CS203 Java Programming and Applciations
Winter 2019

Assignment 3 (4 Questions, 100 marks)

Assigned Date: February 13, 2019
Due Date: March 14, 2019 @ 00:00 (Midnight)

QUESTION 1 (20 Marks) Exception Handling

PART I: (10 Marks)

Create an exception hierarchy of `ExceptionA`, `ExceptionB` and `ExceptionC` such that `ExceptionB` inherits from `ExceptionA` and `ExceptionC` inherits from `ExceptionB`. Write a test program to show that the catch block for supertype exception (`ExceptionA`) can catch all the subtype exceptions (`ExceptionB` and `ExceptionC`). Print out a message such as "ExceptionB caught" to indicate which type of exception is caught. Name your class file as `Assignment3Question1Part1.java`.

Hints:

1. To build the exception hierarchy, `ExceptionA`, `ExceptionB` and `ExceptionC` may have empty body.
2. You may use `System.err.println("ExceptionB caught");` instead of `System.out.println("ExceptionB caught");` so that the message will be output in red color and in the format of exception messages. This is a more common way to output error messages.

PART II: (10 Marks)

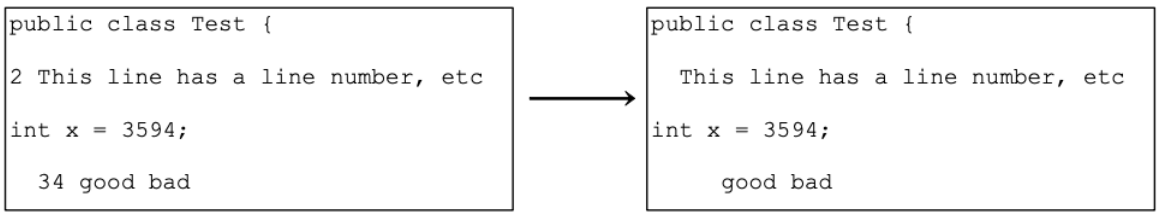
Use the exception hierarchy from PART I to demonstrate that the order of catch block is important if you want to specifically catch every exception in the same hierarchy. In other words, now you should be able to catch subtype exceptions (`ExceptionB` and `ExceptionC`) separately rather than using one catch block of supertype exception (`ExceptionA`) as you did in PART I.

You should define a method called `someMethod()` that can throw all three exceptions ***randomly***. To do this, generate a random integer of 1, 2 or 3 representing

ExceptionA, ExceptionB or ExceptionC respectively, and then throw the corresponding exception. Output the random number in someMethod() to show which type of exception is randomly generated. Invoke someMethod() in your test program and catch all three exceptions separately in three catch blocks. In each catch block, print out a message such as "ExceptionA caught" to indicate which type of exception is caught. Run your program a few times to make sure that someMethod() may throw different exception for different run and you are able to catch all three types of exceptions. Name your class file as Assignment3Question1Part2.java.

QUESTION 2 (20 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 Assignment3Question2.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 3 (40 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: (15 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 $\sqrt{p(p-\texttt{side1})(p-\texttt{side2})(p-\texttt{side3})}$ where p is half the perimeter, that is, $p = \frac{\texttt{side1}+\texttt{side2}+\texttt{side3}}{2}$.)
- 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: (20 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 4 (20 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:

```
public class Assignment3Question4 {
  public static void main(String[] args) {
    Point p1 = new Point(3, 4);
    Point p2 = new Point(3.4, 1.4);
    System.out.println(p1.equals(p2));
    System.out.println(p1.equals(p1));
    System.out.println(p1.compareTo(p2));
    System.out.println(p2.compareTo(p1));

    Point p3 = (Point)(p1.clone());
    System.out.println(p3.equals(p1));
    System.out.println(p3);
  }
}
```

Name your test class file as `Assignment3Question4.java`.