PS Programming Methodology

University of Innsbruck - Department of Computer Science

Blaas A., Ernst M., Frankford E., Huaman Quispe E., Hupfauf B., Kaltenbrunner L., Moosleitner M., Priller S., Simsek U.

universität innsbruck
Department of Computer Science

April 4th, 2022

Exercise Sheet 3 – Inheritance, UML, Encapsulation

Exercise 1 (Visibility) [1 Point]

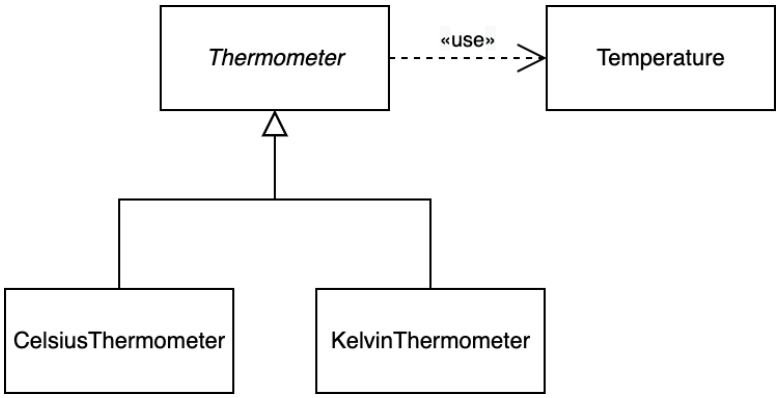
Describe in your own words what the difference between `private` and `public` variables and methods is and what they are used for. In this context, also describe the usage of `this`.

Submit

`at/ac/uibk/pm/gXX/zidUsername/s03/e01/exercise1.txt`

Exercise 2 (Thermometer) [1 Point]

Given the classes shown in the UML diagram and their implementation (`Temperature` OLAT, `Thermometer` OLAT, `CelsiusThermometer` OLAT, and `KelvinThermometer` OLAT).



Execute the program defined by `ThermometerApplication` OLAT and analyze its code and output. Record and explain any problems you notice! Also analyze the hierarchy `Thermometer` OLAT, `CelsiusThermometer` OLAT, and `KelvinThermometer` OLAT, write down any questionable design choices and explain why you regard them as questionable.

Submit

`at/ac/uibk/pm/gXX/zidUsername/s03/e02/exercise2.txt`

Exercise 3 (Polymorphism and Late Binding) [1 Point]

Java's polymorphism to a large degree capitalizes on late binding. The following code example depicts a small Java program showcasing Java's polymorphism using late binding. Try to figure out the sequence of digits printed on the screen **before** compiling and running the code - this way you will get the most out of this exercise. Explain your answers!

```java
public class Top {
    public void m(Top t) {
        System.out.print(1);
    }
    public void m(Middle m) {
        System.out.print(2);
    }
    public void m(Bottom b) {
        System.out.print(3);
    }
}
```

```java
public class Middle extends Top {
    public void m(Top t) {
        System.out.print(4);
    }
    public void m(Bottom b) {
        System.out.print(5);
    }
}
```

```java
public class Bottom extends Middle {
    public void m(Middle m) {
        System.out.print(6);
    }
}
```

```java
public class Exercise3Application {
    public static void main(String[] args) {
        Top o1 = new Middle();
        Middle o2 = new Bottom();
        o1.m(o1);
        o1.m(o2);
        o2.m(o1);
        o2.m(o2);
    }
}
```

Submit

`at/ac/uibk/pm/gXX/zidUsername/s03/e03/exercise3.txt`

Exercise 4 (UML) [2 Points]

In this exercise, you have to model a software component using a class diagram in UML. The planned software implements an elementary geometry library. The requirements for the software component are as follows:

- All shapes should support the following operations:
  - computation of the circumference
  - computation of the area
  - get and set the background color
- The software supports the following shapes:
  - Rectangle
  - Triangle
  - Circle

While working on your design, keep in mind what you learned about data encapsulation and inheritance! The submitted class diagram should have a high level of detail. It has to include at least the access modifiers, names and data types for the attributes and operations of all classes.

Hint

Online UML design tools:

- https://app.diagrams.net/
- https://apollon.ase.in.tum.de/

Submit

`at/ac/uibk/pm/gXX/zidUsername/s03/e04/exercise4.png`

Exercise 5 (Inheritance) [2 Points]

Using Java's inheritance mechanism and the UML diagram you have drawn in exercise 4, implement the required system.

a) Implement Shape.java.

b) Given Shape.java, extend your system providing the triangle, rectangle and circle classes.

c) Prepare a short demo where you calculate the circumferences and areas of triangles, rectangles and circles.

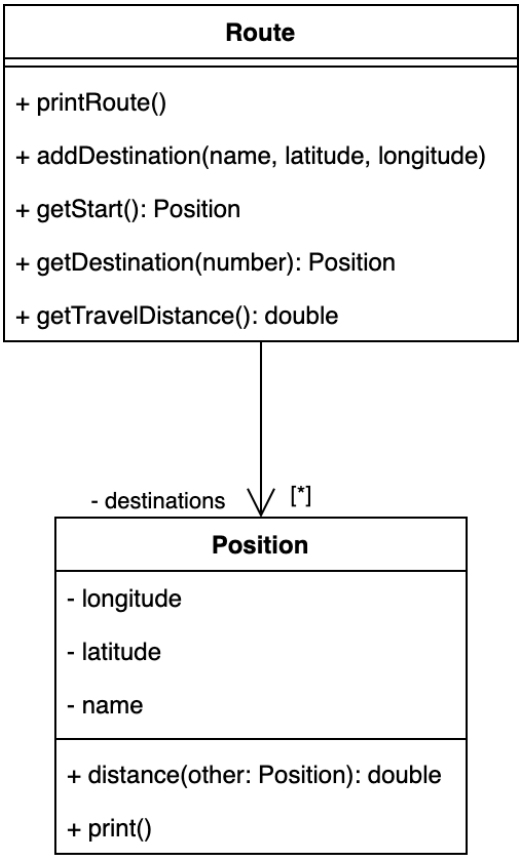
Submit

`at/ac/uibk/pm/gXX/zidUsername/s03/e05/Shape.java`

`...`

Exercise 6 (Implement an UML diagram) [3 Points]

Implement the classes Route and Position based on the following UML diagram. Choose appropriate data structures where they are not given in the diagram. Implement constructors as needed by your application The method `distance` should compute the spherical distance (beeline) in kilometers. Additionally, you have to implement a `RouteApplication.java`, where you create a route and demonstrate the implemented functionality.



You may use the following output as inspiration for your implementation of `RouteApplication.java`.

```
Start
-----
Innsbruck: longitude = 11.39454 latitude = 47.26266
    distance = 386.67km
Vienna: longitude = 16.37208 latitude = 48.20849
    distance = 523.39km
Berlin: longitude = 13.404954 latitude = 52.520008
    distance = 877.05km
Paris: longitude = 2.349014 latitude = 48.864716
    distance = 694.81km
Innsbruck: longitude = 11.39454 latitude = 47.26266
-----
End
Total distance = 2481.92km
```

Hint

- You can approximate planet earth as a sphere with a radius r of 6370km.
- The following formula may be helpful in computing the distance d between two points on a sphere:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right).$$

  where φ_1, φ_2 are the latitude of point 1 and latitude of point 2, and λ_1, λ_2 are the longitude of point 1 and longitude of point 2.
- The trigonometric functions provided by the class `Math` expect arguments in radian!

Submit

- `at/ac/uibk/pm/gXX/zidUsername/s03/e06/Route.java`
- `at/ac/uibk/pm/gXX/zidUsername/s03/e06/Position.java`
- `at/ac/uibk/pm/gXX/zidUsername/s03/e06/RouteApplication.java`
- `...`

Important: Submit your solution to OLAT and mark your solved exercises with the provided checkboxes. The deadline ends at 6:00 pm (18:00) on the day before the discussion.