Submission deadline: April 5, 2020 23:59:59.

**Name : Punyaja Mishra  
Student id : 0660001**

**For all coding questions: submit Java source files, and screenshots of your output. For all modeling questions: submit PDF files for your models.**

Question#1 (Total 4 points)

import java.util.Date;

public class InternetCharge implements Cloneable {

double bandwith; boolean isLimited; double monthlyRate; Date startDate;

public InternetCharge(double bandwith, boolean limited,

double monthlyRate, Date startDate) { this.bandwith = bandwith; this.isLimited = limited; this.monthlyRate = monthlyRate; this.startDate = startDate;

}

public void setMonthlyRate(int rate) throws Exception {

if (rate < 0)

throw new Exception("Rate cannot be negative");

if (isLimited) {

this.monthlyRate = rate \* 1; } else if (!isLimited) {

this.monthlyRate = rate \* 1.5;

}

}

public double getMonthlyCharge() {

return monthlyRate \* bandwith \* 0.5;

}

@Override

public Object clone() {

// Write your code here

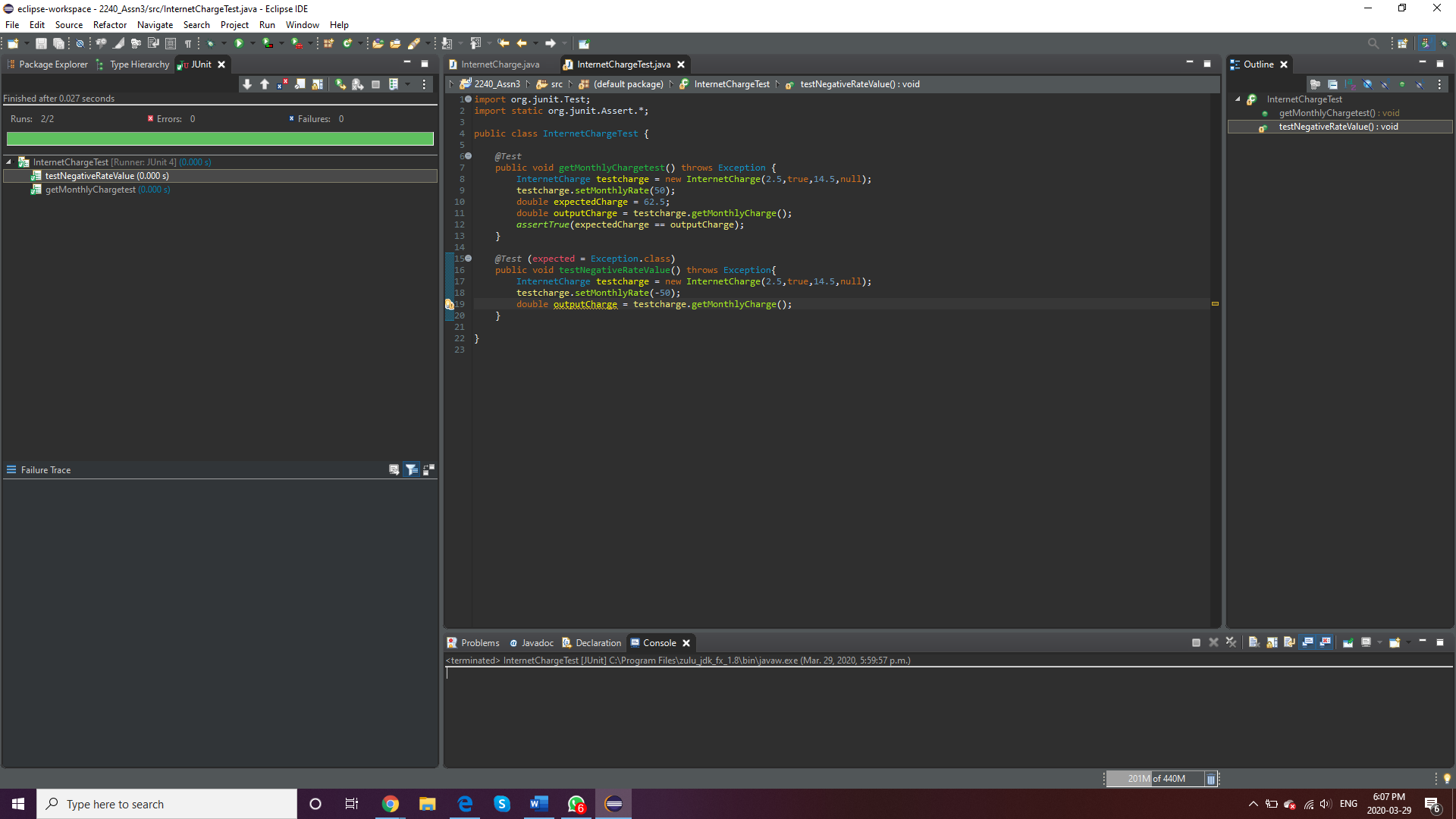
}

}

1. The method clone is provided in the Object class. Override the method clone to perform deep copy for InternetCharge. (2 points)

Java code source files included in the folder.

b) Write two test methods for InternetCharge. The first test method is to test whether the getMonthlyCharge operation is correct. The second test method is to test whether an exception is thrown by the setMonthlyRate method. (2 points)



Question#2: (Total 3 points)

1. What the Adapter pattern is useful for? What is the difference between class adapter and object adapter? (1 points)

Adapter Design Pattern is very important.

🡺We can make two interfaces work which are not related without modifying the original code. 🡺Also, we can make two existing classes work with others without modification of the source code.

🡺There is a single adapter class which joins the functionality of two incompatible interfaces or classes.

This is the reason that adapter tool is very essential because it can convert incompatibility to compatible code, without any modification.

Class Adapters vs Object Adapters

🡺Class adapters uses inheritance and object adapters use the composition.

🡺Class adapters can wrap only classes as oppose to the object adapters that can wrap both classes and interfaces.

🡺Class adapters can’t wrap interfaces directly, it must be derived from a base class.

Thus, the object adapter is easier and more useful in most cases/scenarios.

1. Explain what Singleton pattern is useful for? What is the difference between lazy and eager instantiation in the Singleton design pattern. (1 points)

Singleton pattern is one of the simple design pattern in terms of modelling and hence easy to implement.

🡺There is only one instance of class, hence the name singleton. This one unique instance is the global point of access which has access to the resource and storage-related state information making it useful. It prevents other objects from instantiating their own copies of the singleton object.

Lazy and Eager Instantiation

🡺In eager instantiation, an instance is already created and is ready to use. So, as the name suggests, eager instantiation,

🡺In lazy instantiation, an instance is only created when a getInstance() method is called. So, like a lazy way, no instance is created unless someone needs it.

1. Explain what Observer pattern is useful for? (1 points)

Observer Pattern, as the name suggests is used when there is a one-to-many relationship between objects such as if we modify one object, all objects depended to it will also be notified. Thus, It can be used, when a subject has to be observed by one or many observers.

Question#6: (3 points)

On Week#10, we discussed JavaFX. Explain in writing the relationship between a Stage, Scene, Nodes, and Parents (Control or Pane). Illustrate using two examples, one using Shape and another using a Control such as Label.

Stage:

A stage is the visual area which contains all the objects or controls of the Java Fx application. It is like the top-level container.

🡺The primary stage is created by Java FX itself and passed as a parameter to the start of application class.

🡺There is a stage class present in the javafx.stage package

🡺Stage just creates a content area and the title only where we can format the borders of stage and set titles

Scene :

A scene basically represents the physical contents of the java fx

🡺Every stage has a scene. So a stage object is holding the scene

🡺We show scene using stage object, thus scene is out into the stage

🡺Scene class is basically present in the javafx.scene package

🡺Java fx creates a scene graph while adding items to scene, where scene is the root of this graph. A three-type structure is creates while creating a scene graph.

Nodes :

Nodes are all components that are attached to the scene graph we talked earlier under scene.

🡺It is like a control object or a graphical element

🡺Nodes are subclassses of a JavaFX class called javafx scene

🡺Nodes can be manipulated by user.

🡺Nodes is a super class to – TextField, Label. AnchorPane, Canvas, Group, VBox, Button

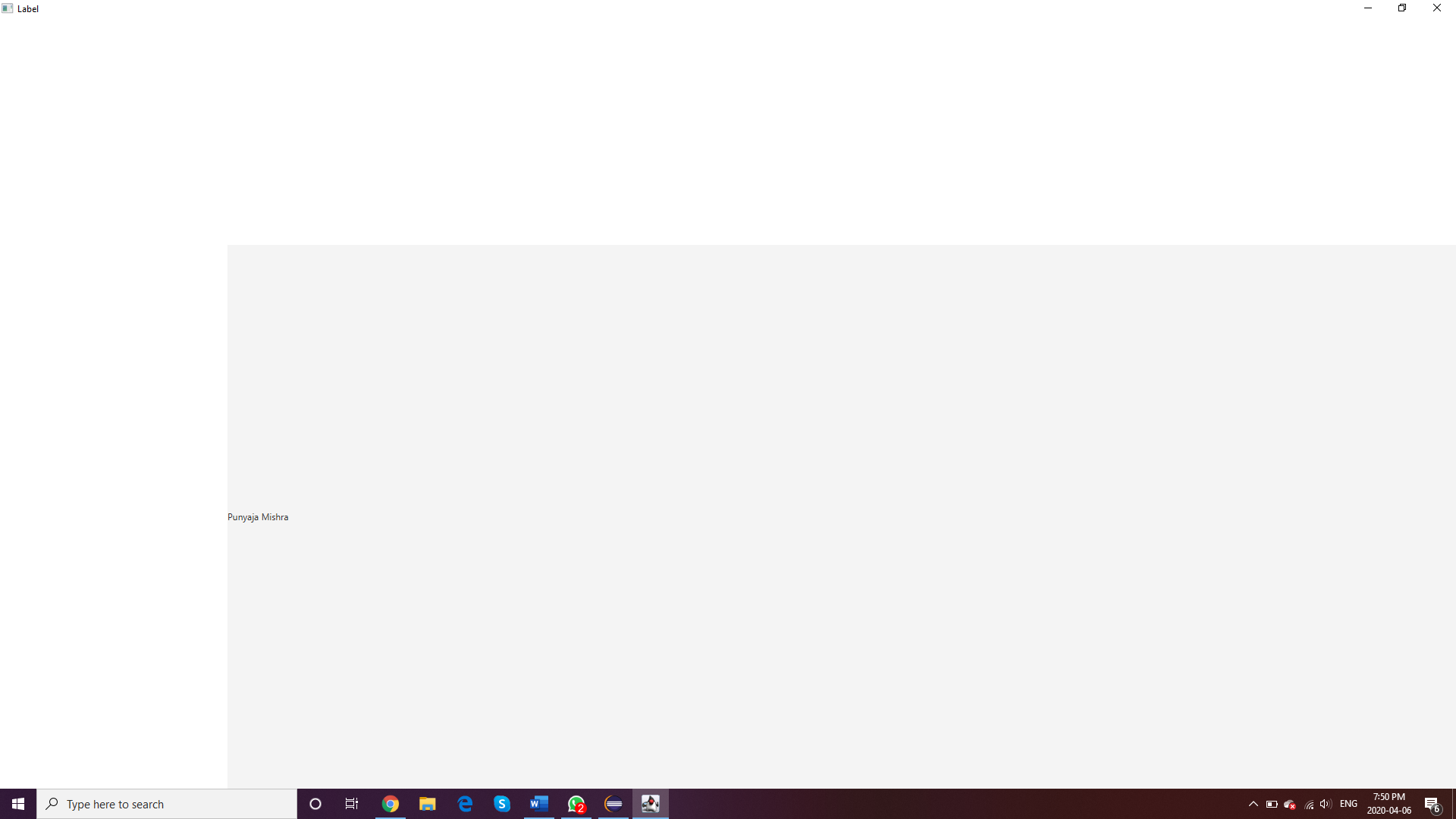
Parents:

Parents, like the name, is like a base class to all nodes that have child classes in the scene graph.

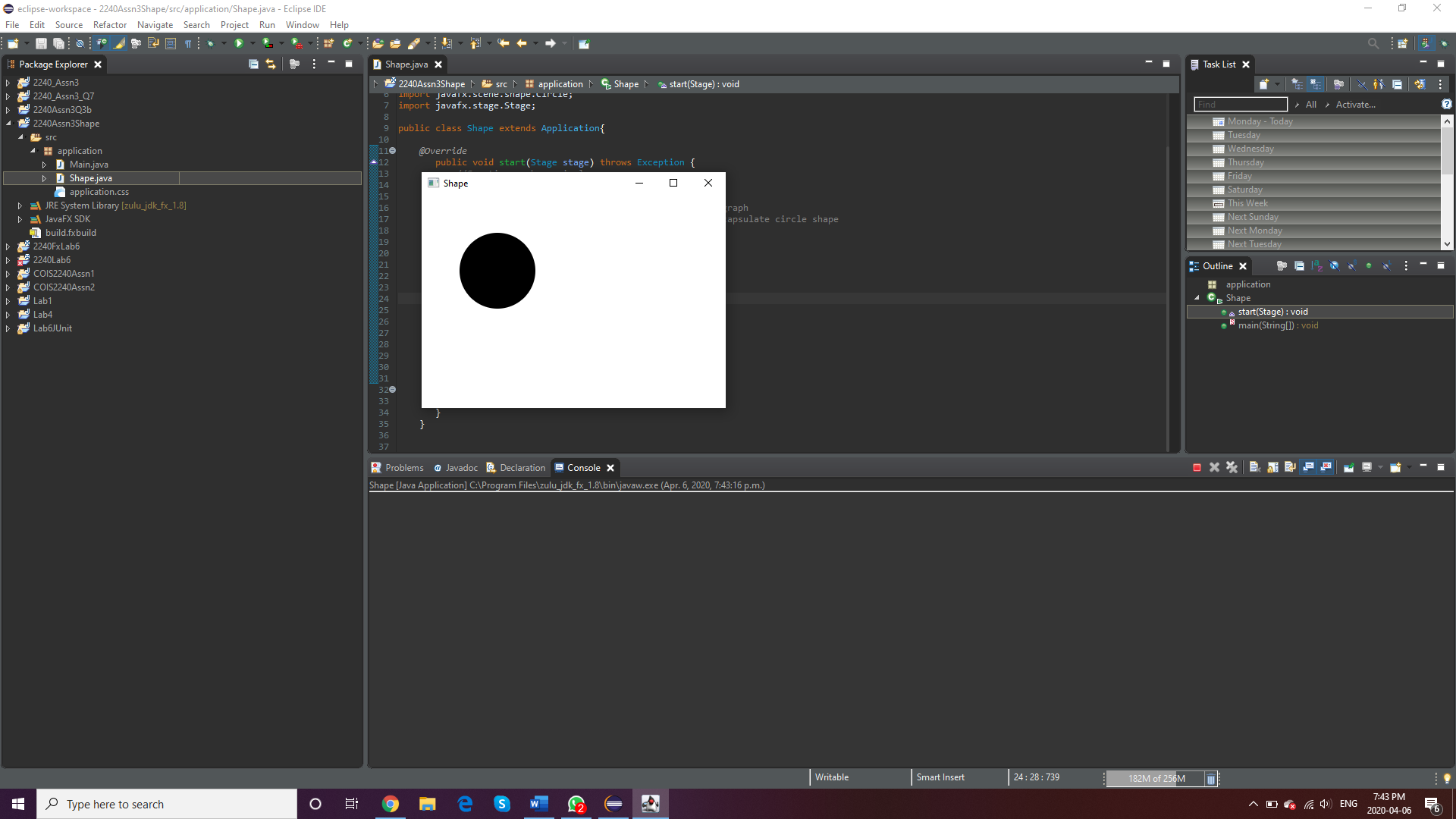
🡺So in a tree structure, the node which contains other nodes is called a parent.

I have two examples I used in JavaFX program, which are attached in the zip folder. Here is the output for the two of them:

In Label, there is a stage that contains scene that contains the label. There is no problem in adding a label to the scene like in shape. So we create a label, let’s say my name, then we set our scene, and put the label in the new scene with dimensions. The we put that scene in the stage and show. I can edit font size and style using correct format functions on the label.



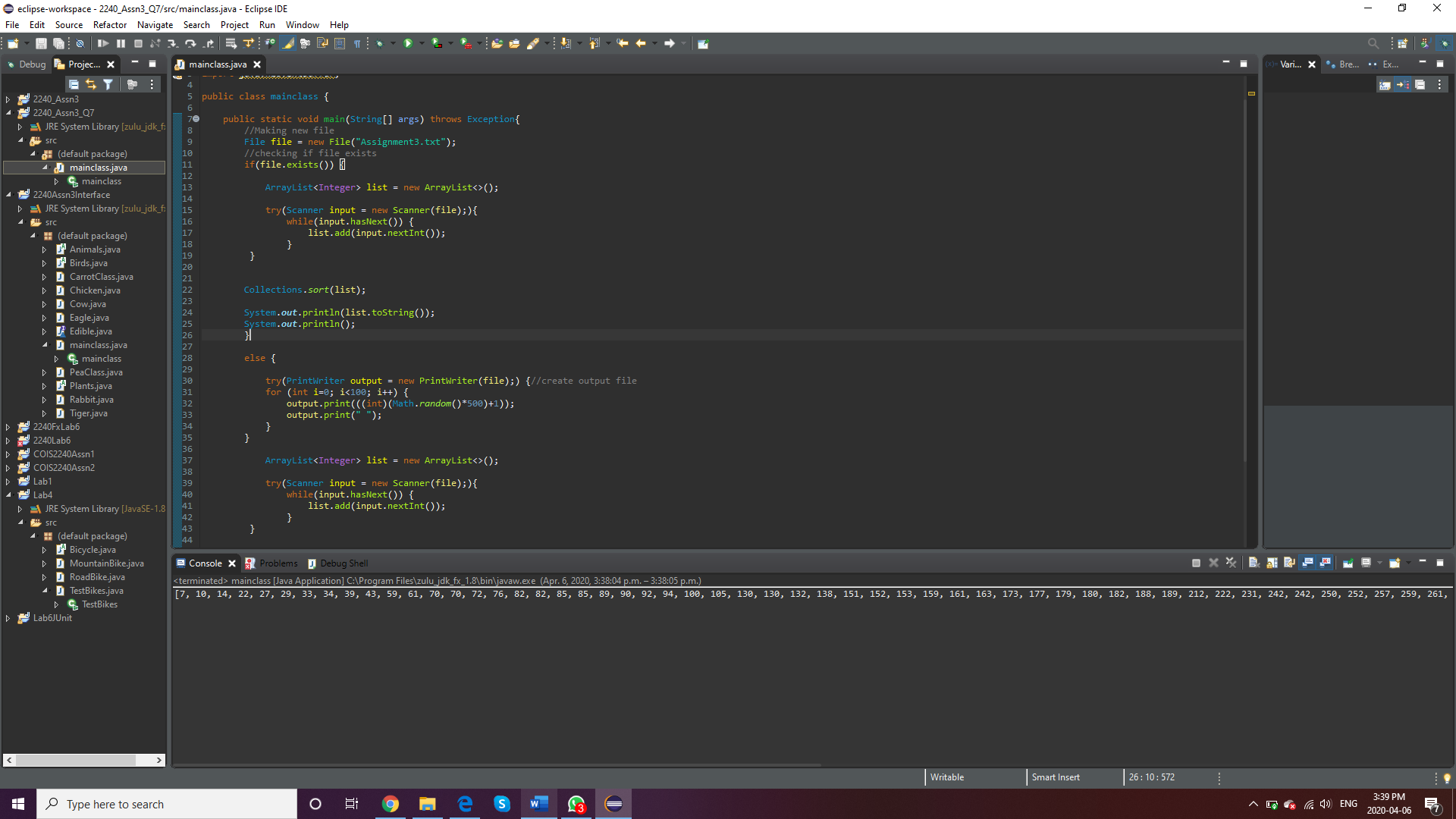
In Shape, the stage contains scene contains AnchorPane which contains the shape. There is this extra thing in between the scene and shape because we are unable to directly add the shape to the scene. So in the example, Anchorpane contains our shape circle, then we give that anchor pane into the new object of scene. Then we set this scene to our stage and give it a title say “Shape”. We can edit the circle using formatting functions to get a colorful circle as well. For that we will sue those functions on circle object.

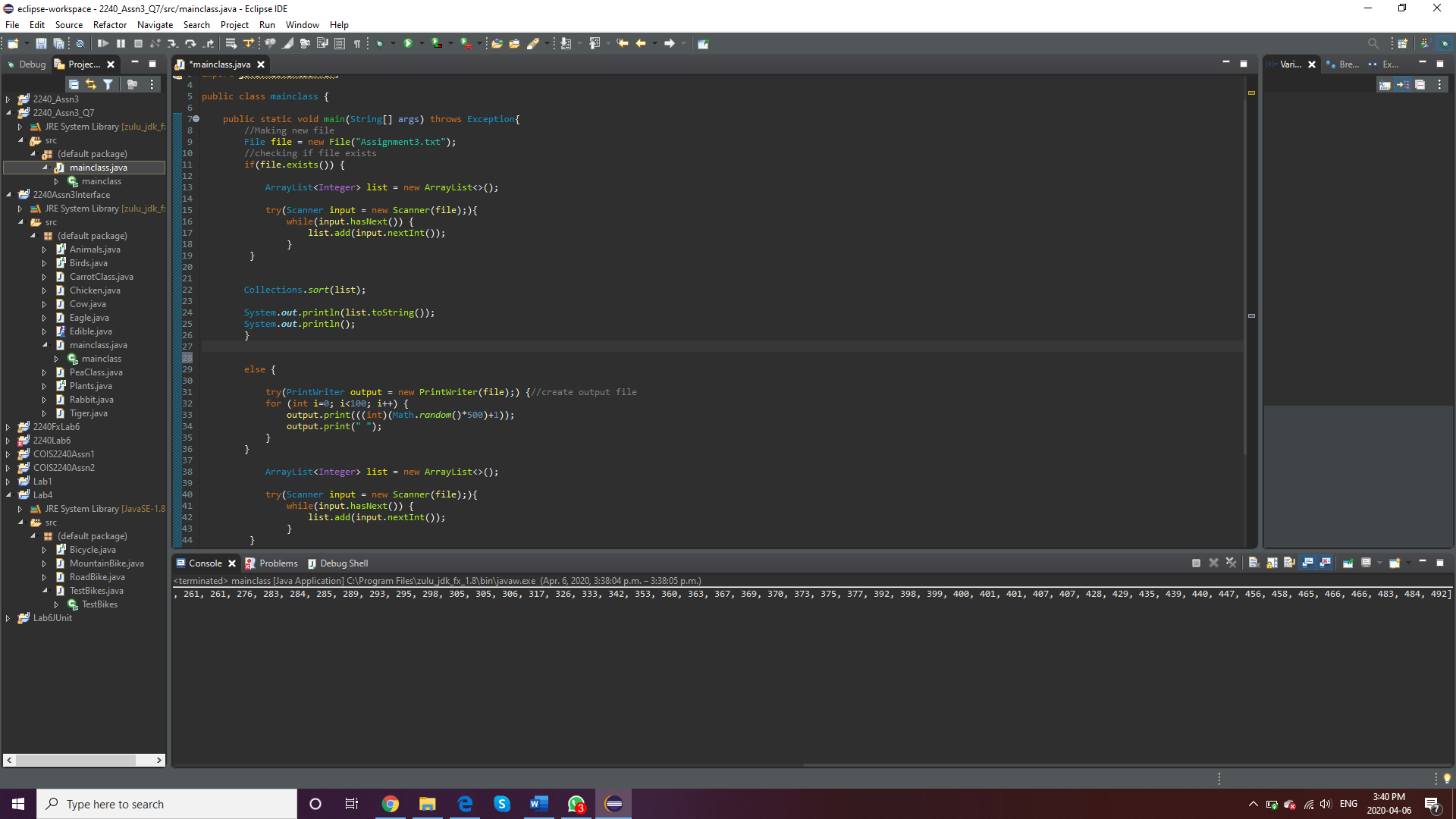


Question#7: (Total 4 points)

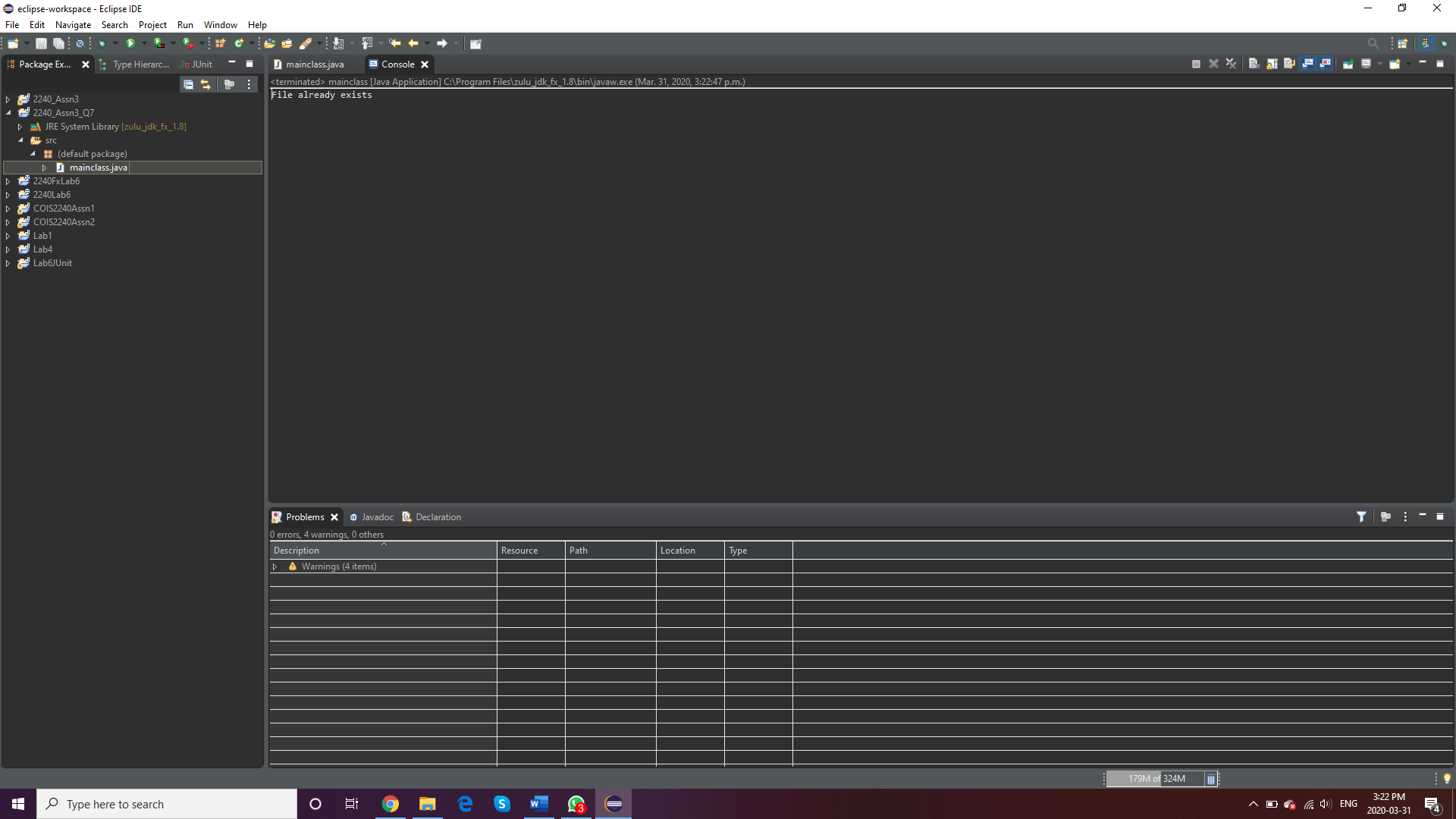
1. Write a Java program to create a file named Assignment3.txt if it does not exist. Write 100 integers created randomly into the file using text I/O. Integers are separated by spaces in the file. Read the data back from the file and display the data in increasing order. (2 points)

The output printing the random 100 integers we stored in the new text file created – “Assignment3.txt”



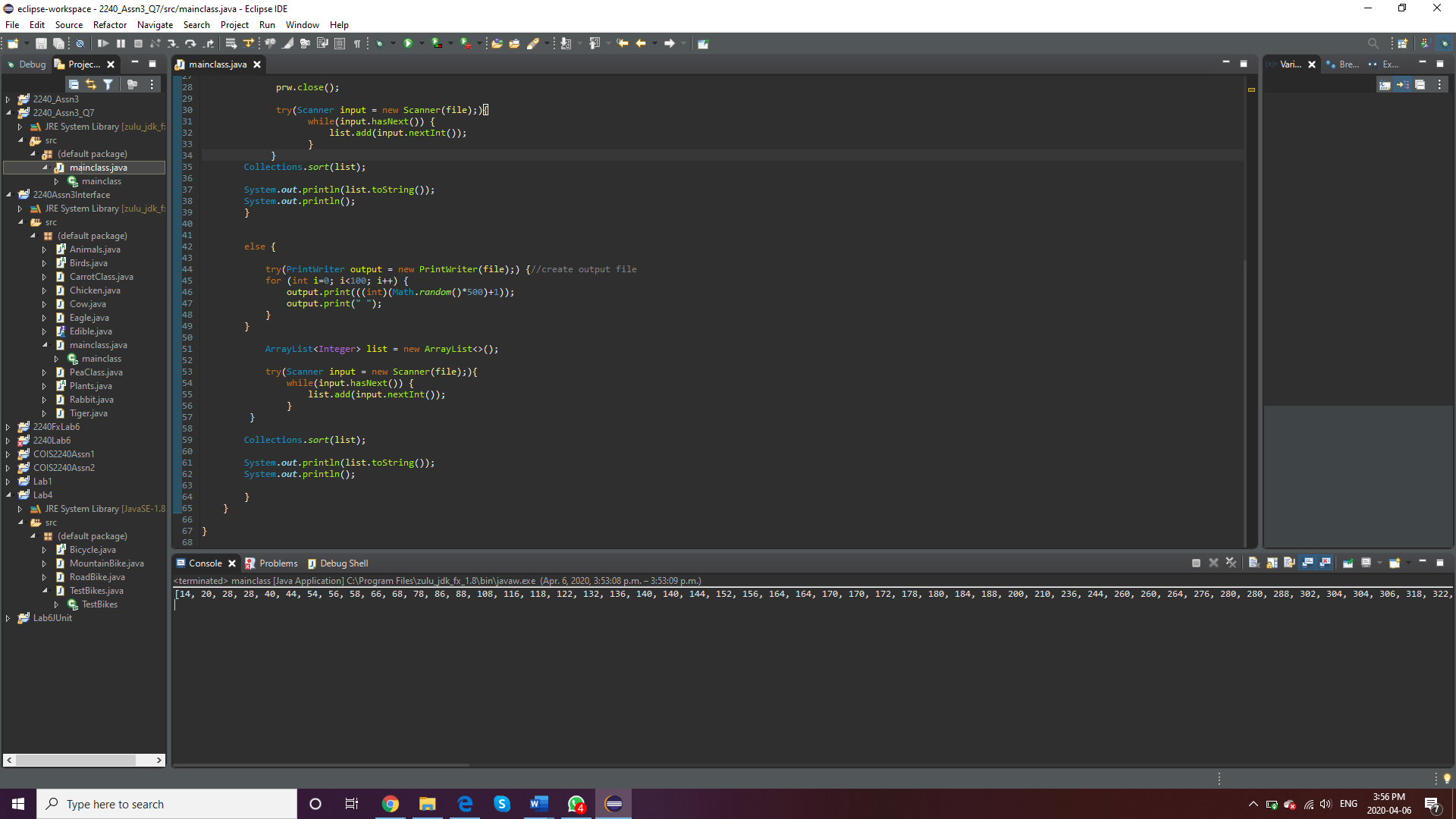


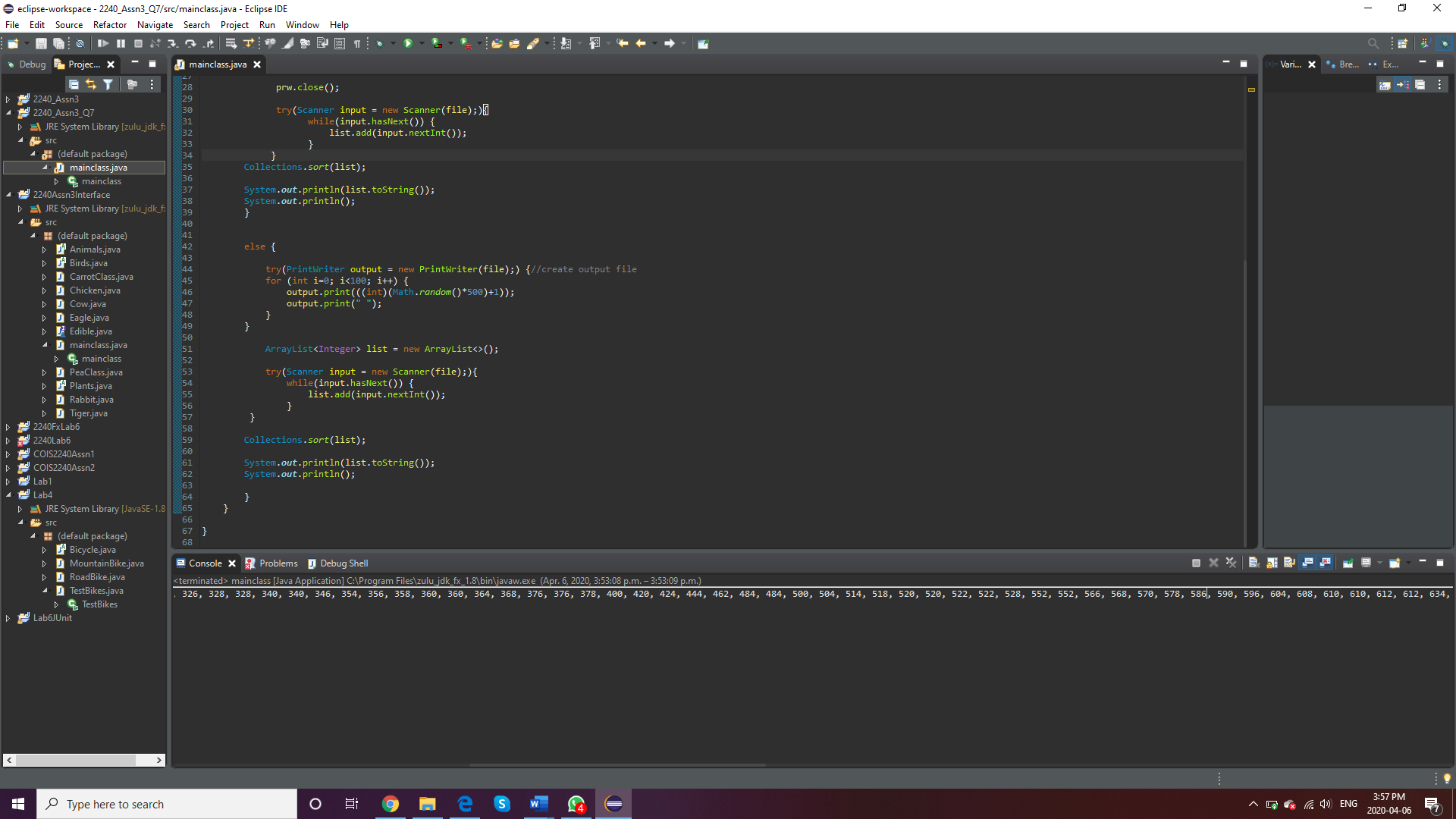
Running the program again to check if try works correctly, since the file already exists.

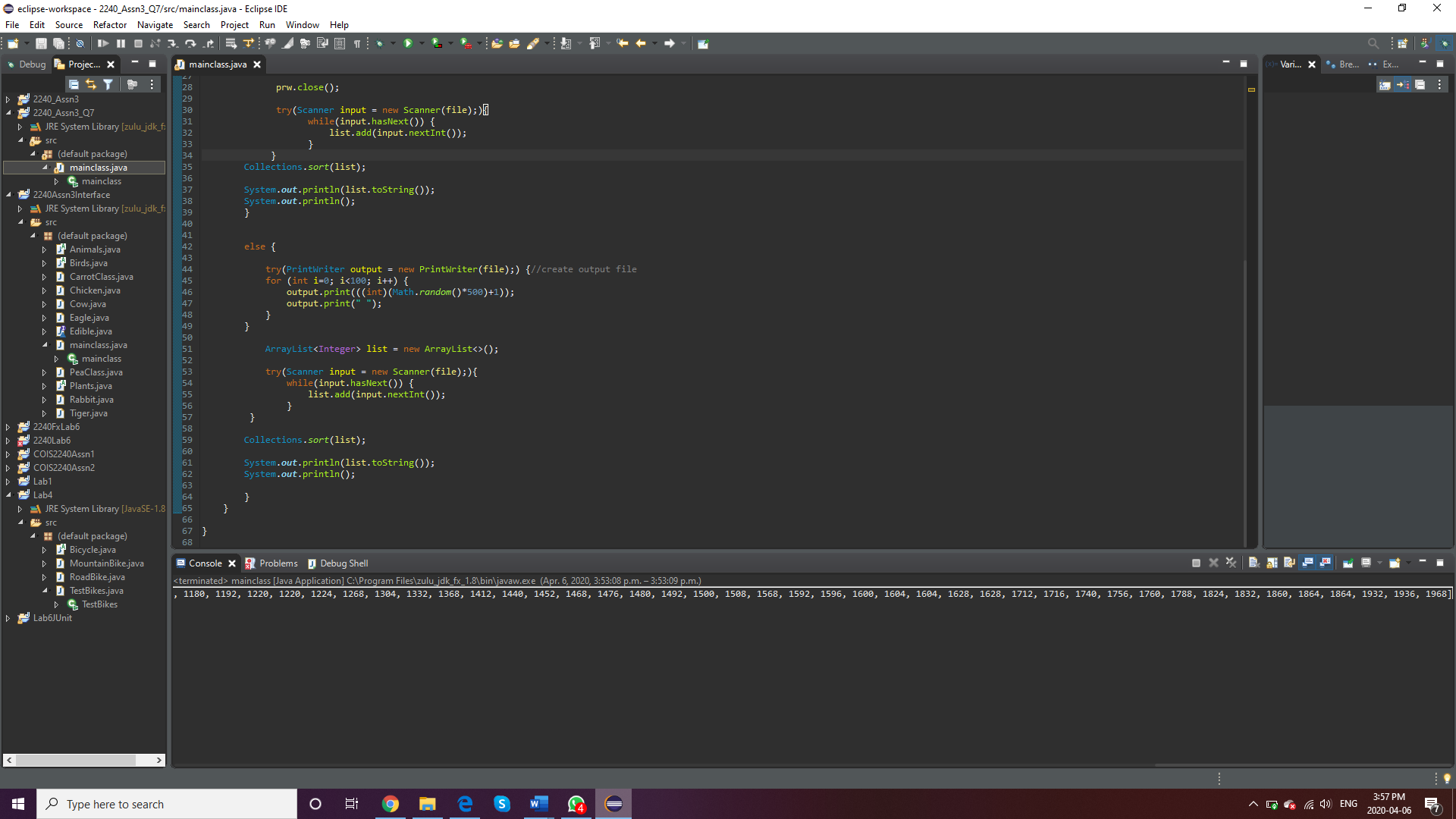


1. Now, in the same Java program replace each number by its multiple of 2. For example, if the number was 4, replace it with 8. (2 points)

Submit Java source file, screenshots of your output and the text file.







Question#8: (2 points)

What is the output of the following program:

class Test {

public static void main(String[] args) {

try { method();

System.out.println("After the method call");

}

catch (RuntimeException ex) {

System.out.println("RuntimeException");

}

catch (Exception ex) {

System.out.println("Exception");

}

}

static void method() throws Exception { try {

String s = "5.6";

Integer.parseInt(s); // Cause a NumberFormatException

int i = 0; int y = 2 / i;

System.out.println("Welcome to Java");

}

catch (NumberFormatException ex) {

System.out.println("NumberFormatException");

throw ex; }

catch (RuntimeException ex) {

System.out.println("RuntimeException");

}

}

}

Output : -

NumberFormatException

RuntimeException