Department of Informatics

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

S2019 Object Oriented Programming

Assignment 05

1 Theory

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

(a) What is a package in Java?

(b) What is the benefit of using different packages as opposed to putting all the source files in the same one?

(c) Explain what is the concept of *shadowing* in general.

(d) Relate the concept of *shadowing* to the usage of different packages. *Hints: two packages with same name.*

(e) What is the default visibility of an attribute/method? Make a concrete example where this is beneficial to use.

(f) What is the practical difference between package private and protected? Make a concrete example. *Hint: you might import a package of someone else.*

(g) Using the star notation "*" for importing packages is not always a good idea. Why? Also, make an example where it is and one where it is not.

2 Implementation

A possible classification for animals is shown in Figure 1 as an UML diagram[1] where green stands for `implements` and blue stands for `extends`. When it comes to classify the platypus [2] (see figure 2) you realize that it nurses but it also laying eggs. So implement the class diagram shown in Figure 1 including the poor platypus and answer the following questions:

(a) What is the difference between the `Mammal` class and the `Oviparus` class?

(b) There are two different ways to add the platypus into this hierarchy. Describe the one you did not chose for the implementation.

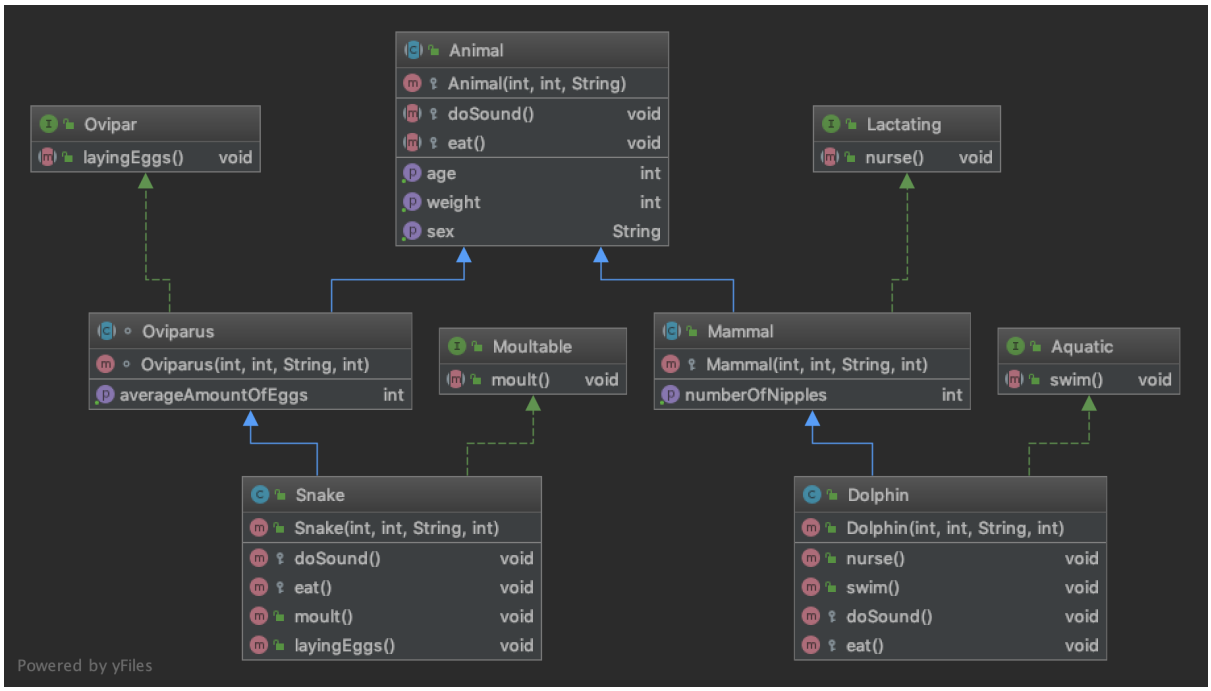

Figure 1: UML Diagram

Figure 2: A Platypus

[1] https://en.wikipedia.org/wiki/Unified_Modeling_Language
[2] https://en.wikipedia.org/wiki/Platypus

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`, `Character.java` and `PrettyPrinter.java` attached to this series for this exercise.

(a) Run the `Main` file. Notice that you have multiple compilations error: both in file `Main.java` and `PrettyPrint.java`. Report them and explain with your own words what is happening for both cases.

(b) Fix the problem in `Main.java` by modifying line 3 appropriately.

(c) Fix the problem in `PrettyPrint.java` by modifying it appropriately.

(d) Finally run the code. Observe that the two `System.out` output different values. Explain why and what is the number that gets printed by line 9.

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

```java
package core;

import java.lang.Character;
import core.PrettyPrinter;

public class Main {
    public static void main(String[] args) {
        Character character = new Character("How to Avoid Huge Ships",
           "John W. Trimmer", 35);
        System.out.println(character);
        System.out.println(PrettyPrinter.printCharacter(character));
    }
}
```

```java
package core;

public class Character {
    final String bookTitle;
    final String name;
    final int age;

    public Character(String bookTitle, String name, int age) {
        this.bookTitle = bookTitle;
        this.name = name;
        this.age = age;
    }
}
```

```java
public class PrettyPrinter {
    public static String printCharacter(Character character) {
        return "Character{" +
                "bookTitle='" + character.bookTitle + '\'' +
                ", name='" + character.name + '\'' +
                ", age=" + character.age +
                '}';
    }
}
```

4 Bonus exercise

For this exercise you are required to submit all source files for the implementation described below.

(a) Define an abstract class `Animal` with protected fields for name and the following methods:

- `printName()` which just prints the name to console.
- `abstract talk()`.
- `abstract doSomething()`.

(b) Define an interface `Ground` which exposes the method `run()`

(c) Define an interface `Flying` which exposes the method `fly()`

(d) Define an interface `Marine` which exposes the method `swim()`

(e) Implement 3 different classes, which extend from `Animal` and implements each a different interface. Implement `doSomething()` by calling the peculiar method of the interface (e.g. cat implements Ground and runs). Override the missing methods with a reasonable implementation by providing a meaningful message on console Define constructor and other methods of the class as necessary.

(f) Implement a `Main` class with a `main()` method which create some animals adds them to a list. Iterate over such list and make them talk. If you did everything correctly you will enjoy the powerfulness of polymorphism.

(g) In the `Main` class add additional code that makes every ground animal run for 10 times and then talk again. Beware: only the ground animals should be performing this action. You are NOT allowed to use `instanceof` in any way. *Hint 1: multiple collections. Hint 2: mind the type of the list!.*

5 Project

This week we want to build the package structure of our project. The starting point of our structure are the three folders `Model`, `View` and `Controller`. These folders represent the `MVC`[3] design patter which will be presented later in the course.

5.1 Planning & Implementation

Think about the structure of your packages and then apply this structure onto your project. Do not forget to start from the three base folders and build your structure from there.
Make sure that your classes, method and fields have their scope as restricted as possible.

5.2 Tester

Buld again a little tester and who that your code is still working.

[3] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller