Department of Informatics

Professor: R. Ingold
Teaching Assistants: M. Alberti, L. Vögtlin
Assistant: R. Geissbühler

S2019 Object Oriented Programming

Assignment 02

1 Theory

In this exercise you have to answer the following theoretical questions. Keep your answers short and precise.

(a) What is the difference between `int` and `Integer`?

(b) Objects may have multiple references to them at a given time. Make an example where this would be beneficial and one in which it would not. Describe the use case with words, not code

(c) What is the operator == doing ? Why must it not be used to compare two strings?

(d) What is the proper way to deal with comparison with `null` and `NaN`?

(e) When writing the `equals()` method, which other method or methods need to be overridden? Why?

(f) What can go wrong if one forgets to override the `equals()` method in a class?

(g) The keyword `this` is often used to reference the object's attributes/methods when their names are shadowed in the same scope i.e. in the constructors. This is useful, but not necessary as we could just choose the names differently. In which case the use of `this` is mandatory and could not be done by any other means?

(h) What is the difference between a `static` attribute and a non-`static` one ?

(i) Why is not possible to reference non static fields from a static method ? What would be the problem if that would be possible?

(j) Factory methods:

1. What is called a factory method?
2. What is their main advantage?
3. Why would one use them instead of a regular constructor?
4. How many of them can there be in a class?
5. What modifications should be done to the class constructors when using them?

2 Implementation

For this exercise you are required to submit two files: Person.java and Main.java, implemented as described below.

(a) Implement a class `Person` with private fields for first name, last name and age. Add factory methods to allow the creation of a person with either first name and age or last name and age. If one tries to create a person which has already been created (same first and last name) print an error message on the console. Important: this has to be implemented in `Person` and not in `Main`. Define constructor and other methods of the class as necessary.

(b) Implement a `Main` class with a `main()` method which creates multiple `Person` objects and exhaustively test the functionalities of class `Person` e.g. verify that the error message is printed on creation of the same person.

3 Debugging

In this exercise we want to practice the use of the debugger and familiarize with the error messages of the compiler. Refer to the files `Main.java` and `Car.java` attached to this series for this exercise.

(a) Report the *exact* compiler warning message at line 6 of Main.java. Explain what is the issue and how to fix it.

(b) Report the *exact* compiler error message and explain what is the problem if you were to run `Main()`

(c) Modify the code to fix the problem. There are multiple solutions, pick any.

(d) Now that the code runs without error messages, explain why the string "They are the same car!" is not being printed. What is the mistake?

(e) Modify the code to fix the problem, such that the string "They are the same car!" is printed on execution of `Main()`.

(f) Set a breakpoint at line 37: "`return returnValue;`" and report the value of the variable `returnValue`. *Hint:* don't cheat your way out by printing it if you want to learn something.

For reference (you don't have to rewrite them yourself, we provide the java files attached to the series):

```java
package core;

public class Main {
    public static void main(String[] args) {
        Car car1 = new Car("MyCar");
        System.out.println(car1.NUMBER_OF_WHEELS);

        Car car2 = new Car(new String("MyCar"));

        if(car1.equals(car2)){
            System.out.println("They are the same car!");
        }

        car1.hashCode();
    }
}
```

```java
package core;

public class Car {

    public static int NUMBER_OF_WHEELS = 4;
    private String model;

    public Car(String model){
        this.model = model;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public static void printModel(){
        System.out.println(this.model);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Car car = (Car) o;

        return model != null ? model == car.model : car.model ==
            null;
    }

    @Override
    public int hashCode() {
        int returnValue = model != null ? model.hashCode() : 0;
        return returnValue;
    }
}
```

4 Bonus exercise

This is a bonus exercises. Provided you solve it properly, it can recover previous assignment shall you have failed it. Otherwise it would contribute to have a pass in this assignment.

In this exercise you have to answer the following theoretical questions. Keep your answers short and precise.

(a) Can there be multiple constructors in a class? What is the limitation?

(b) When is it necessary to replace the default constructor ? *Hint: visibility (but not only)*

(c) What is the privacy level of an attribute/method if no keywords are specified ? What is the major difference between that level and `protected` ?

(d) Why is encapsulation important? Make a concrete example where without encapsulation there would be issues.

(e) Can one in Java be really sure that one object is being destroyed? Why ? Is the Java garbage collector deterministic?

(f) What happens if there is a circular reference e.g. object A refers to object B and vice versa, but nothing else in the running program refers to either A or B. Would these items be collected by the garbage collector or its a memory leak ?

5 Project

For this step of the project, you will have to implement and test a `SpriteList` and Factory Methods for the class `Sprite`. You can either use your implementation of `Sprite` and `Tester` from the previous series or download the solutions from moodle.

5.1 Create Factory Methods for Sprite

In this section we will apply the presented design pattern "Factory Methods". Create three Factory Methods for the class `Sprite` which each ret rus an`Sprit` object with a different image but you can define the x/y-coordinates for the `Sprite`.

Hint: Do not forget to set the `Sprite` constructor to private.

5.2 Implement an own SpriteList

You will now implement a data structure to store an arbitrary amount of `Sprites`. Create a new class called `SpriteList` and add the following methods:

add: add a item to the list

remove: remove an item at a certain position

getSize: return the size of the list

get: get an element at a certain position

To internally store your elements you can use an array and also we recommend you to keep track of the size with the help of a size field. If the array reaches its capacity create a new one and copy all the elements of the old array into the new array (hint: `copyOf()`[1]). To remove an element at position `i` concatanate the subarrays of your storage array to a new array (hint: `arraycopy()`[2]).

5.3 Writing a Test

To test your Factory Methods and the `SpriteList` create a Tester class (you can take the one from last week and adapt it). Create at least four different `Sprites` and save them into the `SpritList`. Then draw the `Sprites` from the List.

[1] https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#copyOf-boolean:A-int-

[2] https://docs.oracle.com/javase/8/docs/api/java/lang/System.html#arraycopy-java.lang.Object-int-java.lang.Object-int-int-