

CS203 Java Programming and Applications

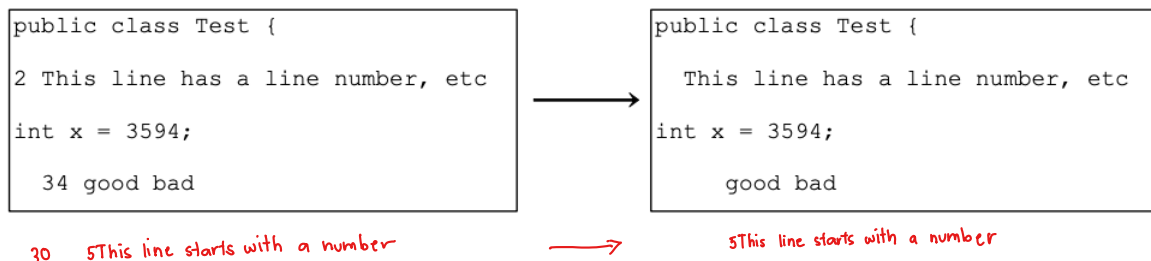
Fall 2019

Assignment 3 (6 Questions, 100 marks)

Assigned Date: November 5, 2019
Due Date: November 24, 2019 @ 23:59

QUESTION 1 (15 Marks) Remove Line Numbers in a File

Some lines in the file have a line number, as shown in the following figure. A line number is the first word in the line. A line number is delimited by one or more whitespace characters. A line number may also be preceded with one or more whitespace characters. Write a program that prompts the user to enter a file name, reads the lines from the file, and replaces the line numbers in the file with appropriate numbers of spaces. For example, you need to replace the line number 2 with one space and line number 34 with two spaces, and you should keep the two spaces preceding line number 34. Name your class file as `RemoveLineNumber.java`. You do not need to submit the .txt file that you use to test your program.



Hints:

1. You can create a .txt file with the above example to test your program. When running your test program, you should give the file path starting from the working directory. The default working directory in NetBeans is your **project** folder. Thus, you should give the file path starting from your **project** folder. Alternatively, you may put your .txt file in the **project** folder. Then you can simply use the file name when running your program.
2. To remove the line numbers, you may need to use String methods `trim()`, `matches()`, and `split()`. Check the URL <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html> for more information about these methods. Please note that `split()`

has an overloaded version with a second parameter, which might be useful in this question. And please also note that `matches()` and `split()` accept regular expressions as their parameters.

QUESTION 2 (25 Marks) Abstract Classes & Exception Handling

The `GeometricObject`, `Circle` and `Rectangle` classes are given for this question. You may need to make necessary changes to the classes other than what is mentioned below. As a summary, the files in your submission for this question should at least include: `GeometricObject.java`, `Circle.java`, `Rectangle.java`, `Triangle.java`, `TestTriangleWithException.java`, and `TestLarger.java`. Please bundle the above files and your class files in a folder named `Assignment3Question3`.

PART I: (5 Marks)

Declare the `getPerimeter()` and `getArea()` methods in the `GeometricObject` class. These two methods should be declared as abstract because they cannot be implemented in the `GeometricObject` class. Override and implement `getPerimeter()` and `getArea()` in the subclasses `Circle` and `Rectangle`. Override and implement `toString()` method in `Circle` class as:

```
return "Circle: radius = " + radius;
```

and in `Rectangle` class as:

```
return "Rectangle: width = " + width + ", height = " + height;
```

Use `@Override` annotation for all overridden methods.

PART II: (10 Marks)

Design and implement a class named `Triangle` that extends `GeometricObject`. The class contains:

- Three `double` data fields named `side1`, `side2`, and `side3` with default values `1.0` to denote three sides of a triangle.
- A no-arg constructor that creates a default triangle.
- A constructor that creates a triangle with the specified `side1`, `side2`, and `side3`.

- Override and implement the abstract methods `getPerimeter()` and `getArea()` in `GeometricObject` class.
(*Hint*: The area of a triangle is given by $\sqrt{p(p - \text{side1})(p - \text{side2})(p - \text{side3})}$ where p is half the perimeter, that is, $p = \frac{\text{side1} + \text{side2} + \text{side3}}{2}$.)
- Override and implement the `toString()` method as:

```
return "Triangle: side1 = " + side1 + ", side2 = " + side2
    + ", side3 = " + side3;
```

Use `@Override` annotation for all overridden methods.

In a triangle, the sum of any two sides is greater than the other side. The `Triangle` class must adhere to this rule. Create the `IllegalTriangleException` class and modify the constructor of the `Triangle` class to throw an `IllegalTriangleException` object if a triangle is created with sides that violate the rule. The constructor of `IllegalTriangleException` must encapsulate all three sides of the triangle and a string message, as follows:

```
public IllegalTriangleException(double side1, double side2,
                               double side3, String message)
```

Write a test program `TestTriangleWithException` to test your `Triangle` class and `IllegalTriangleException` by creating two objects of `Triangle` with one of them violating the rule. Print the perimeter and area of the legal triangle. Print the sides and string message of the illegal triangle from the `IllegalTriangleException` caught. You may need additional methods in `IllegalTriangleException` other than what is mentioned above. Format your output to two decimal places. A sample run is as follows:

```
Legal triangle:
Perimeter: 6.50
Area: 10.25

Illegal triangle:
Side1 = 1.00
Side2 = 2.00
Side3 = 3.00
The sum of any two sides is greater than the other side
BUILD SUCCESSFUL (total time: 2 seconds)
```

PART III: (10 Marks)

Implement a static method called `larger` that takes two geometric objects of the same type as arguments and returns the object with larger area. If the two objects have the same area, the method returns `null`. If the two geometric objects are not of the same type, the method throws `DifferentTypeException`. `DifferentTypeException` is a user-defined exception. Define it for your `larger` method. The method signature or `larger` is as follows:

```
static GeometricObject larger(GeometricObject g1, GeometricObject g2)
    throws DifferentTypeException
```

Write a test program called `TestLarger`. Implement `larger` method in `TestLarger` and test if your `larger` method works properly. Create different types of geometric objects to invoke the `larger` method. You should test your `larger` method with two circles, two rectangles, two triangles, and two different object types which should throw `DifferentTypeException`. Print out appropriate details about the object returned by `larger` method such as radius, width, height, sides, perimeter, and area. Format your output to two decimal places. A sample output is as follows:

```
Testing two circles of same radius:
Two objects have equal area

Testing two circles of different radius:
The returned larger object is: Circle: radius = 3.0
The area is 28.27
The perimeter is 18.85

Testing two rectangles of different sizes:
The returned larger object is: Rectangle: width = 3.0, height = 3.0
The area is 9.00
The perimeter is 12.00

Testing two triangles of different sizes:
The returned larger object is: Triangle: side1 = 2.0, side2 = 3.0, side3 = 2.3
The area is 17.59
The perimeter is 7.30

Testing two different object types:
Two objects are of different type
```

Please note that you are not allowed to overload `larger` method to accommodate `Circle`, `Rectangle` and `Triangle`. You must test whether the two geometric objects are of the same type in your `larger` method and throw `DifferentTypeException` in the case of different types.

QUESTION 3 (15 Marks) Interfaces

Design a class named `Point` that meets the following requirements:

- Two data fields `x` and `y` for representing a point with getter methods
- A no-arg constructor that constructs a point for (0,0)
- A constructor that constructs a point with the specified x and y values
- Override the `equals` method. Point p1 is said to be equal to point p2 if `p1.x == p2.x` and `p1.y == p2.y`
- Implement the `Comparable<Point>` interface and the `compareTo` method. Point p1 is said to be greater than p2 if `p1.x > p2.x` or if `p1.x == p2.x` and `p1.y > p2.y`
- Override the `toString` method to return a string as `[x value, y value]`
- Implement the `Cloneable` interface and `clone` method

Use `@Override` annotation for all overridden methods. Test your program using the following code:

```
13 public class TestPoint {
14     public static void main(String[] args) {
15         Point p1 = new Point(3, 4);
16         Point p2 = new Point(3.4, 1.4);
17         System.out.println(p1.equals(p2));
18         System.out.println(p1.equals(p1));
19         System.out.println(p1.compareTo(p2));
20         System.out.println(p2.compareTo(p1));
21
22         Point p3 = (Point)p1.clone();
23         System.out.println(p3.equals(p1));
24         System.out.println(p3);
25     }
26 }
```

Name your test class file as `TestPoint.java`.

QUESTION 4 (15 Marks) Characters Around Circle JavaFX

Write a program that displays a string "Welcome to Java " around a circle, as shown in Figure 1. Note there is a space at the end of the string. Name your file as `CharactersAroundCircle.java`. Hint: Create a `Text` instance for each character.

You need to display each character in the right location with appropriate rotation using a loop.

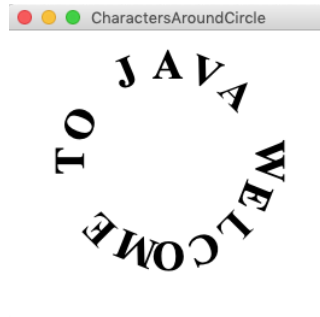


Figure 1: A string is displayed around the circle

QUESTION 5 (15 Marks) Draw A Detailed Clock *JavaFx*

Modify the `ClockPane` class discussed in the lecture to draw the clock with more details on the hours and minutes, as shown in Figure 2. Name the class as `DetailedClockPane`. Write `DetailedClockPane` by modifying `ClockPane`. Do not let `DetailedClockPane` extend `ClockPane`. Write a complete `DetailedClockPane` class by providing proper constructors, setter and getter methods, and other methods necessary in your implementation. Include your test program (name it as `TestDetailedClockPane`) in your submission. The file `ClockPane.java` is given.

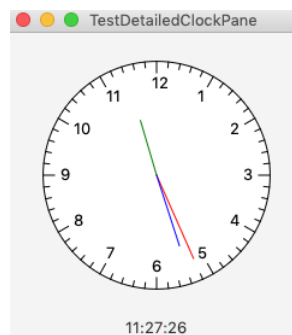
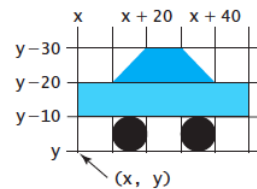
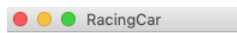


Figure 2: A detailed clock

QUESTION 6 (15 Marks) Racing Car

JawaFX

Write a program that simulates car racing, as shown in Figure 3(a). The car moves from left to right. When it hits the right end, it restarts from the left and continues the same process. You can use a timer to control animation. Redraw the car with a new base point (x, y) , as shown in Figure 3(b). Also let the user pause/resume the animation with a button press/release and increase/decrease the car speed by pressing the UP and DOWN arrow keys. Name your test class as `RacingCar.java`.



- (a) The program displays a moving car. (b) You can redraw a car with a new base point.

Figure 3: Racing car