Lab Exercise 8

Chapter 9 Objects and Classes, part 1

COMP217 Java Programming
Spring 2019
Instructor: Gil-Jin Jang

Text: Liang, Introduction to Java Programming, Tenth Edition
Chapter 9

Exercise 8-1 Multiple Classes

- Problem: there are more than one classes
  1. Putting them in a single file
     - Only a single class can be “public” and can have “main” method
     - The name of “.java” file should match the “public” class name
     - The other class(es) are accessible one another, without “public” modifier
  2. Splitting them in multiple files (one “.java” file per class)
     - If the files are in the same folder, java automatically compiles all required “.java” files
     - To be accessible from outside the file, the class should have “public” modifier

- Practice multiple classes with a single and multiple files

Exercise 8-1 Multiple Classes

- The BMI (body mass index) and UseBMI classes

BMI	
name: String age: int weight: double height: double	The name of the person The age of the person The weight of the person The height of the person
+BMI(newName: String, newAge: int, newWeight: double, newHeight: double) +getBMI(): double +getStatus(): String	Creates a BMI object with the specified name, age, weight, and height Calculates and returns the BMI Returns the BMI status (normal/overweight)

UseBMI	
	No field
+main(args: String[]): void	public: + static: underlined void: return type

Ex8-1 Single file for BMI1 and UseBMI1

- File: UseBMI1.java
  - Because the public class name is **UseBMI1**, the file name should be **UseBMI1.java**

```java
// when there are more than two classes in a single file
// only one class can be public,
// indicating being accessible from outside
// method main() should be accessible from outside,
// so usually the public class has it

public class UseBMI1 {
  public static void main(String[] args) {
    BMI1 b1 = new BMI1("John Doe", 18, 145, 70);
    System.out.println("The BMI for "
    + b1.name + " is "
    + b1.getBMI() + " " + b1.getStatus());

    BMI1 b2 = new BMI1("Peter King", 20, 215, 70);
    System.out.println("The BMI for "
      + b2.name + " is "
      + b2.getBMI() + " " + b2.getStatus());
  }
}
```

```java
// not a public class, inaccessbile from outside
class BMI1 {
  String name;
  int age;
  double weight; // in pounds
  double height; // in inches

  static final double KILOS_PER_POUND = 0.45359237;
  static final double METERS_PER_INCH = 0.0254;

  BMI1(String newName, int newAge, double newWeight, double newHeight) {
    name = newName; age = newAge;
    weight = newWeight; height = newHeight;
  }

  double getBMI() {
    double weightInKilograms = weight * KILOS_PER_POUND;
    double heightInMeters = height * METERS_PER_INCH;
    double bmi = weightInKilograms / (heightInMeters * heightInMeters);
    return Math.round(bmi * 100) / 100.0;
  }

  String getStatus() {
    double bmi = getBMI();
    if (bmi < 18.5) return "Underweight";
    else if (bmi < 25) return "Normal";
    else if (bmi < 30) return "Overweight";
    else return "Obese";
  }
}
```

Ex8-1 Two files: BMI2.java and UseBMI2.java

BMI2.java

```
// public modifier: to be used outside
public class BMI2 {
  String name;
  int age;
  double weight; // in pounds
  double height; // in inches

  static final double KILOS_PER_POUND = 0.45359237;
  static final double METERS_PER_INCH = 0.0254;

  BMI2(String newName, int newAge, double newWeight, double newHeight) {
    name = newName; age = newAge;
    weight = newWeight; height = newHeight;
  }

  double getBMI() {
    double weightInKilograms = weight * KILOS_PER_POUND;
    double heightInMeters = height * METERS_PER_INCH;
    double bmi = weightInKilograms / (heightInMeters * heightInMeters);
    return Math.round(bmi * 100) / 100.0;
  }

  String getStatus() {
    double bmi = getBMI();
    if (bmi < 18.5) return "Underweight";
    else if (bmi < 25) return "Normal";
    else if (bmi < 30) return "Overweight";
    else return "Obese";
  }
}
```

UseBMI2.java

- BMI2.java and UseBMI2.java should be in the same folder
- Both classes BMI2 and UseBMI2 should be public

```
// UseBMI2 uses class defined in BMI2.java
public class UseBMI2 {
  public static void main(String[] args) {
    BMI2 b1 = new BMI2("John Doe", 18, 145, 70);
    System.out.println("The BMI for "
    + b1.name + " is "
    + b1.getBMI() + " " + b1.getStatus());

    BMI2 b2 = new BMI2("Peter King", 20, 215, 70);
    System.out.println("The BMI for "
      + b2.name + " is "
      + b2.getBMI() + " " + b2.getStatus());
  }
}
```

Exercise 8-1 Submission

- Write 3 files: UseBMI1.java, BMI2.java, UseBMI2.java
- Compile
  - $ javac UseBMI1.java --- will generate UseBMI1.class and BMI1.class
  - $ javac UseBMI2.java --- also generates UseBMI2.class and BMI2.class
    - Java finds all necessary classes and compiles the source if necessary
- Execute
  - $ java UseBMI1
  - $ java UseBMI2
- Note:
  - Online compilers may not support multiple files
  - Practice with offline compilers (**javac** and **java** from DOS console)
    - If necessary, install JDK
- Submission
  - 3 files: UseBMI1.java, UseBMI2.java, BMI2.java

Exercise 8-2 Class TV2

- We want to design a TV with extra functionality

UseTV2
+main(args: String[]): void

TV2	
channel: int	Channel number, between 1 and 120 (included)
volumeLevel: int	Volume level, between 1 and 7 (included)
on: boolean	If TV is on or off
previousChannel: int	Previous Channel
+TV2(ch: int, vol: int)	Creates a TV with given channel and volume.
+getState(): String	Returns a string describing the state of the TV.
+turnToggle(): void	If TV is on, turns it off. If off, turns it on.
+setChannel(int newChannel): void	Change the channel to the given channel number.
+gotoPreviousChannel(): void	Change the channel to the previous one.
+setVolume(int newVolumeLevel): void	Change the volume with the given level
+channelUp(): void	Increase the channel number by 1.
+channelDown(): void	Decrease the channel number by 1.
+volumeUp(): void	Increase the volume level by 1.
+volumeDown(): void	Decrease the volume level by 1.

Ex8-2: template

TV2.java template

```
public class TV2 {
  int channel;
  int volumeLevel;
  boolean on;
  int previousChannel;

  public String getState() {
    String out = "On:" + on;
    if ( on )
      out = out + ", Ch:" + channel + ", Vol:" + volumeLevel;
    return out;
  }
  /* complete TV2.java by implementing other methods */
```

UseTV2.java

```
public class UseTV2 {
  public static void main(String[] args) {
    TV2 tv = new TV2(60,5);
    tv.turnToggle(); tv.gotoPreviousChannel();
    System.out.println(tv.getState());
    tv.setChannel(75); tv.channelUp(); tv.channelUp();
    tv.volumeUp(); tv.gotoPreviousChannel();
    System.out.println(tv.getState());
    tv.turnToggle();
    tv.channelUp(); tv.gotoPreviousChannel();
    System.out.println(tv.getState());
    tv.turnToggle();
    tv.setVolume(2); tv.volumeDown(); tv.volumeDown();
    tv.channelDown(); tv.channelDown();
    tv.gotoPreviousChannel();
    System.out.println(tv.getState());
  }
}
```

[Output]

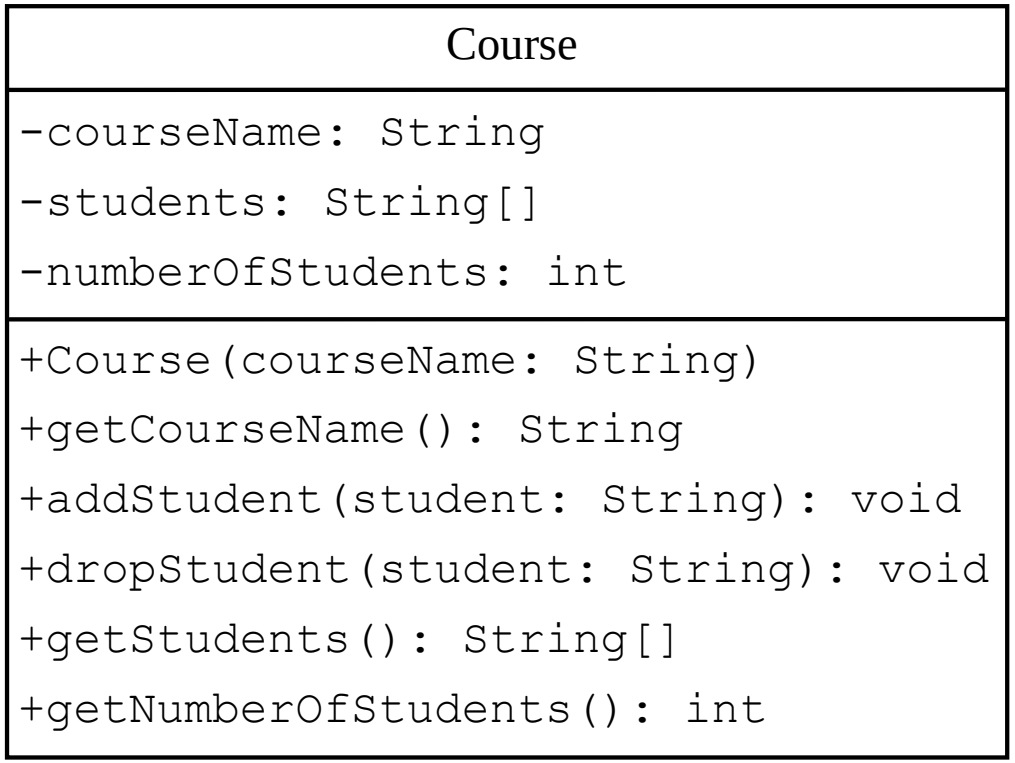
```
$ java UseTV2
On:true, Ch:60, Vol:5
On:true, Ch:76, Vol:6
On:false
On:true, Ch:75, Vol:1
```

Exercise 8-2 Submission

- Write 2 files: TV2.java, UseTV2.java
- Compile
  - $ javac UseTV2.java --- will generate UseTV2.class and TV2.class
- Execute
  - $ java UseTV2
- Submission
  - 2 files: TV2.java, UseTV2.java

Exercise 8-3 The Course Class

- Write a java code “Course.java” according to the following UML, implement all the methods.
  - Note: ignore ‘-’ sign for courseName, students, and numberOfStudents



The name of the course.

An array to store the students for the course.

The number of students (default: 0).

Creates a course with the specified name.

Returns the course name.

Adds a new student to the course.

Drops a student from the course.

Returns the students in the course.

Returns the number of students in the course.

Exercise 8-3 Submission

- In Course.java, method dropStudent():
  1. Search for the student to drop in the list (use either String.equals() or String.compareTo() method).
  2. All the elements after the dropped student should be shifted to fill the deleted element.
- UseCourse.java: write your own main method to test your implementation.
- Submission
  - 2 files: Course.java, UseCourse.java

Ex8-4

CircleWithStaticMembers Class

```
public class CircleWithStaticMembers {
  /** The radius of the circle */
  double radius;

  /** The number of the objects created */
  static int numberOfObjects = 0;

  /** Construct a circle with radius 1 */
  CircleWithStaticMembers() { radius = 1.0;  numberOfObjects++; }

  /** Construct a circle with a specified radius */
  CircleWithStaticMembers(double newRadius) { radius=newRadius; numberOfObjects++; }

  /** Return numberOfObjects */
  static int getNumberOfObjects() { return numberOfObjects; }

  /** Return the area of this circle, not involved with static */
  double getArea() { return radius * radius * Math.PI; }
}
```

Ex8-4

TestCircleWithStaticMembers

```
public class TestCircleWithStaticMembers {
  /** Main method */
  public static void main(String[] args) {
    System.out.println("Before creating objects");
    System.out.println(
      "The number of Circle objects is " +
      CircleWithStaticMembers.numberOfObjects);

    // Create c1
    CircleWithStaticMembers c1 = new CircleWithStaticMembers();

    // Display c1 BEFORE c2 is created
    System.out.println("\nAfter creating c1");
    System.out.println("c1: radius (" + c1.radius +
      ") and number of Circle objects (" + c1.numberOfObjects + ")");

    // Create c2
    CircleWithStaticMembers c2 = new CircleWithStaticMembers(5);

    // Modify c1
    c1.radius = 9;

    // Display c1 and c2 AFTER c2 was created
    System.out.println("\nAfter creating c2 and modifying c1");
    System.out.println("c1: radius (" + c1.radius + ") and number of Circle objects (" + c1.numberOfObjects + ")");
    System.out.println("c2: radius (" + c2.radius + ") and number of Circle objects (" + c2.numberOfObjects + ")");
  }
}
```

```
$ java TestCircleWithStaticMembers
Before creating objects
The number of Circle objects is 0

After creating c1
c1: radius (1.0) and number of Circle
objects (1)

After creating c2 and modifying c1
c1: radius (9.0) and number of Circle
objects (2)
c2: radius (5.0) and number of Circle
objects (2)
```

Note: static members can be accessed either by **className.member** or **instance.member** .
Example)
CircleWithStaticMembers.numberOfObjects …
c1.numberOfObjects …
c2.numberOfObjects …

Exercise 8-4 Submission

- Type-in 2 files: CircleWithStaticMembers.java, TestCircleWithStaticMembers.java
- Compile, Execute, Check the results
- Submit two files