

Name: \_\_\_\_\_

## ISTE-121

### Bonus Lab A: Exceptions

#### Exercise 1 - Exception Handling Concept (5 points)

##### Part 1: Throw Exceptions (Checked Exceptions) (Circle.java)

Open the file `Circle.java` in today's downloads. Add the body to the Constructor that is already present in the `Circle` class. Check to see if the radius parameter entered is greater than 0.0 or not (i.e. is the circle valid?) If the value of the radius is less than zero, throw an `Exception`. Otherwise, proceed as normal by initializing the instance variable `radius` to the parameter entered. A `try/catch` block is not needed here, but the method signature will require modification.

Similarly, in the `setRadius()` method, add the implementation to the body such that if the `newRadius` parameter is less than or equal to 0.0, throw an `Exception`. Otherwise, set the `radius` to this `newRadius` value.

Again, in the `stretchBy()` method, add the implementation to the body such that if the `factor` parameter is less than or equal to 0.0, throw an `Exception`. Otherwise, change the `radius` by multiplying it by the `factor`.

Finally, write the `toString()` method to return the string "Circle: " with the value of the `radius` appended to the end. Make sure the class compiles. A test class will be necessary to run the program.

##### Part 2: User-Defined Exception Class (ShapeException)

Create a file called `ShapeException.java` with the class `ShapeException` which extends the `Exception` class.

This `ShapeException` class has only ONE method, the constructor that takes ONE parameter, a `String` message stating the nature of the exception. Inside the constructor, call the parent's one-parameter constructor with the message as a parameter. Note: this is a very short class!

Now change the `Circle` class by replacing `Exception` with `ShapeException`. When a `ShapeException` is thrown (`throw new ShapeException("...")`), include an appropriate message as the argument to the constructor. Compile both classes. Again, a test class will be required.

**Part 3: Handle the Exceptions (TestCircle.java)**

Use TestCircle from today's downloads.

This program reads in the first command-line argument and uses it to create a circle of the specified radius. Next, print out the radius value using the toString() method of the Circle class.

**Sample Output before Handling Exceptions (TestCircle.java)**

```
Command Prompt

dkpvcs> java TestCircle 5
Circle: 5.0

dkpvcs> java TestCircle -10
Exception in thread "main" ShapeException: Bad radius in constructor: -10.0
    at Circle.<init>(Circle.java:22)
    at TestCircle.main(TestCircle.java:14)

dkpvcs> java TestCircle fubar
Exception in thread "main" java.lang.NumberFormatException: For input string: "fubar"
    at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
    at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
    at java.base/java.lang.Double.parseDouble(Double.java:549)
    at TestCircle.main(TestCircle.java:11)

dkpvcs>
```

Now it's time to use try/catch to prevent the Exceptions shown in the above sample output from ever making it to the terminal.

Copy the above program (TestCircle.java) to another file named TestCircleA.java. Set up catch blocks for the following exceptions: Exception, ShapeException, ArrayIndexOutOfBoundsException, and NumberFormatException. Pay attention to the order of the catch blocks. For each exception, it is OK to print out a message saying what kind of Exception occurred.

**Sample Output after Handling Exceptions (TestCircleA.java)**

```
Command Prompt

dkpvcs> java TestCircleA 20
Circle: 20.0

dkpvcs> java TestCircleA
ArrayIndexOutOfBoundsException occurred...

dkpvcs> java TestCircleA -10
ShapeException occurred...

dkpvcs> java TestCircleA fubar
NumberFormatException occurred...

dkpvcs> _
```

Submit your .java files to the Bonus Lab A Assignment folder when Exercise 1 is working correctly.

## Exercise 2 - Applications Using Exception Handling (5 points)

### Part 1: Catching Exceptions when using Scanner (TestCircleB.java)

Copy TestCircleA.java to a file named TestCircleB.java.

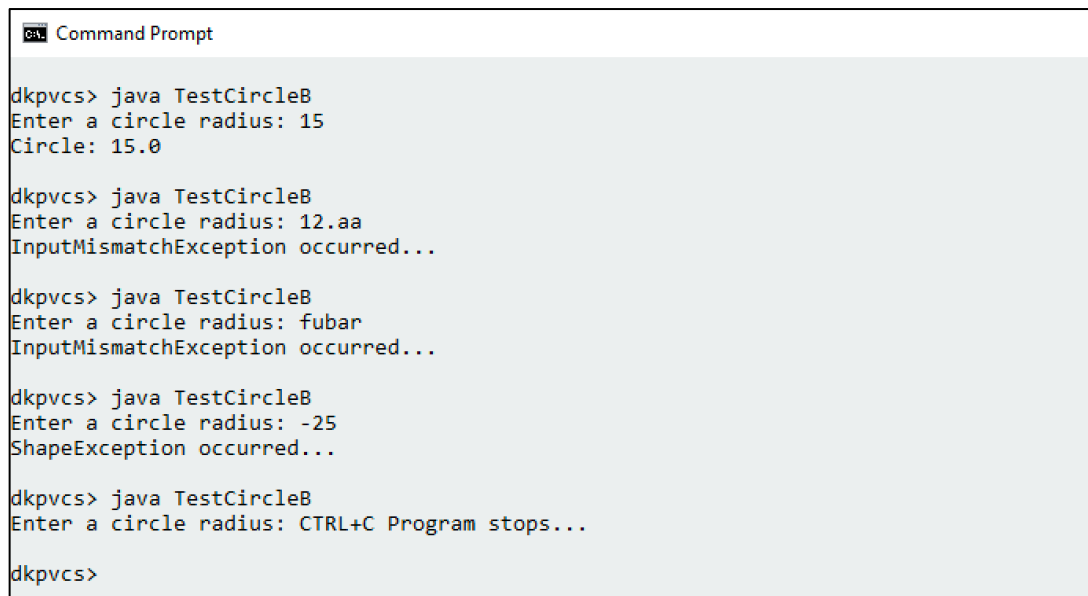
Modify the program to print a prompt for the input value and to read in the double input value using the **Scanner** class. Do not test the input using the `hasNextInt` method.

Modify your program to catch the exceptions thrown by the **Scanner** class. Set up try/catch for the following exceptions: `Exception`, `ShapeException`, `InputMismatchException`, and `NoSuchElementException`.

When the exception for entering the `Control-C` occurs, print the message "CTRL+C entered - Program terminated." For all other exceptions, it is sufficient to print out a message saying what kind of exception occurred.

NOTE: jGRASP catches a Control-C and does not pass it on to your program. You must execute your program from a command prompt window to see if your program catches the Control-C properly. Run your program and when it prompts for a value, press Control-C.

### Sample Output after Handling Exceptions



```
Command Prompt

dkpvcs> java TestCircleB
Enter a circle radius: 15
Circle: 15.0

dkpvcs> java TestCircleB
Enter a circle radius: 12.aa
InputMismatchException occurred...

dkpvcs> java TestCircleB
Enter a circle radius: fubar
InputMismatchException occurred...

dkpvcs> java TestCircleB
Enter a circle radius: -25
ShapeException occurred...

dkpvcs> java TestCircleB
Enter a circle radius: CTRL+C Program stops...

dkpvcs>
```

## Part 2: Repeating Prompt (TestCircleC.java)

Copy your program (TestCircleB.java) to another file named TestCircleC.java.

Put the try/catch in a loop so that the user gets to re-enter the input until it is a valid integer.

Modify your program to catch the exceptions thrown by the **Scanner** class, print a message, and then allow the user to re-enter the input. When valid input is entered, print the radius of the circle and have the program terminate.

### Notes:

- Use a boolean variable to determine when to exit the loop
- When a Control-C is entered, print a message "Control-C entered - program terminated."
- Try to be efficient - use as few statements as possible in the try block

### Sample Output after Handling Exceptions

```

C:\ Select Command Prompt

dkpvcs> java TestCircleC
Enter a circle radius: 12.aa
InputMismatchException occurred...
Enter a circle radius: fubar
InputMismatchException occurred...
Enter a circle radius: -20
ShapeException occurred...
Enter a circle radius: 25
Circle: 25.0

dkpvcs> java TestCircleC
Enter a circle radius: -10
ShapeException occurred...
Enter a circle radius: 38.ww
InputMismatchException occurred...
Enter a circle radius: CTRL+C entered - Program terminated.

dkpvcs> _
```

**Submit your .java files to the Bonus Lab A Assignment folder when Exercise 2 is working correctly.**