Abstract Classes and Interfaces

Tobias Hanf, Maik Göken

January 16, 2023

Learn Programming with Java

Outline

Revision

Abstract Classes

Interfaces

Static Members

Exercise

Revision

Quiz

https://pingo.coactum.de/169407

Abstract Classes

Abstract function

An abstract function is a function with no concrete implementation.

- Method
- Only has declaration
- No concrete implementation

Abstract class

An abstract class is a class which has one or more abstract functions.

- At least one abstract function
- Can have
  - Attributes
  - Normal functions

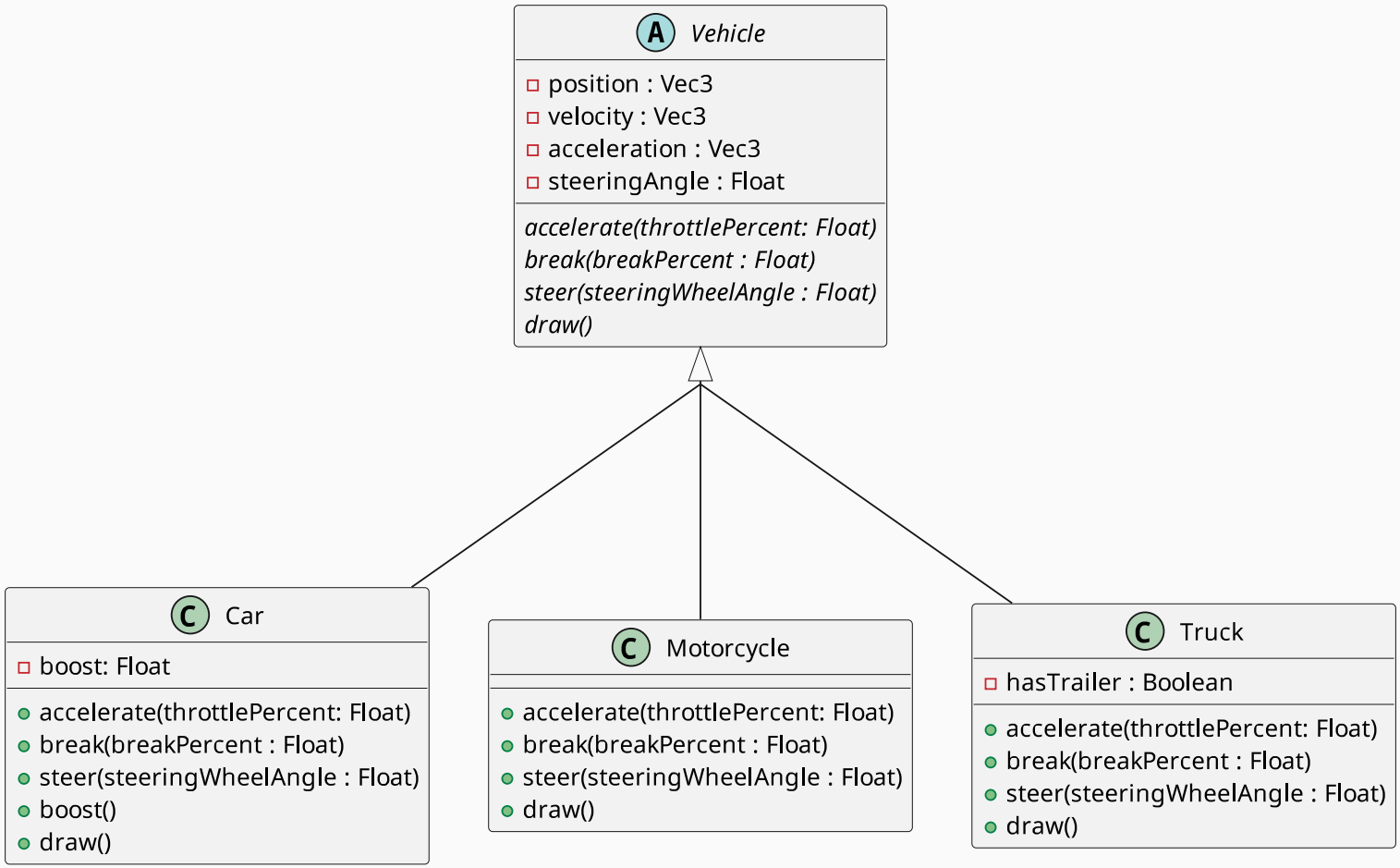
Why do we use abstract classes?

Sometime we know we need a method but we do not have a meaningful implementation for the superclass.

- Superclass has function declaration
- Subclass has concrete implementation
- ”Forces” implementation in Subclass

Example: Vehicle

We have a lot of redundant code (definitions). To circumvent this a Superclass can be introduced:



Abstract Classes in Java

For abstract functions and classes we use the `abstract` keyword.

```
1 // abstract class
2 public abstract class <class-name>  {
3
4     // abstract function
5     public abstract <ret-type> <method-name> (...);
6     ...
7     }
8 }
```

Interfaces

Interface

An interface is group of related methods with no implementation, which are realized (implemented) by other classes.

It specifies the:

- Name of the function
- Return type
- Parameter list

An interface could also be seen as an abstract class with no non-abstract functions and no attributes.

Why do we use interfaces?

Sometimes we do not care how an objects looks (what concrete class it is), we just need a specified set of functions.

Improves:

- Abstraction
- (Generalization)
- Coupling
  - very important for large Software

Interfaces in Java

- Contains only `public` functions
- Classes can implement multiple interfaces
- Interfaces can extend other interfaces (multiple)

```
1 public interface <interface-name> {
2     public <ret-type> <function-name> (...);
3 }
4
5
6 public class <class-name> implements <interface-name> {
7     ...
8 }
```

Example

We want to model a shipment system in which we shall process different kinds of items (eg. Postcards, Packets, Bulky Goods, ...). One requirement is to track the location of each item. Because how an item is tracked depends heavily on the type of the item, we introduce an interface `Trackable`.

```
1 public interface Trackable {
2     public Location getLastKnownLocation();
3
4     public List<Location> getLocationTrace();
5 }
```

Example

```
1  public class Packet implements Tackable {
2      ...;
3
4      @Override
5      public Location getLastKnownLocation() {
6      ...;
7      }
8
9      @Override
10     public List<Location> getLocationTrace() {
11     ...;
12     }
13 }
```

Example

We can now do the following:

```
1  public static void main(String[] args) {
2
3      public static void main(String[] args) {
4
5        Trackable[] items = new Trackable[2];
6
7        items[0] = new Postcard();
8        items[1] = new Packet();
9
10       for(int i = 0; i < items.length; i++) {
11       items[i].getLastKnownLocation();
12       }
13   }
14 }
```

Static Members

Static Variable

- Also known as Class Variables
- Only one variable per Class
- Same Variable can be accessed from every object of the class
- Or from outside the class via the class name

```
1 Test.counter;
```

Static Functions

- Also known as Class Methods
- One function for all objects of a class
- Can modify static variables
- But cannot access instance variables or methods directly
- Can also be accessed via the class name

```
1 Test.getCounter();
```

Static in Java

```
1 class Test {
2     public static int counter;
3
4     public static int getCounter() {
5     return counter;
6     }
7 }
```

Another useful keyword is `final` which disallows the modification of a variable.

The combination of a `static` and `final` is often used for constants.

```
1 public static final PI = 3;
```

Exercise

URPv3

In the next version of the University Resource Planer software we want to track all the things a student has received a grade on. This could be an exam, lab or his/her thesis. Depending on the graded object, the grading process and data needed for it could be very different. But we want a simple way of getting the grade disregarding what type of object we look at.

Every student should have a list of graded items he/she received. They should be used to calculate the average grade of the student and the old grade tracking system should be removed.

Discuss the modeling

Discuss how the new requirements could be modeled.

Hint: Apply the newly learned concepts

Implement the model

Create a copy your current URP project and implement the new model.