Inheritance & Interfaces in Java

Guillaume Muller
based on work from:
Ch. Gravier, F. Laforest, J. Subercaze

Télécom Saint-Étienne

{pénom.nom}@univ-st-etienne.fr

14 September 2020

Plan

télécom saint-étienne
école d'ingénieurs nouvelles technologies

Declare a Class

Listing 1: Person.java

```
1   public class Person {
2     // Attributes
3     private String lastName;
4     private String firstName;
5     // Methods (Constructor)
6     public Person (String n, String p) {
7       lastName=n;
8       firstName=p;
9     }
10    public void display () {
11      System.out. println (" First Name: "+firstName+", Last Name:
            "+lastName);
12    }
13    public String getName() {
14      return name;
15    }
16  }
```

Inheritance – 1

Listing 2: Student.java

```
1   public class Student extends Person {
2     // Attributes
3     private int year;
4     // Methods (Constructor)
5     public Student(String n, String p, int a) {
6       super(n,p);
7       year=a;
8     }
9     @Override
10    public void display () {
11      System.out. println ("Name: "+firstName+" "+lastName+" ("+year+")");
12    }
13  }
```

Inheritance – 2

- Problem:
  Parent & Child classes are **tightly coupled** ⇒ low reusability

- To prevent too much coupling, Java:
  - Does **not** allow multiple inheritance,
  - Proposes the concept of `interfaces`.

Single Inheritance

- In Java it is **NOT** possible to inherit from multiple classes
  ~~`public class MyClass extends Class1, Class2`~~
  ~~`public class MyClass extends Class1, extends Class2`~~
- Solves many problems at compilation time ⇒ speed
- Does not reduces expressivity

- ⇒ `interfaces`

Plan

télécom saint-étienne
école d'ingénieurs nouvelles technologies

Interfaces

Interfaces in Java

An interface defines a behaviour (method prototypes) that must be implemented by classes that realize it.

télécom saint-étienne
école d'ingénieurs nouvelles technologies

Interfaces

- **Interfaces** list **methods** common to all the classes that will implementing it;
- Inheritance is reserved for when necessary (ex: a Student **is a** Person);
- Interfaces are **contracts** that the implementing classes must follow;
- Interface do **NOT** detail the **implementation** of these methods (i.e. all methods are abstract!)[1];
- When a class implements an interface, one can be **certain** that the methods declared in the interface are present.

[1] This changed in Java 8!

Interface: Example – 1[2]

Listing 3: Bicycle.java

```
1   interface Bicycle {
2     // Interfaces do not have attributes
3
4     void changeCadence(int newValue); // Wheel revolutions per minute
5
6     void changeGear(int newValue);
7
8     void speedUp(int increment);
9
10    void applyBrakes(int decrement);
11  }
```

[2] http://docs.oracle.com/javase/tutorial/java/concepts/interface.html

télécom saint-étienne
école d'ingénieurs nouvelles technologies

Interface: Example – 2

Listing 4: ACMEBicycle.java

```
1   class ACMEBicycle implements Bicycle {
2     int cadence = 0; // Class has attributes
3     int speed = 0;
4     int gear = 1;
5     void changeCadence(int newValue) { // Methods ARE implemented
6       cadence = newValue;
7     }
8     void changeGear(int newValue) {
9       gear = newValue;
10    }
11    void speedUp(int increment) {
12      speed = speed + increment;
13    }
14     ...
15  }
```