Object Oriented Programming - Handout 3

Konrad Wojda, 9307820244

Exercise 3.1

```
// TestDo.java: Test the do-while loop
import javax.swing.JOptionPane;

public class TestDo {
  /** Main method
  public static void main(String[] args) {
    int data
    int sum = 0;

    // Keep reading data until the input is 0
    do {
      // Read the next data
      Stirng dataString = JOPTionPane.showInputDialog(null,
        "Enter an int value, \nthe program exits if the input is 0",
        "TestDo, JOptionPane.QUESTION_MASSAGE);

      data = Integer.parseInt(dataStirng);

      sum += data
    } while (data != 0};

    JOptionPane.showMessageDialog(null, "The sum is " + sum,
      "TestDo", JOptionPlane.INFORMATION_MESSAGE);

    System.exit(0);
  }
}
```

Errors:

1. Line 05 - Comment not closed.

Comments that starts with `/**` are multiline and they have to be closed with `*/`.

2. Line 07 - Missing semicolon
3. Line 13 - Typo in `String`. `Stirng` is not a type.
4. Line 13 - Typo in `JOptionPane`.
5. Line 15 - Missing closing quotation mark after `"TestDo`

6. Line 15 - `QUESTION_MASSAGE` is a typo - should be `QUESTION_MESSAGE`
7. Line 17 - Typo in variable name - `dataStirng` should be `dataString`
8. Line 19 - Missing semicolon - should be `sum += data;`
9. Line 20 - Wrong closing - should use `)` instead of `}`
10. Line 23 – Typo in `JOptionPane` - not `JOptionPlane`

Corrected code:

```
// TestDo.java: Test the do-while loop
import javax.swing.JOptionPane;

public class TestDo {
  /** Main method */
  public static void main(String[] args) {
    int data;
    int sum = 0;

    // Keep reading data until the input is 0
    do {
      // Read the next data
      String dataString = JOptionPane.showInputDialog(null,
        "Enter an int value, \nthe program exits if the input is 0",
        "TestDo", JOptionPane.QUESTION_MESSAGE);

      data = Integer.parseInt(dataString);

      sum += data;
    } while (data != 0);

    JOptionPane.showMessageDialog(null, "The sum is " + sum,
      "TestDo", JOptionPane.INFORMATION_MESSAGE);

    System.exit(0);
  }
}
```

Exercise 2.2

Code:

```
class Rectangle {
    private int x1, y1; // top-left
    private int x2, y2; // bottom-right

    public Rectangle(int x1, int y1, int x2, int y2) {
        this.x1 = Math.min(x1, x2);
        this.y1 = Math.min(y1, y2);
        this.x2 = Math.max(x1, x2);
        this.y2 = Math.max(y1, y2);
    }

    public Rectangle(Rectangle other) {
        this.x1 = other.x1;
        this.y1 = other.y1;
        this.x2 = other.x2;
        this.y2 = other.y2;
    }

    public Rectangle enclosing(Rectangle other) {
        int newX1 = Math.min(this.x1, other.x1);
        int newY1 = Math.min(this.y1, other.y1);
        int newX2 = Math.max(this.x2, other.x2);
        int newY2 = Math.max(this.y2, other.y2);
        return new Rectangle(newX1, newY1, newX2, newY2);
    }

    public void display() {
        System.out.println("Top-left: (" + x1 + ", " + y1 + "),
            Bottom-right: (" + x2 + ", " + y2 + ")");
    }
}
```

Testing class:

```
public class RectangleTest {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(1, 1, 4, 4);
        Rectangle r2 = new Rectangle(2, 3, 6, 6);
        Rectangle r3 = new Rectangle(0, 2, 3, 5);
        Rectangle r4 = new Rectangle(5, 0, 7, 3);

        Rectangle r12 = r1.enclosing(r2);
        Rectangle r123 = r12.enclosing(r3);
        Rectangle r1234 = r123.enclosing(r4);

        System.out.println("Rectangle 1:");
        r1.display();
        System.out.println("Rectangle 2:");
        r2.display();
        System.out.println("Rectangle 3:");
        r3.display();
        System.out.println("Rectangle 4:");
```

```java
        r4.display();
        System.out.println("Rectangle enclosing all (R1234):");
        r1234.display();
    }
}
```

Output:

```
Rectangle 1:
Top-left: (1, 1), Bottom-right: (4, 4)
Rectangle 2:
Top-left: (2, 3), Bottom-right: (6, 6)
Rectangle 3:
Top-left: (0, 2), Bottom-right: (3, 5)
Rectangle 4:
Top-left: (5, 0), Bottom-right: (7, 3)
Rectangle enclosing all (R1234):
Top-left: (0, 0), Bottom-right: (7, 6)
```

Exercise 2.3

Address class

```java
public class Address {
    private String street;
    public String city;
    public String zipCode;

    public Address(String street, String city, String zipCode) {
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getStreet() {
        return street;
    }

    public void printAddress() {
        System.out.println(street + ", " + city + " " + zipCode);
    }

    @Override
    public String toString() {
        return street + ", " + city + " " + zipCode;
    }
}
```

Person class

```
public class Person {
    public String firstName;
    public String lastName;
    public Address address;

    public Person(String firstName, String lastName, Address address) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
    }

    @Override
    public String toString() {
        return firstName + " " + lastName + "\n" + address.toString();
    }
}
```

Student class

```
public class Student extends Person {
    public String identificationNumber;
    public int absentTime;
    private int scoresMidtermExam;
    private int scoresFinalExam;

    public Student(String firstName, String lastName, Address address,
                   String id, int absentTime, int midterm, int finalExam) {
        super(firstName, lastName, address);
        this.identificationNumber = id;
        this.absentTime = absentTime;
        setScoresMidtermExam(midterm);
        setScoresFinalExam(finalExam);
    }

    public void setScoresMidtermExam(int score) {
        if (score >= 0 && score <= 50)
            this.scoresMidtermExam = score;
        else
            throw new IllegalArgumentException();
```

```java
    }

    public void setScoresFinalExam(int score) {
        if (score >= 0 && score <= 50)
            this.scoresFinalExam = score;
        else
            throw new IllegalArgumentException();
    }

    public int getScoresAltogether() {
        return scoresMidtermExam + scoresFinalExam;
    }

    public boolean passed() {
        return getScoresAltogether() > 60;
    }

    @Override
    public String toString() {
        return super.toString() + "\nID: " + identificationNumber +
               "\nAbsent Time: " + absentTime +
               "\nMidterm: " + scoresMidtermExam +
               "\nFinal: " + scoresFinalExam +
               "\nTotal: " + getScoresAltogether() +
               "\nPassed: " + (passed() ? "Yes" : "No");
    }
}
```