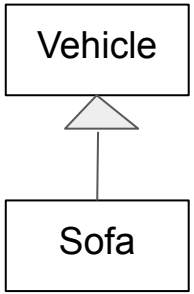
Programming Software Systems

Lab 7

Inheritance and Polymorphism



Exercises from previous lab

1) Create a class which represents Animal class and its basic properties: height, weight, color, and basic operations: eat, sleep, animalSound. Also create child classes which represent the exact animals: cow, cat, dog and override properties / methods.
2) Implement classes for different shapes: Circle, Rectangle, Triangle, Square, Ellipse. Add corresponding members and methods to calculate the area and perimeter of the shapes.

Note: Use inheritance for minimizing amount of code

Exercise to be graded.

Define classes for shapes of type Circle, Rectangle, Triangle, Square, Ellipse. First, you have to parse an input by parameters if the input consists of:

- one number (such as **1**) - the length of the radius of a Circle
- two numbers (such as **4 5**) - the length of the sides of a Rectangle or Square (**4 4**)
- three numbers (such as **2 3 4**) - the length of the sides of a Triangle, if there is no triangle with such sides, then output -1
- one symbol (e) and two numbers (such as **e 2 4**) - the length of the radiuses of an ellipse

After you have parsed the input and defined the shape, calculate its perimeter and area, and write the result into a output file.
Note: For floating results, print only one number after the point. For example, if your calculations result in a number such as **2.3852**, then output **2.3**.
Please consider the following formula for calculation the perimeter of ellipse:

$$p \approx 2\pi\sqrt{\frac{a^2 + b^2}{2}}$$

Exercise 1

- Create the abstract class *Creature* with abstract methods *bear()* and *die()* and String field *name* equal to *null* and boolean *isAlive* equal to *false*. Also, create non-abstract method *shoutName()*, which should print the name, if it's not equal to null. Otherwise, it should print error message
- Create classes *Human*, *Dog* and *Alien* which should inherit the *Creature*. Override all abstract methods for all 3 classes differently
- For *bear()* method each of them should assign the *name* and print the message "*[class name]* called *[name]* has born"
- For *die()* method each of them should print the message "*[class name]* called *[name]* has died"
- Add a method *bark()* to a class *Dog*

Exercise 1 (cont.)

- Create the *AbstractClassDemonstration* class, to demonstrate the functionality
- Modify Exercise 1 *AbstractClassDemonstration* class, so that array of creatures of different types (*Human, Dog, Alien*) is created. For each element of the array call methods *bear()* and *die().*

Hint: you can use ArrayList instead of array

Discussion

1. Creature _dog = new Dog();
2. _dog.bark();

Exercise 2

- Create an interface *Swimmable* with methods *swim()* and *stopSwimming()*
- Create an interface *Flyable* with methods *fly()* and *stopFlying()*
- Create an interface *Living* with default method *live()* that prints "*[class name]* lives"
- Create class *Submarine* which implements *Swimmable* and override methods
- Create class *Duck* which implements *Swimmable, Flyable* and *Living,* and override non-default methods
- Create class *Penguin* which implements *Swimmable* and *Living,* and override non-default methods
- Create the *InterfaceDemonstration* class, to demonstrate the functionality.

Hint: to stop swimming/flying creature has to be swimming/flying

Exercise 2 (cont.)

- Modify Exercise 2 *InterfaceDemonstration* class, so that array of living objects of different types (*Duck, Penguin*) is created. For each element of the array call method *live().*

Discussion

- What should happen if *swim()* is called for the elements of this array?
- Can instance of a *Submarine* be added to this array?

References

- Inheritance, abstract classes, interfaces https://medium.com/@isaacjumba/overview-of-inheritance-interfaces-and-abstract-classes-in-java-3fe22404baf8
- Polymorphism https://codegym.cc/groups/posts/99-how-to-use-polymorphism