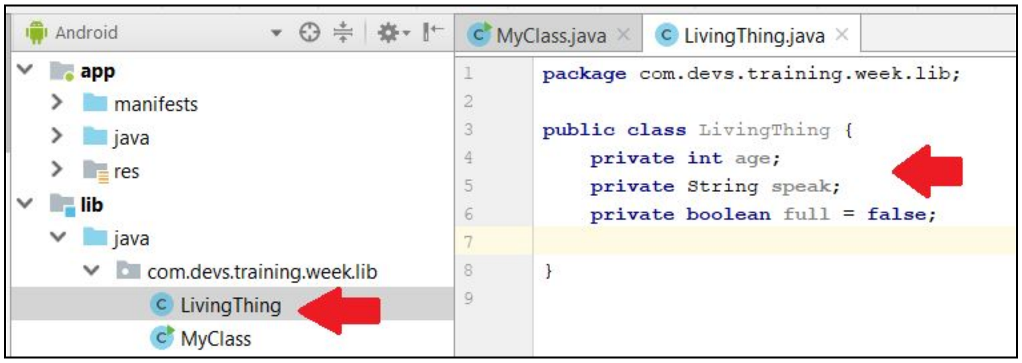
Concepts & Demonstration

1. Class
2. Quick tips ALT-INSERT
   a. Create constructor
   b. Create getter & setter
3. Quick tips using light bulbs to generate code
   a. Throwing error
   b. Catching error

1. Quickly add another class in the same package as MyClass. Right click the package and choose New => Java Class

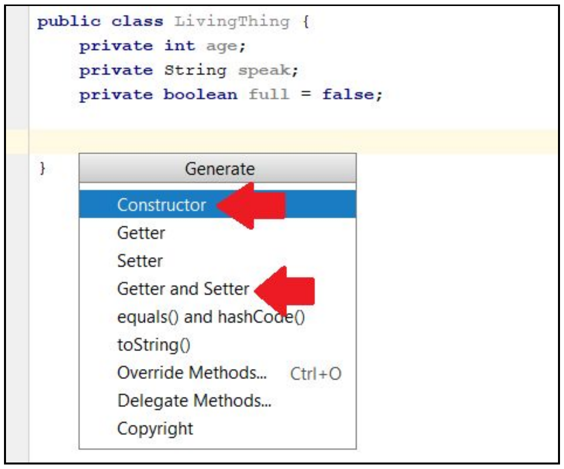


2. In Create New Class dialog, name the class LivingThing and press OK.
3. Modify LivingThing class code.



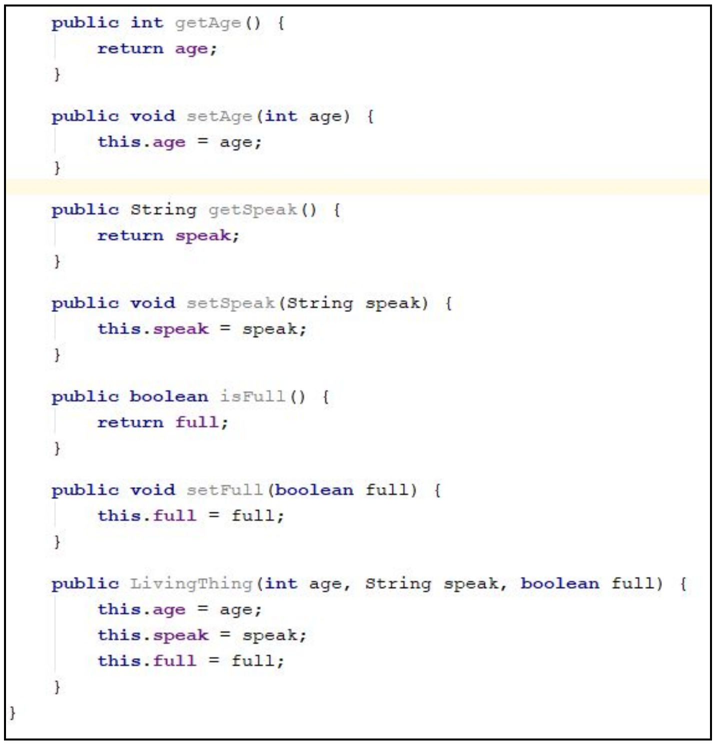
```
package com.devs.training.week.lib;

public class LivingThing {
    private int age;
    private String speak;
    private boolean full = false;

}
```

4. As a quick tips we are going to use ALT-INSERT to quickly insert code into the class. As general rule of thumb, place cursor where the code to be inserted and then press ALT-INSERT.



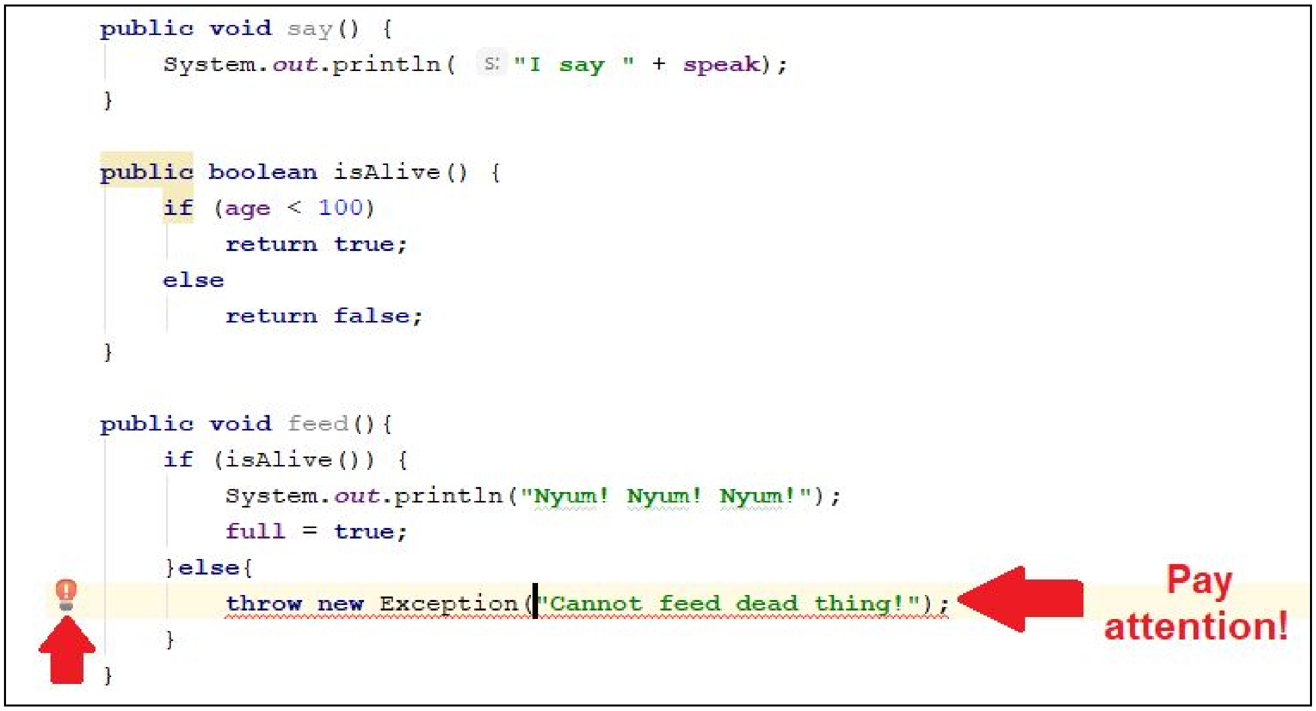
```
public class LivingThing {
    private int age;
    private String speak;
    private boolean full = false;

}
```

5. Insert Constructor that initialize all the class private variable. In the next dialog, use shift to select variables.
6. Insert Getter and Setter for all class private variable. In the next dialog, use shift to select variables
7. Generated code should be as follows.



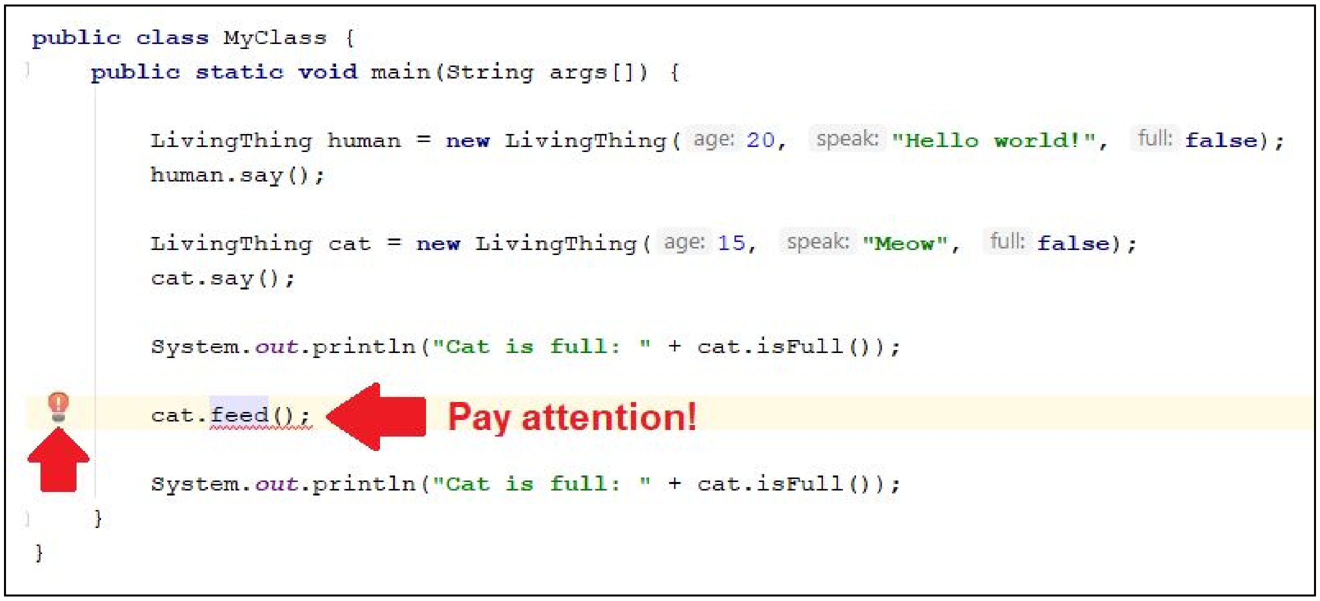
```
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getSpeak() {
        return speak;
    }

    public void setSpeak(String speak) {
        this.speak = speak;
    }

    public boolean isFull() {
        return full;
    }

    public void setFull(boolean full) {
        this.full = full;
    }

    public LivingThing(int age, String speak, boolean full) {
        this.age = age;
        this.speak = speak;
        this.full = full;
    }
}
```

8. Add additional methods or actions for the class.



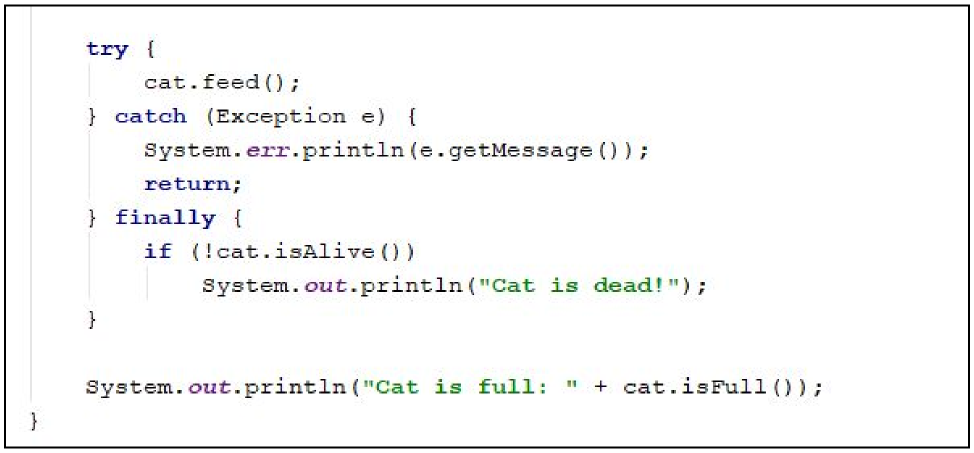
```
    public void say() {
        System.out.println( s: "I say " + speak);
    }

    public boolean isAlive() {
        if (age < 100)
            return true;
        else
            return false;
    }

    public void feed(){
        if (isAlive()) {
            System.out.println("Nyum! Nyum! Nyum!");
            full = true;
        }else{
            throw new Exception("Cannot feed dead thing!");
        }
    }
```

9. Click the red bulb in above picture and select “Add exception to method signature”. After doing this, we will have a method that throw exception!
10. Go back to MyClass and modify the code as below



```
public class MyClass {
    public static void main(String args[]) {

        LivingThing human = new LivingThing( age: 20, speak: "Hello world!", full: false);
        human.say();

        LivingThing cat = new LivingThing( age: 15, speak: "Meow", full: false);
        cat.say();

        System.out.println("Cat is full: " + cat.isFull());

        cat.feed();

        System.out.println("Cat is full: " + cat.isFull());
    }
}
```

11. Click the red bulb and select “Surround with try/catch”. Try-catch codes will be automatically created for us! Modify the try-catch to be as below



```
    try {
        cat.feed();
    } catch (Exception e) {
        System.err.println(e.getMessage());
        return;
    } finally {
        if (!cat.isAlive())
            System.out.println("Cat is dead!");
    }

    System.out.println("Cat is full: " + cat.isFull());
}
```

12. Set a breakpoint and debug through code execution to better understand the code flow. Change cat age from 15 to 115 to see error being thrown.

Part 4: More Concepts

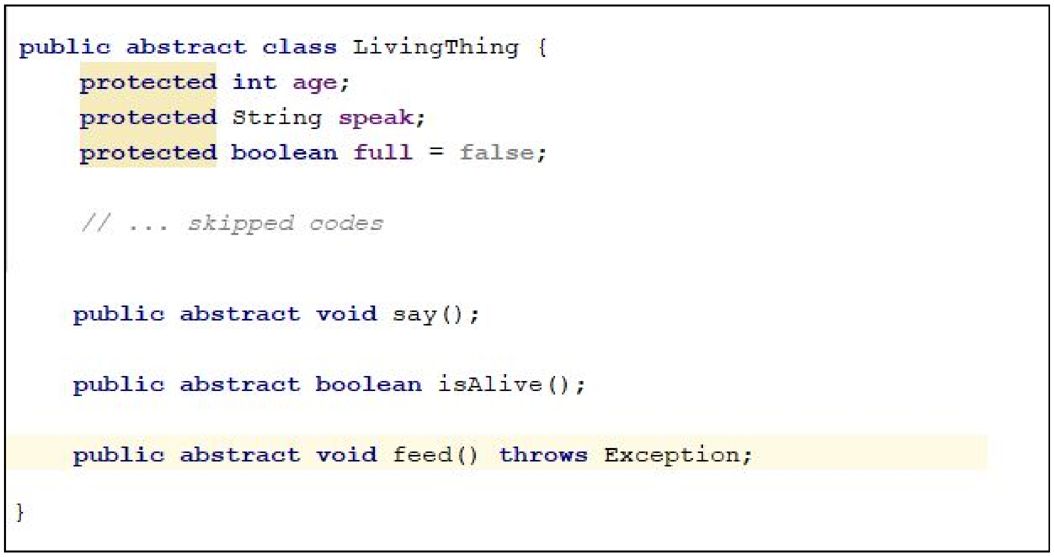
Concepts & Demonstration

1. Abstract Class
2. Interface
3. Listener and Callback

1. **Abstract Class**

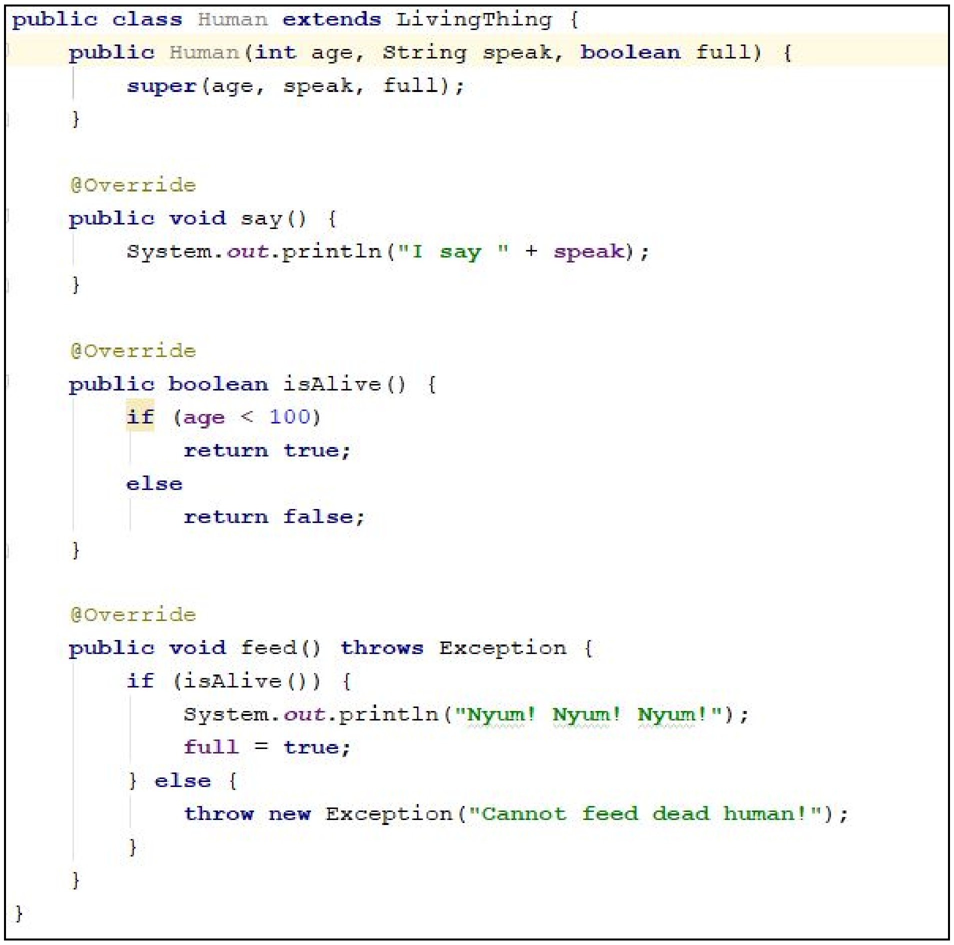
   Let's make some changes to demonstrate OOP inheritance and polymorphism concept. We will turn LivingThing into an Abstract Class as well as add another 2 child class; Human and Animal that inherit from LivingThing.

2. Codes for LivingThing.



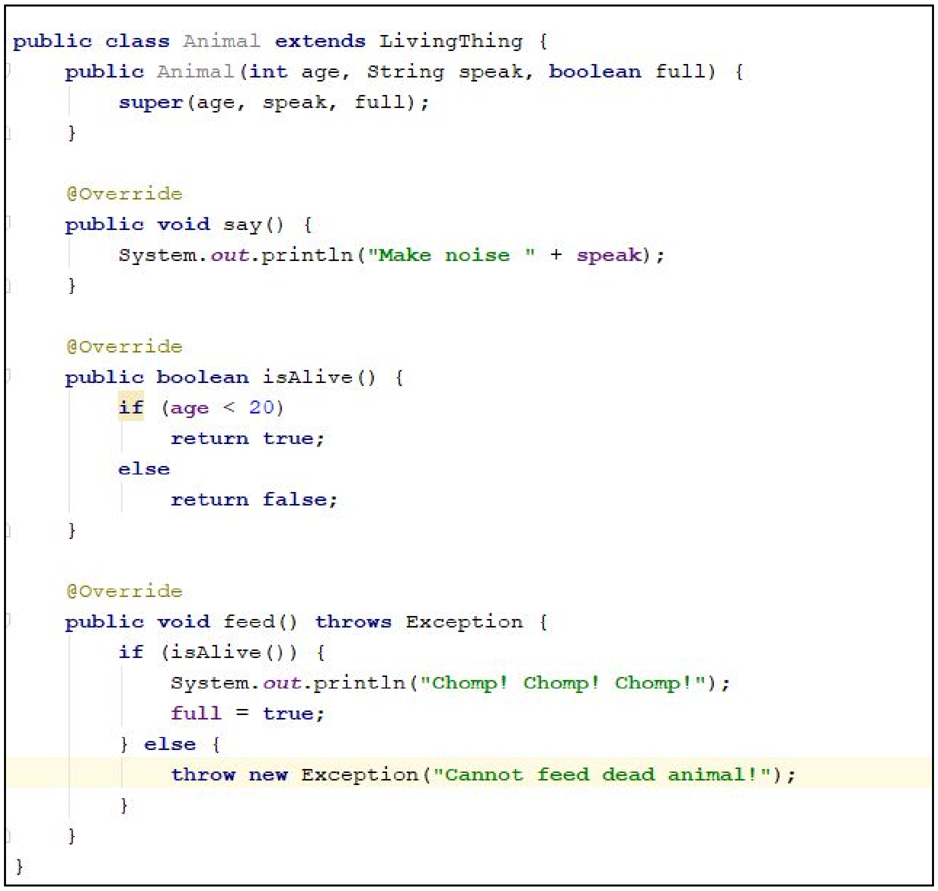
```java
public abstract class LivingThing {
    protected int age;
    protected String speak;
    protected boolean full = false;

    // ... skipped codes


   public abstract void say();

   public abstract boolean isAlive();

   public abstract void feed() throws Exception;

}
```

3. Codes for Human.
   - Take note to set LivingThing as Superclass in Create New Class dialog.
   - Take note to "Implement methods" using light bulb shortcut.
   - Take note to "Create constructor matching Super" using light bulb shortcut.



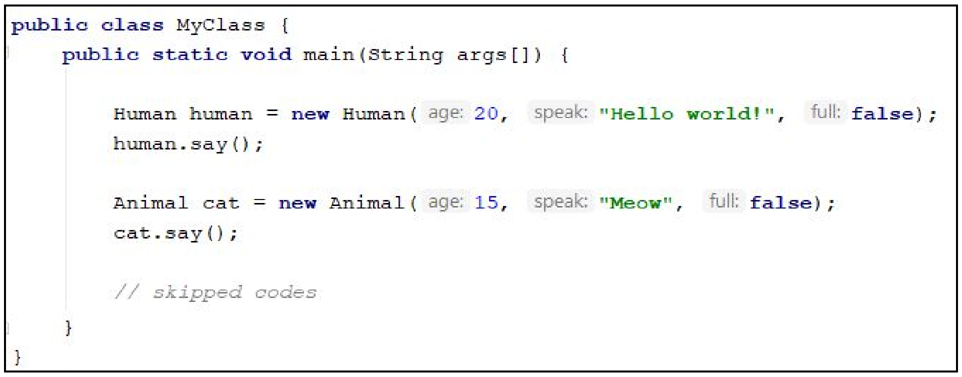
```java
public class Human extends LivingThing {
    public Human(int age, String speak, boolean full) {
        super(age, speak, full);
    }

    @Override
    public void say() {
        System.out.println("I say " + speak);
    }

    @Override
    public boolean isAlive() {
        if (age < 100)
            return true;
        else
            return false;
    }

    @Override
    public void feed() throws Exception {
        if (isAlive()) {
            System.out.println("Nyum! Nyum! Nyum!");
            full = true;
        } else {
            throw new Exception("Cannot feed dead human!");
        }
    }
}
```

4. Codes for Animal.
    - Take note to set LivingThing as Superclass in Create New Class dialog.
    - Take note to “Implement methods” using light bulb shortcut.
    - Take note to “Create constructor matching Super” using light bulb shortcut.



```java
public class Animal extends LivingThing {
    public Animal(int age, String speak, boolean full) {
        super(age, speak, full);
    }

    @Override
    public void say() {
        System.out.println("Make noise " + speak);
    }

    @Override
    public boolean isAlive() {
        if (age < 20)
            return true;
        else
            return false;
    }

    @Override
    public void feed() throws Exception {
        if (isAlive()) {
            System.out.println("Chomp! Chomp! Chomp!");
            full = true;
        } else {
            throw new Exception("Cannot feed dead animal!");
        }
    }
}
```

Codes for MyClass.



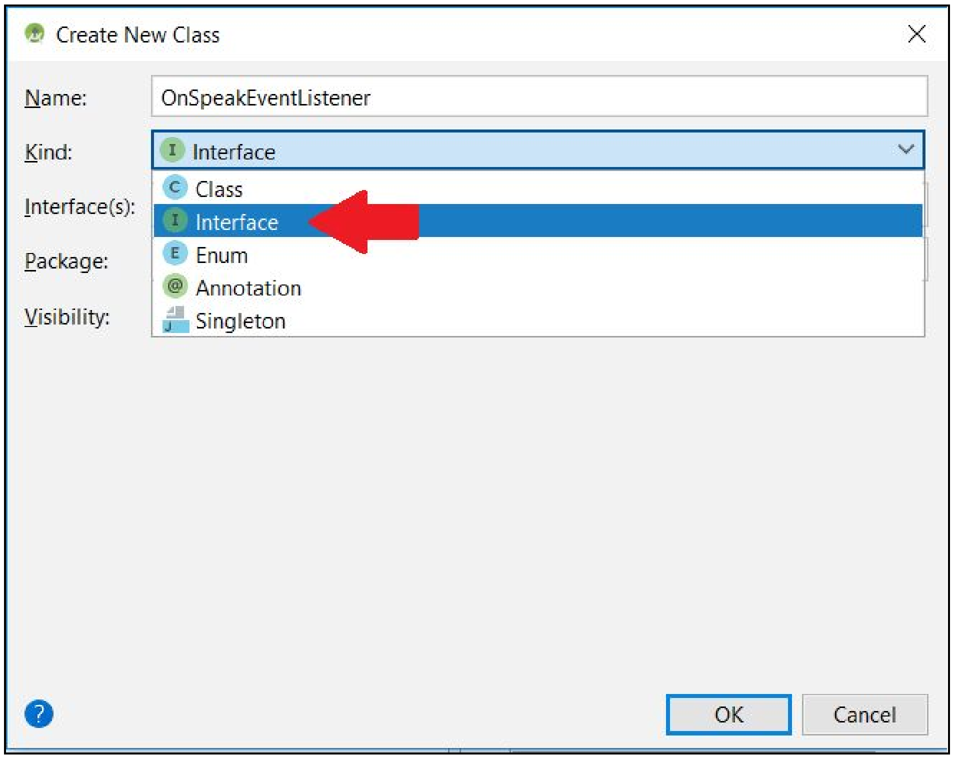
```java
public class MyClass {
    public static void main(String args[]) {

        Human human = new Human( age: 20, speak: "Hello world!", full: false);
        human.say();

        Animal cat = new Animal( age: 15, speak: "Meow", full: false);
        cat.say();

        // skipped codes
    }
}
```

Set a breakpoint and do debug. Change cat age as necessary to see output.

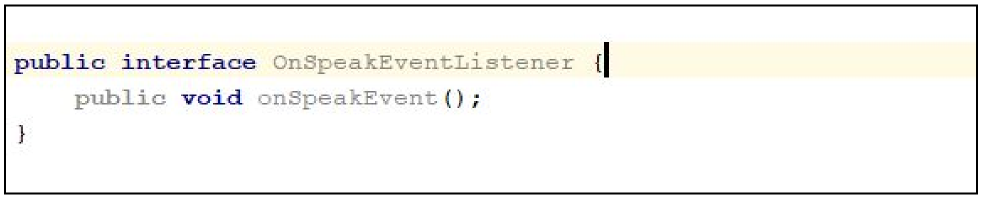
5. **Interface**

   For this section, we are going to use interface to implement event listener and callback for Human and Animal class. A very common pattern in Android development.

   New Interface is created the same way as class. By right clicking package name and choose New => Java Class. However, set the Kind to Interface in Create New Class dialog as below image.

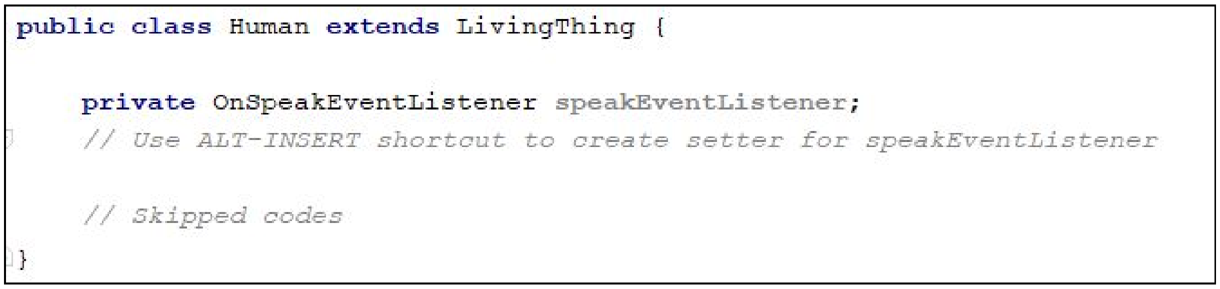


6. Create OnSpeakEventListener interface with bellow codes



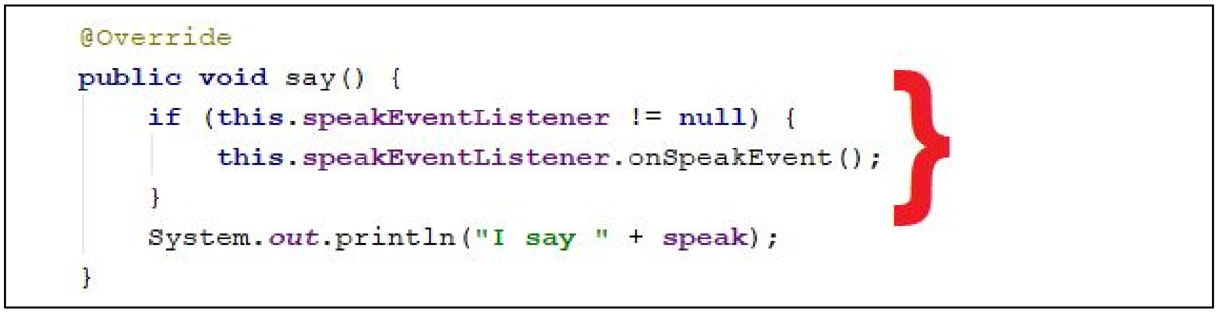
```java
public interface OnSpeakEventListener {
    public void onSpeakEvent();
}
```

7. Create OnSpeakEventListener member variable in Human class and generate setter using ALT-INSERT shortcut.



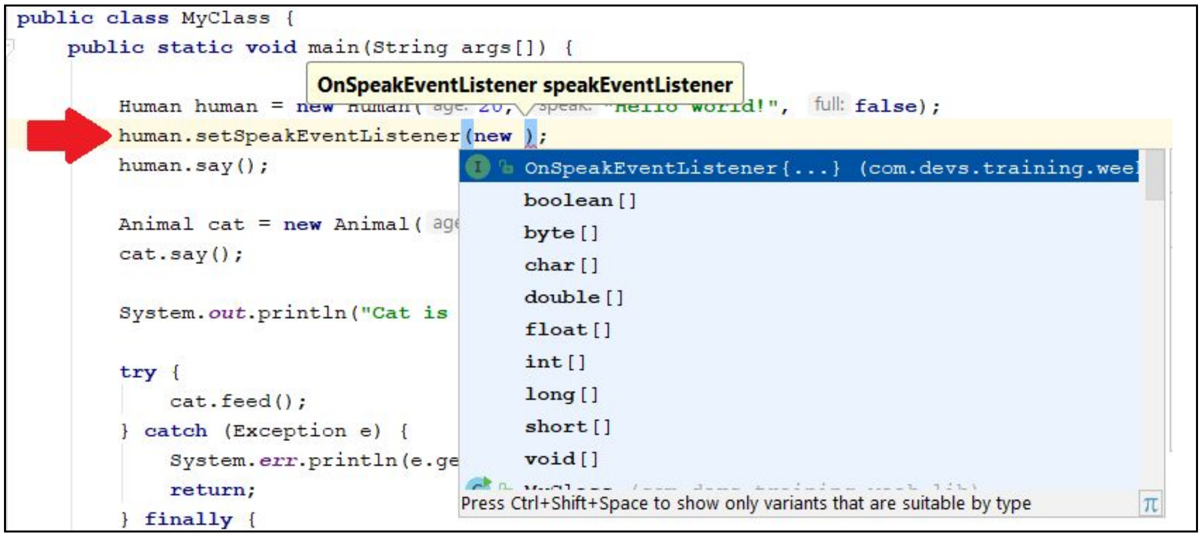
```
public class Human extends LivingThing {

    private OnSpeakEventListener speakEventListener;
    // Use ALT-INSERT shortcut to create setter for speakEventListener

    // Skipped codes
}
```

8. Modify say() method in Human class to trigger the speakEventListener.onSpeakEvent() method as per below code.



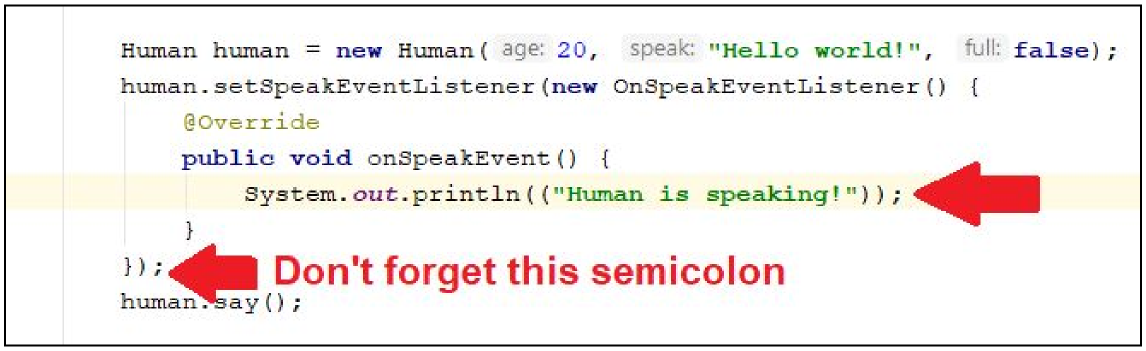
```
    @Override
    public void say() {
        if (this.speakEventListener != null) {
            this.speakEventListener.onSpeakEvent();
        }
        System.out.println("I say " + speak);
    }
```

9. Now we need to modify MyClass to define a callback to be called by speakEventListener in Human class. Type the code as below. The trick is to add space after new and press CTRL-SPACE to generate the callback code.



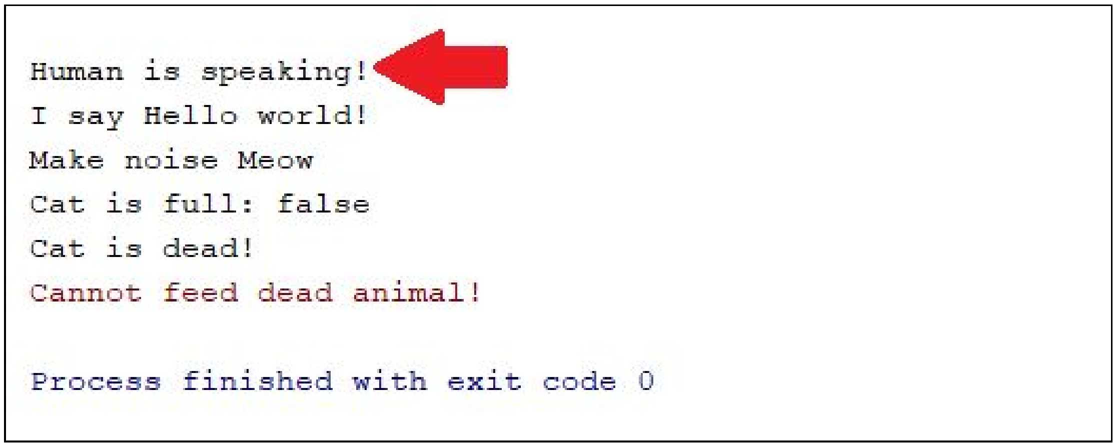
```
public class MyClass {
    public static void main(String args[]) {
                          OnSpeakEventListener speakEventListener
        Human human = new Human( age: 20, speak: "Hello world!", full: false);
        human.setSpeakEventListener(new );
        human.say();                   OnSpeakEventListener{...} (com.devs.training.wee
                                       boolean[]
        Animal cat = new Animal( age   byte[]
        cat.say();                     char[]
                                       double[]
        System.out.println("Cat is     float[]
                                       int[]
        try {                          long[]
            cat.feed();                short[]
        } catch (Exception e) {        void[]
            System.err.println(e.ge
            return;                    Press Ctrl+Shift+Space to show only variants that are suitable by type
        } finally {
```

10. Complete the generated callback code as per below picture.



```
Human human = new Human( age: 20,  speak: "Hello world!",  full: false);
human.setSpeakEventListener(new OnSpeakEventListener() {
    @Override
    public void onSpeakEvent() {
        System.out.println(("Human is speaking!"));
    }
});
human.say();
```

11. Set breakpoint and debug to go through the code flow. Ensure that “Human is speaking!” is shown in the output.



```
Human is speaking!
I say Hello world!
Make noise Meow
Cat is full: false
Cat is dead!
Cannot feed dead animal!

Process finished with exit code 0
```

12. Repeat step 8 to 11 on Animal class to add the same event listener and callback to cat.