**Workshop**

Topics:

1. OOP
2. Immutable Objects
3. Object Lifecycle & Garbage Collector
4. Exceptions

Prerequisites:

1. IntelliJ IDEA
2. Java SDK 7

OOP Exercises:

Create a new java project with IDEA and JDK 7.

1. In the new project create a ro.teamnet.zerotohero.oop.graphicshape package.
2. Create a new class named Shape with a public area method, that returns a double value, and one primitive data field of type int, named color, and one primitive of type float, named saturation. Give the area method a default implementation. For color and saturation create setter methods.
3. Place the Shape class, in ro.teamnet.zerotohero.oop.graphicshape package.
4. Make the color and saturation fields, of the class Shape, protected (we will test the protected access modifiers property).
5. Create a new class Circle, in ro.teamnet.zerotohero.oop.graphicshape package. In this class create three private data fields of type int: xPos, yPos and radius, and a constructor with no parameters in which the three primitives are initializes with values (of you choice).
6. Create a new interface, named ShapeBehaviour, in ro.teamnet.zerotohero.oop.graphicshape package, with a area method that returns a double value.
7. Now let’s use the interface. Make the class Shape implement ShapeBehaviour.
8. Create a abstract class AbstractShape, in ro.teamnet.zerotohero.oop.graphicshape package, with a area method that returns a double value.
9. As with interfaces we can use abstract classes by extending them and inheriting their properties and functionalities. Make the Shape class extend AbstractShape.
10. Inheritance can be used by extending a concrete class as well. Make Circle class extend Shape class and implement the area method so that it calculates the circle area. Import java.lang.Math.PI with static import for that calculation. Overwrite the area method so that it returns the area of the circle.
11. Create a new class Square that extends the Shape class with a private int primitive named side and a constructor that takes one parameter which initializes the side variable. Overwrite the area method so that it calculates the square area.
12. Create a new class named Circles with a method getAreaPub with double return type. In this method create a new Circle object and return its area (call to public method area(), within package).
13. Create a runapp package in ro.teamnet.zerotohero.oop. In this package create a new class named RunApp with a main method. In this method create a new Circles object and print the default circle area by caling getAreaPub of the Circles object (The message printed should be something like ‘The default circle area is 2827.4333882308138’).
14. Overload the Circle class constructor by creating three more constructors with one, two, and three parameters.
15. Every class extends the Object class which has a default toString method. Overwrite the toString method in the Cirlce class in order to return the xPos, yPos and radius in the following format: “center = (x,y) and radius = z”. The toString method prototype is: public String toString();
16. In the Circle class create an overloaded method named fillColour. One method should have no parameters, one with an int parameter and one with a float parameter. The fillColour method with no parameters should print the super classes color primitive variable. The fillColour method with an int parameter should set the super classes color primitive variable and print a message after. The message should be like "The circle color is now 2". The fillColour method with a float parameter should set the superClass saturation parameter.
17. In class Circles create a method named getAreaDef with void return type. In this method create a new Circle object and call the three fillColour methods.
18. To use the getAreaDef call it in the main method of the RunApp class.
19. Create a canvas package in in ro.teamnet.zerotohero and in int create a Canvas class. In this class create a method getArea that return the default Circle area. The getArea method should have the default access modifier. From the main method of the RunApp class create a new Canvas object and try to call the getArea method. Observe that the compiler won’t let you do so.
20. Now lets test the runtime polymorphism. In the main method of the RunApp class create a new Shape object and initialize it with a new Circle(10) object. Print the new Shape object area. Create a ShapeBehaviour object by instantiating it with a new Square(10) object. Print the new Square object area.
21. Let’s dive deeper into overwriting. Create a new Point class in

ro.teamnet.zerotohero.oop.graphicshape package. This method should have two primitives int variables xPos and yPos. Create a constructor, for the Point class, with two parameters, that initializes the two data variables. Another method inherited from the Object class in equals. Let’s overwrite it by creating a new method equals like so :

// override the equals method to perform "deep" comparison of two Point objects

@Override

public boolean equals(Object other) {

if(other == null)

return false;

// check if the dynamic type of 'other' is Point

// if 'other' is of any other type than 'Point', the two objects cannot be

// equal if 'other' is of type Point (or one of its derived classes), then

// downcast the object to Point type and then compare members for equality

if(other instanceof Point) {

Point anotherPoint = (Point) other;

// two points are equal only if their x and y positions are equal

if( (xPos == anotherPoint.xPos) && (yPos == anotherPoint.yPos) )

return true;

}

return false;

}

Create, in the main method of the RunApp class, three objects and instantiate them with Point values :

Object p1 = new Point(10, 20);

Object p2 = new Point(50, 100);

Object p3 = new Point(10, 20);

Now if you type:

System.out.println("p1 equals p2 is " + p1.equals(p2));

System.out.println("p1 equals p3 is " + p1.equals(p3));

This code will print :

p1 equals p2 is false

p1 equals p3 is true

1. In inheritance the subclass can call the super classes constructor by using the super keyword. Create a new class, in ro.teamnet.zerotohero.oop.graphicshape, named Point3D, that extends Point class, with a private int zPos and a constructor with three parameters. The first two parameters will be used in the call of the super classes constructor like so super(x, y) and the third will be used to set the zPos;

Immutable Object Exercises:

1. Create a new immutable class and test its immutability.

Object Lifecycle & Garbage Collector

1. Create new package: ro.teamnet.zerotohero.gc
2. Create a new class GCExample, with main method:

public static void main(String[] args) {

while(true) {

new DemoObject();

}

}

1. Create a nre class DemoObject:

public class DemoObject {

private static final int bufferSize = 100000;

private String temp;

private String objectRef;

private static int count = 0;

public DemoObject() {

this.objectRef = this.toString();

}

}

1. In the default constructor, generate a large String object, and set this value into temp variable.
2. Update variable count (instances of DemoObject).
3. Open your application in JvisualVM.
4. Force GC to run after more than 5 instances of DemoObject are created.
5. For your application, set VM options: -Xmx6M -Xms2M
6. In GCExample class, modify the main method:

public static void main(String[] args) {

List<DemoObject> demoObjects = new ArrayList<DemoObject>();

while(true) {

demoObjects.add(new DemoObject());

}

}

Open your application in JvisualVM. What's happening?

Exception handling:

1. Define a new exception.
2. Create a new example of throwing an exception by using you newly defined exception.
3. Create an example of exception propagation.
4. Create an example of nested exceptions in which the newly thrown exception.
5. Create an example in which you make use of try/catch, try/catch with resources, try/catch/finally, try/finally and multi-catch.