Java

Inheritance

Tobias Hanf, Manik Khurana

15. November 2021

Java-Course

Overview

Arrays

Array

An array is a data-type that can hold a **fixed number** of elements. An Element can be any simple data-type or object.

```
    public static void main(String[] args) {

        int[] intArray = new int[10];
        intArray[8] = 7; // assign 7 to the 9th element
        intArray[9] = 8; // assign 8 to the last element

        System.out.println(intArray[8]); // prints: 7
    }

```

You can access every element via an index. A n-element array has indexes from 0 to (n-1).

Array Initialization

You can initialize an array with a set of elements.

```
    public static void main(String[] args) {

        int[] intArray = {3, 2, 7};

        System.out.println(intArray[0]); // prints: 3
        System.out.println(intArray[1]); // prints: 2
        System.out.println(intArray[2]); // prints: 7
    }

```

Alternative Declaration

There two possible positions for the square brackets.

```java
    public static void main(String[] args) {

        // version 1
        int[] intArray1 = new int[10];

        // version 2
        int intArray2[] = new int[10];
    }

```

2-Dimensional Array

Arrays work with more than one dimension. An m-dimensional array has m indexes for one element.

```java
    public static void main(String[] args) {

        // an array with 100 elements
        int[][] intArray = new int[10][10];

        intArray[0][0] = 0;
        intArray[0][9] = 9;
        intArray[9][9] = 99;
    }

```

Assignment with Loops

Loops are often used to assign elements in arrays.

```java
    public static void main(String[] args) {

        int[][] intArray = new int[10][10];

        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++) {
                intArray[i][j] = i*10 + j;
            }
        }
    }

```

Arrays with objects

Loops are often used to assign elements in arrays.

```
    public static void main(String[] args) {

        Student[][] studentArray = new Student[10][10];

        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++) {
                intArray[i][j] = new Student();
            }
        }
    }

```

Inheritance

Example

A school has students and teacher.
Every student has the following attributes:

- name
- age
- email
- lessons
- Name of his class

Every teacher has the following attributes:

- name
- age
- email
- subjects
- salary
- hours

Creating Inheritance

They have the same subset of attributes:

- name
- age
- email

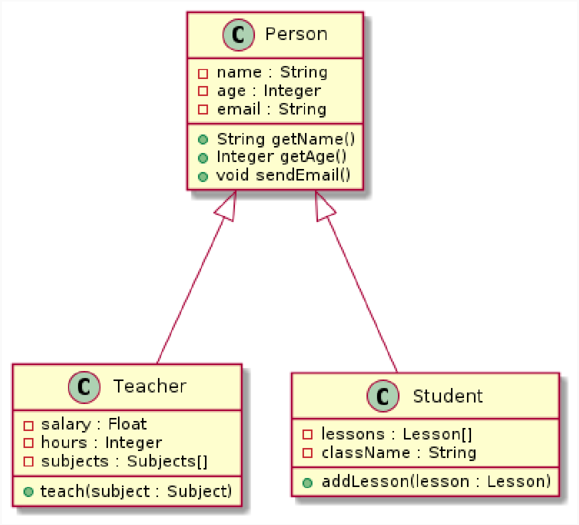
Why not define these attributes somewhere else and say these are part of a student and teacher?

What is inheritance?

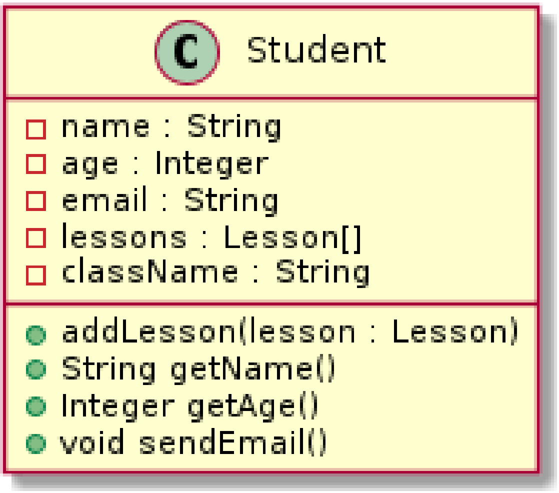
This is inheritance.

- basing an object or class upon another object or class
- creates an hierarchy of classes
- classes on top of the hierarchy are the **superclasses**
- classes below a **superclass** are called **subclasses**
- objects create from **superclasses** are called **parent objects**
- objects create from **subclasses** are called **child objects**

UML class diagram



Student class



A special Person

Our class *Student* is a kind of *Person* denoted by the keyword **extends**.

- *Student* is a **subclass** of the class *Person*
- *Person* is the **superclass** of the class *Student*

```java
public class Student extends Person {

}

```

As mentioned implicitly above a class can has multiple subclasses. But a class can only inherit directly from one superclass.

Example

We have the classes: *School*, *Person* and *Student*. They will be used for every example in this section and they will grow over time.

```
    public class Person {

        private String name;
        private int age;
        private String email


        public void setEmail(String email) {
            this.email = email;
        }

        public void sendEmail(String message){
            System.out.println("To: " this.email
                                        + " \n" + message);
        }
    }

```

Inherited Methods

The class *Student* also inherits all methods from the superclass *Person*.

```java
public class School {

    public static void main(String[] args) {

        Student Student = new Student();

        Student.setEmail("john@school.org");

        Student.sendEmail("Hello");

        // prints:  To: john@school.org
        //          Hello
    }
}

```

Override Methods

The method sendEmail() is now additional definded in *Student*.

```java
public class Student extends Person {

    @Override
    public void sendEmail(String message){
        System.out.println("To: " this.email + " \n" +
                        "Dear student :" + message);
    }
}

```

`@Override` is an annotation. It helps the programer to identify overwritten methods. It is not neccessary for running the code but improves readability. What annotations else can do we discuss in a future lesson.

Override Methods

Now the method `sendEmail()` defined in *Student* will be used instead of the method defined in the superclass *Person*.

```java
    public class School {

        public static void main(String[] args) {

            Student Student = new Student();

            Student.setEmail("john@school.org");

            Student.sendEmail("Hello");
            // prints:  To: john@school.org
            //          Dear student: Hello
        }
    }

```

Super()

If we define a **constructor with arguments** in *Person* we have to define a constructor with the same list of arguments in every subclass.

```java
    public class Person {

        private String name;
        private int age;
        private String email;

        public Person(String name, int age, String email) {
            this.name = name;
            this.age = age;
            this.email = email;
        }

        public void sendEmail(String message){
            System.out.println("To: " this.email
            + " \n" + message);
        }
    }

```

Super()

For the constructor in the subclass *Student* we can use `super()` to call the constructor from the superclass.

```java
public class Student extends Person {

    public Student(String name, int age, String email) {
        super(name, age, email);
    }

    @Override
    public void sendEmail(String message){
        System.out.println("To: " this.email + " \n" +
        "Dear student :" + message);
    }
}

```

Super() - Test

```java
public class School {

    public static void main(String[] args) {
        Student Student =
            new Student("John", 16, "john@school.org");

        Student.sendEmail("Hello");
        // prints:  To: john@school.org
        //          Dear student: Hello
    }
}

```

Object

Every class is a subclass from the class *Object*. Therefore every class inherits methods from *Object*.

See `http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html` for a full reference of the class *Object*.

toString()

Student is a subclass of *Object*. Therefore *Student* inherits the method `toString()` from *Object*.
`System.out.println(argument)` will call `argument.toString()` to receive a printable String.

```java
public class School {

    public static void main(String[] args) {
        Student Student =
            new Student("John", 16, "john@school.org");

        System.out.println(Student);
        // prints: Student@_some_HEX-value_
        // for example: Student@4536ad4d
    }
}

```

Override toString()

```java
public class Student extends Person {

    public Student(String name, int age, String email) {
        super(name, age, email);
    }

    @Override
    public String toString() {
        return "Student: " +  this.name;
    }
}

```

Override toString() - Test

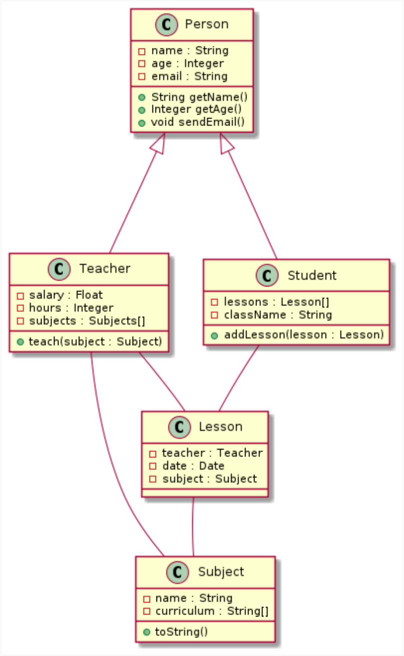
```java
public class School {

    public static void main(String[] args) {
        Student Student =
            new Student("John", 16, "john@school.org");

        System.out.println(Student);
        // Student: John
    }
}

```

Programming


Person
name : String
age : Integer
email : String
String getName()
Integer getAge()
void sendEmail()
Teacher
salary : Float
hours : Integer
subjects : Subjects[]
teach(subject : Subject)
Student
lessons : Lesson[]
className : String
addLesson(lesson : Lesson)
Lesson
teacher : Teacher
date : Date
subject : Subject
Subject
name : String
curriculum : String[]
toString()