**Inner Classes, Anonymous classes & Lambda**

**github-classClassroom Assignment**

* [Acceptance Link](https://classroom.github.com/a/fOIlIi54)

Example of an inner class:

class OuterClass {

    int x = 10;

    class InnerClass {

        int y = 5;

    }

}

public class Main {

    public static void main(String[] args) {

        OuterClass myOuter = new OuterClass();

        OuterClass.InnerClass myInner = myOuter.new InnerClass();

        System.out.println(myInner.y + myOuter.x);

    }

}

// Outputs

15 //(5 + 10)

Example of a private inner class:

class OuterClass {

    int x = 10;

**private** class InnerClass {

      int y = 5;

    }

}

public class Main {

    public static void main(String[] args) {

        OuterClass myOuter = new OuterClass();

        OuterClass.InnerClass myInner = myOuter.new InnerClass();

        System.out.println(myInner.y + myOuter.x);

    }

}

❯ javac Main.java

Main.java:12: error: OuterClass.InnerClass has private access in OuterClass

        OuterClass.InnerClass myInner = myOuter.new InnerClass();

Example of a static inner class:

class OuterClass {

    int x = 10;

    static class InnerClass {

      int y = 5;

    }

}

public class Main {

    public static void main(String[] args) {

        OuterClass.InnerClass myInner = new OuterClass.InnerClass();

        System.out.println(myInner.y);

    }

}

// Outputs

5

**01**

Use the code skeleton in the e01 directory.

1. Create a new class named Person with a private String name property and constructor for setting the name of the person.
2. Create a private inner class Address with a private constructor that has the parameters String street, String city, String zip which set private properties of the same names.
3. Create a private inner class PhoneNumber with a private constructor that has the String countryCode, String number parameters which set private properties of the same names.
4. In the Person class:
   1. Add a private property Address address.
   2. Add a setAddress method with String street, String city, String zip parameters that will create an Address object with the Address constructor and assign it to the person address property
   3. Add a private property PhoneNumber phoneNumber.
   4. Add a setPhoneNumber method with String countryCode, String number parameters that will create an PhoneNumber object with the PhoneNumber constructor and assign it to the person  phoneNumber property.
5. Create a new class named Main with a main method that demonstrates how the Person class, Address inner class, and PhoneNumber inner class can be used by creating a Person object and calling the setAddress and setPhoneNumber methods with the correct arguments.
6. Modify the Person class to include a method named display that displays the person's name, address, and phone number.
7. Modify the Main class to call the dislpay method and run the application.

//Expected output

Name: John Doe

Address: Valtakatu 5, Tampere, 32500

Phone: (358) 050 444 5555

Name: Jane Doe

Address: Koulukatu 7, Tampere, 32500

Phone: (358) 044 123 4467

**02**

Use the code skeleton in the e02 directory.

1. Create a new class named Rectangle that represents a rectangle. The class should have the following properties, private int width and private int height. The class should also have a constructor which has the parameters int width and int height which sets the values of the properties.
2. Implement a method called getArea() which return the area of the rectangle.
3. Create a new public inner class named RectangleDimensions inside the Rectangle class. The RectangleDimensions class should have private width and height properties and a getPerimeter() method that returns an int.
4. Create a new class named Main with a main method. Inside the main method create Rectangle object and call the getArea method.  Inside the main method also create a RectangleDimensions object and call the getPerimeter method.
5. Run the Main class and observe the output. The output should display the area of the rectangle and the perimeter of the rectangle.

// Expected output

Area: 200

Perimeter: 60

**03**

Use the code skeleton in the e03 directory.

1. Create a new class named Car that represents a car. The class should have the following properties, private String make and private String model. The class should also have a constructor which has the parameters String make and String model which sets the values of the properties.
2. Implement the methods getMake and getModel to the Car class.
3. Create a new inner public class named Engine inside the Car class. The Engine class should have a private double displacement property. The class should also have a constructor which has the parameters double displacement which sets the value of the property.
4. Implement a getHorsepower method that returns the horse power of the engine as a double. The horsepower is the displacement of the engine times 100.
5. Create a new class named Main with a main method. Inside the main method create Car object. Inside the main method also create an Engine object.
6. Run the Main class and create the following output. The output should display the make and model of the car and the horsepower of the engine.

// Expected output

Car: Toyota Camry

Engine: 250.0 horsepower

**04**

Use the code skeleton in the e04 directory.

1. Create an interface named Greeting with a single method named sayHello.
2. Create a new class named Main with a main method that demonstrates how anonymous inner classes can be used to implement the Greeting interface.
3. Run the Main class and observe the output. The output should display the message "Hello, anonymous inner class!".

// Expected output

Hello, anonymous inner class!

**05**

Use the code skeleton in the e05 directory.

1. Create a new class named Person that represents a person. The class should have a private property int age. The class should also have a constructor which has the parameter int age which sets the value of the property.
2. Implement a getAge method that returns the age of the person.
3. Create an interface named AgeFilter with a single method named accept. The accept method should take a Person object as an argument and return a boolean.
4. Create a new class named Main with a main method that demonstrates how anonymous inner classes can be used to implement the AgeFilter interface.
5. Create a list of 5 Person objects. Filter for example all the people equal or above age 40

* light bulbHint: Look at Java ArrayLists.

List<Person> people = new ArrayList<>();

people.add(new Person(21));

people.add(new Person(30));

people.add(new Person(40));

people.add(new Person(50));

1. Run the Main class and observe the output. The output should display the ages of the people who are 40 years old or older.

// Expected output

Filtered People:

40

50

**06**

Use the code skeleton in the e06 directory.

1. Create a new class named Main with a main method.
2. Create an array of integers and initialise it with a list of numbers from 1 to 5.
3. Create a Runnable object that prints the numbers in the array when it is run.

// Expected output

1

2

3

4

5

light bulbHint: [Lambda Runnable examples](https://www.codejava.net/java-core/the-java-language/java-8-lambda-runnable-example)