**Exam 2 Instructions**

**OBJECT-ORIENTED PROG**

* This is a take-home exam. You can use any resources that are available for you to finish this exam, except
  + Outsourcing the exam to any person or to any third party websites
  + Copying from other students work
  + Copying direct quotes from the books or internet
* Do not lose your opportunity to learn while working on the exam. Understand the concept and write answers on your own.
* Usually, in life, we have several choices. Unfortunately, you don’t have any choice on this exam. You have to answer all the questions and each part of the problem.
* All the topics on this exam were discussed in class before week 13. So, you cannot claim that the questions are out of the syllabus!
* Refer to Microsoft Word tutorials for proper formatting
* Points will be deducted for grammatical and spelling mistakes
* No two brains think alike unless you are soulmates. Definitely your answers will not be same as other students.
* Read the code of academic integrity before you start the exam. <https://www.nwmissouri.edu/policies/academics/Academic-Integrity.pdf>
* Push your source code to GitHub and provide your GitHub link at the end of the document and in the comment section.
* Don’t use examples that already explained in class or worksheets.
* Provide the input and output screenshots for every program.

**Exam 2 OBJECT-ORIENTED PROG 01FA20 100 pts**

1. (5-Points) (1D-Array - )Write a method that removes the duplicate elements from an array list of integers using the following header:

Public static void removeDuplicate(ArrayList<Integer> list)

Write a test program that prompts the user to enter 10 integers to a list and displays the distinct integers separated by exactly one space. Provide screenshot of executable code with input and output. Here is a sample run:

|  |
| --- |
| Enter ten integers: 34 5 3 5 6 4 33 2 2 4  The distinct integers are 34 5 3 6 4 33 2 |

1(A) Get the ArrayList with duplicate values.

Create another ArrayList.

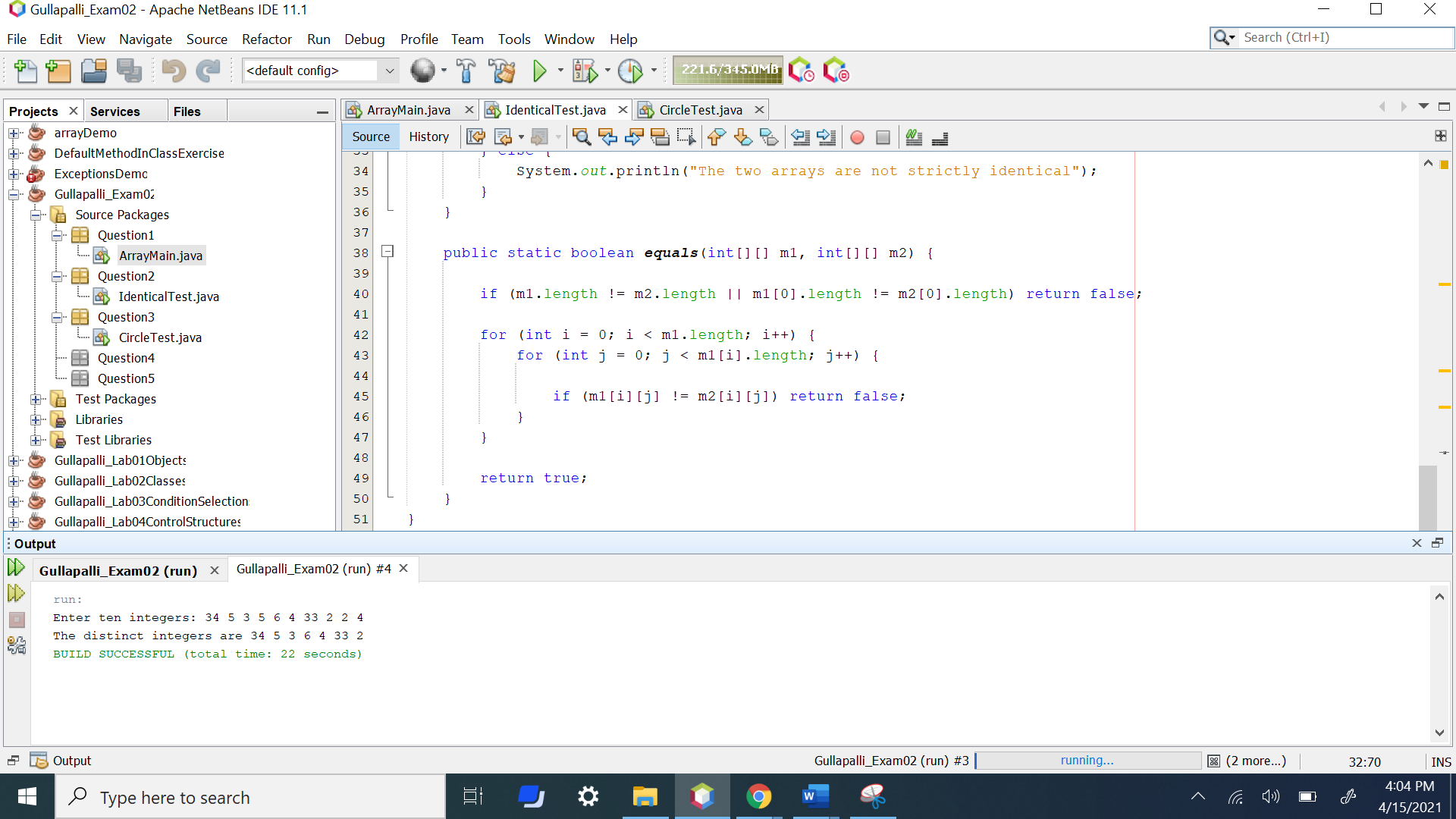
Traverse through the first arraylist and store the first appearance of each element into the second arraylist using contains() method.

The second ArrayList contains the elements with duplicates removed.

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question1;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  import java.util.Scanner;  import java.util.ArrayList;  public class ArrayMain {  public static void main(String[] args) {    Scanner input = new Scanner(System.in);// Created scanner    ArrayList<Integer> list = new ArrayList<Integer>();// Created ArrayList    System.out.print("Enter ten integers: ");// enter ten integers  for (int i = 0; i < 10; i++) {  list.add(input.nextInt());  }    removeDuplicate(list);// Invoke removeDuplicate method  // Display the output  System.out.print("The distinct integers are ");  for (int i = 0; i < list.size(); i++) {  System.out.print(list.get(i) + " ");  }  System.out.println();  }  // Removes the duplicate elements from an array list of integers  public static void removeDuplicate(ArrayList<Integer> list) {  for (int i = 0; i < list.size() - 1; i++) {  for (int j = i + 1; j < list.size(); j++) {  if (list.get(i) == list.get(j))  list.remove(j);  }  }  }  } |

Output :-

|  |
| --- |
| Enter ten integers: 34 5 3 5 6 4 33 2 2 4  The distinct integers are 34 5 3 6 4 33 2 |



1. (5-Points) (2D- Array) The two-dimensional arrays m1 and m2 are strictly identical if their corresponding elements are equal. Write a method that returns true if m1 and m2 are strictly identical, using the following header:

public static boolean equals(int[][] m1, int[][] m2)

Write a test program that prompts the user to enter two 3 \* 3 arrays of integers and displays whether the two are strictly identical. Provide screenshot of executable code with input and output. Here are the sample runs.

|  |
| --- |
| Enter list1: 51 22 25 6 1 4 24 54 6  Enter list2: 51 22 25 6 1 4 24 54 6  The two arrays are strictly identical |

|  |
| --- |
| Enter list1: 51 25 22 6 1 4 24 54 6  Enter list2: 51 22 25 6 1 4 24 54 6  The two arrays are not strictly identical |

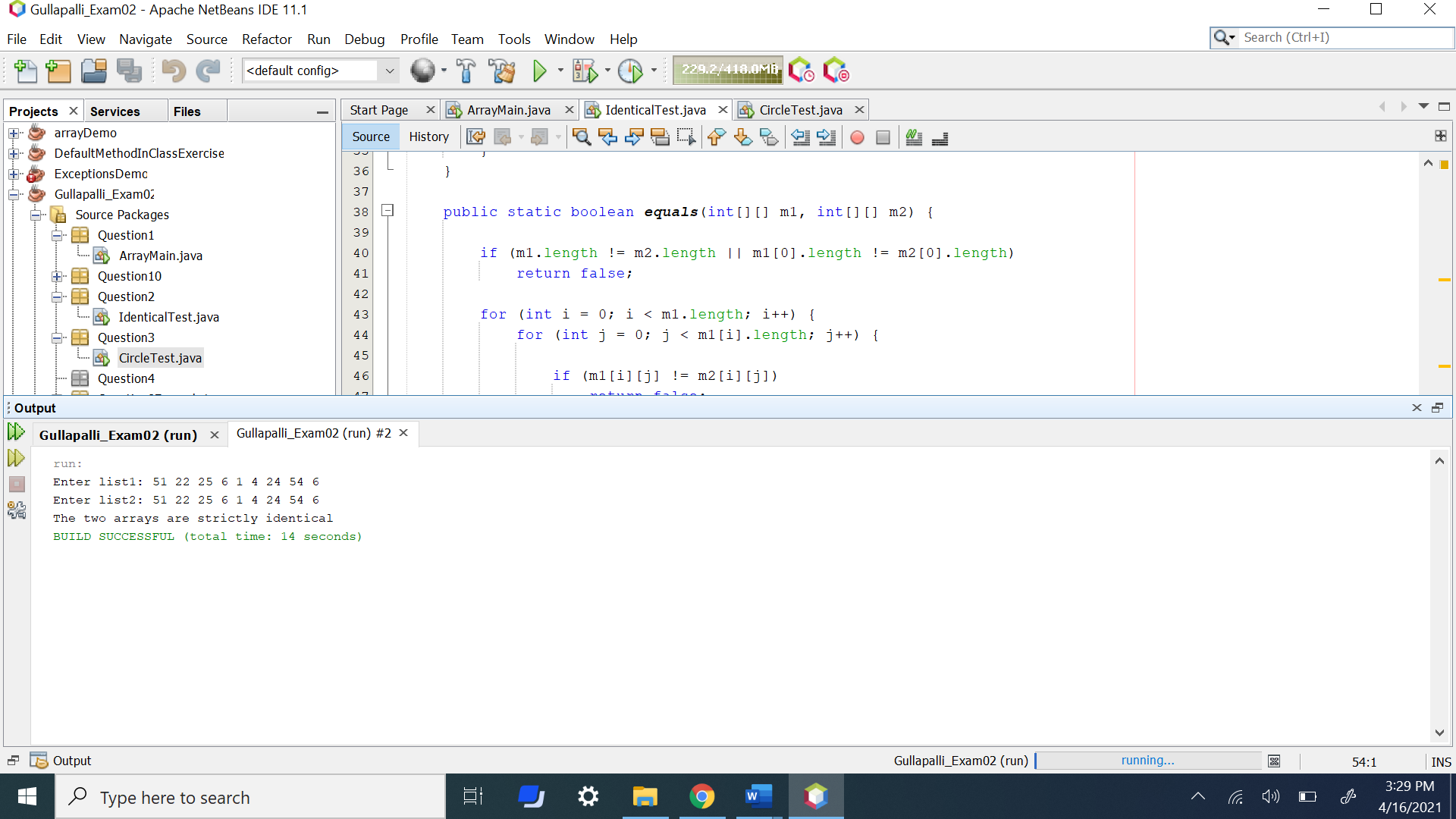
2(A)

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question2;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  import java.util.Scanner;  import java.util.ArrayList;  public class IdenticalTest {  public static void main(String[] args) {  int[][] m1 = new int[3][3];  int[][] m2 = new int[3][3];  Scanner input = new Scanner(System.in);  System.out.print("Enter list1: ");  for (int i = 0; i < m1.length; i++)  for (int j = 0; j < m1[i].length; j++)  m1[i][j] = input.nextInt();  System.out.print("Enter list2: ");  for (int i = 0; i < m2.length; i++)  for (int j = 0; j < m2[i].length; j++)  m2[i][j] = input.nextInt();  if (equals(m1, m2)) {  System.out.println("The two arrays are strictly identical.");  } else {  System.out.println("The two arrays are not strictly identical");  }  }  public static boolean equals(int[][] m1, int[][] m2) {  if (m1.length != m2.length || m1[0].length != m2[0].length) return false;  for (int i = 0; i < m1.length; i++) {  for (int j = 0; j < m1[i].length; j++) {  if (m1[i][j] != m2[i][j]) return false;  }  }  return true;  }  } |

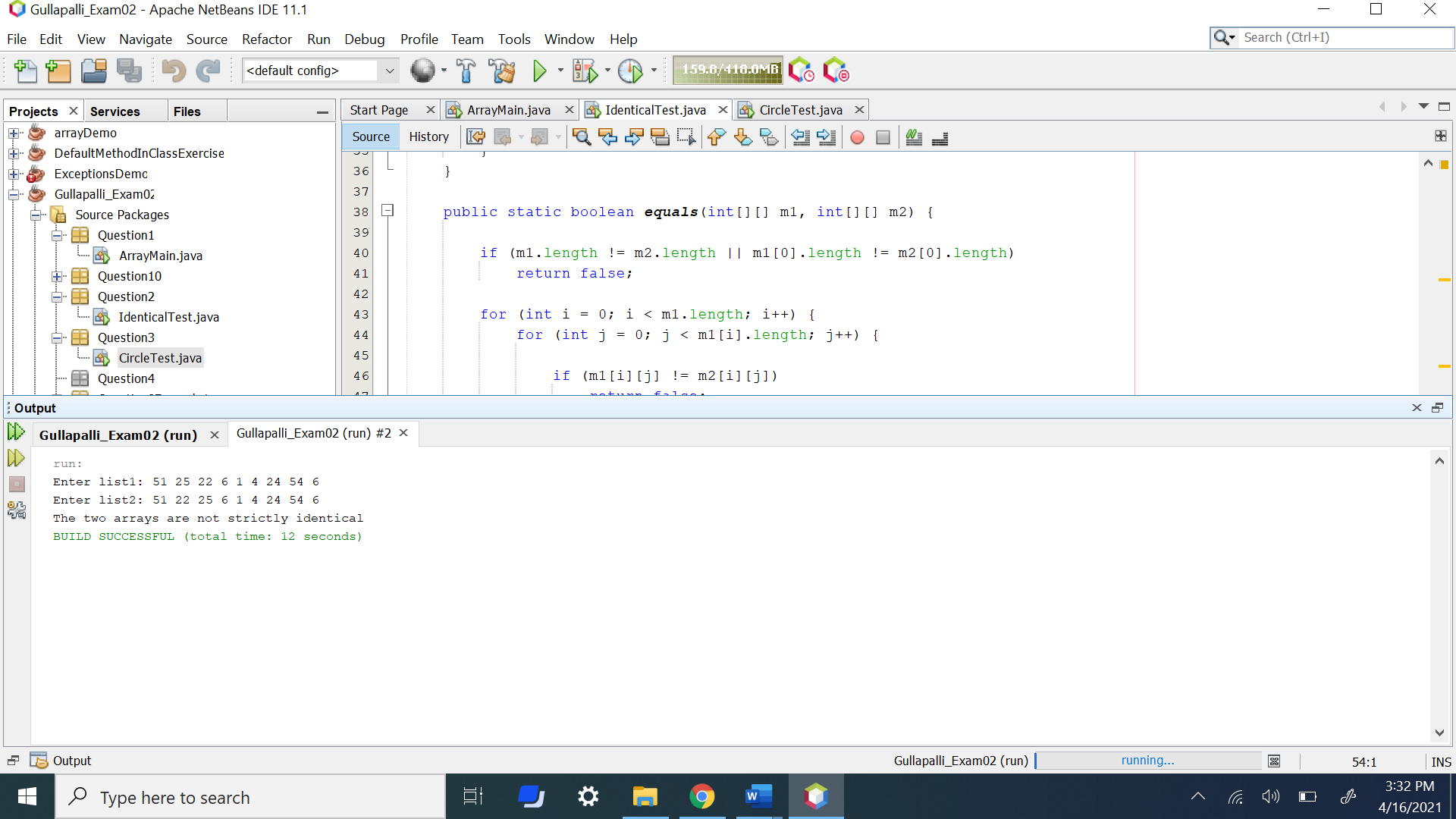
Output :-

|  |
| --- |
| Enter list1: 51 22 25 6 1 4 24 54 6  Enter list2: 51 22 25 6 1 4 24 54 6  The two arrays are strictly identical  Enter list1: 51 25 22 6 1 4 24 54 6  Enter list2: 51 22 25 6 1 4 24 54 6  The two arrays are not strictly identical |

The two arrays are strictly identical



The two arrays are not strictly identical



1. (10-Points) (Array List) Write a program that creates an ArrayList and adds a **Loan** object, a **Date** object (Use inbuilt method. No need to create separate class), a string, and a **Circle** object to the list, and use a loop to display all the elements in the list **by** invoking the object’s **toString**() method.

Note: For **Loan** and **Circle** you can use your own attributes and methods. **Constructor** and **tostring()** are mandatory requirements

CircleTest

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question3;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class CircleTest {  private String name;  private double radius;  private boolean area;  public CircleTest(String name, double radius, boolean area) {  this.name = name;  this.radius = radius;  this.area = area;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public double getRadius() {  return radius;  }  public void setRadius(double radius) {  this.radius = radius;  }  public boolean isArea() {  return area;  }  public void setArea(boolean area) {  this.area = area;  }  @Override  public String toString() {  return "CircleTest{" + "name=" + name + ", radius=" + radius + ", area=" + area + '}';  }  } |

LoanTest

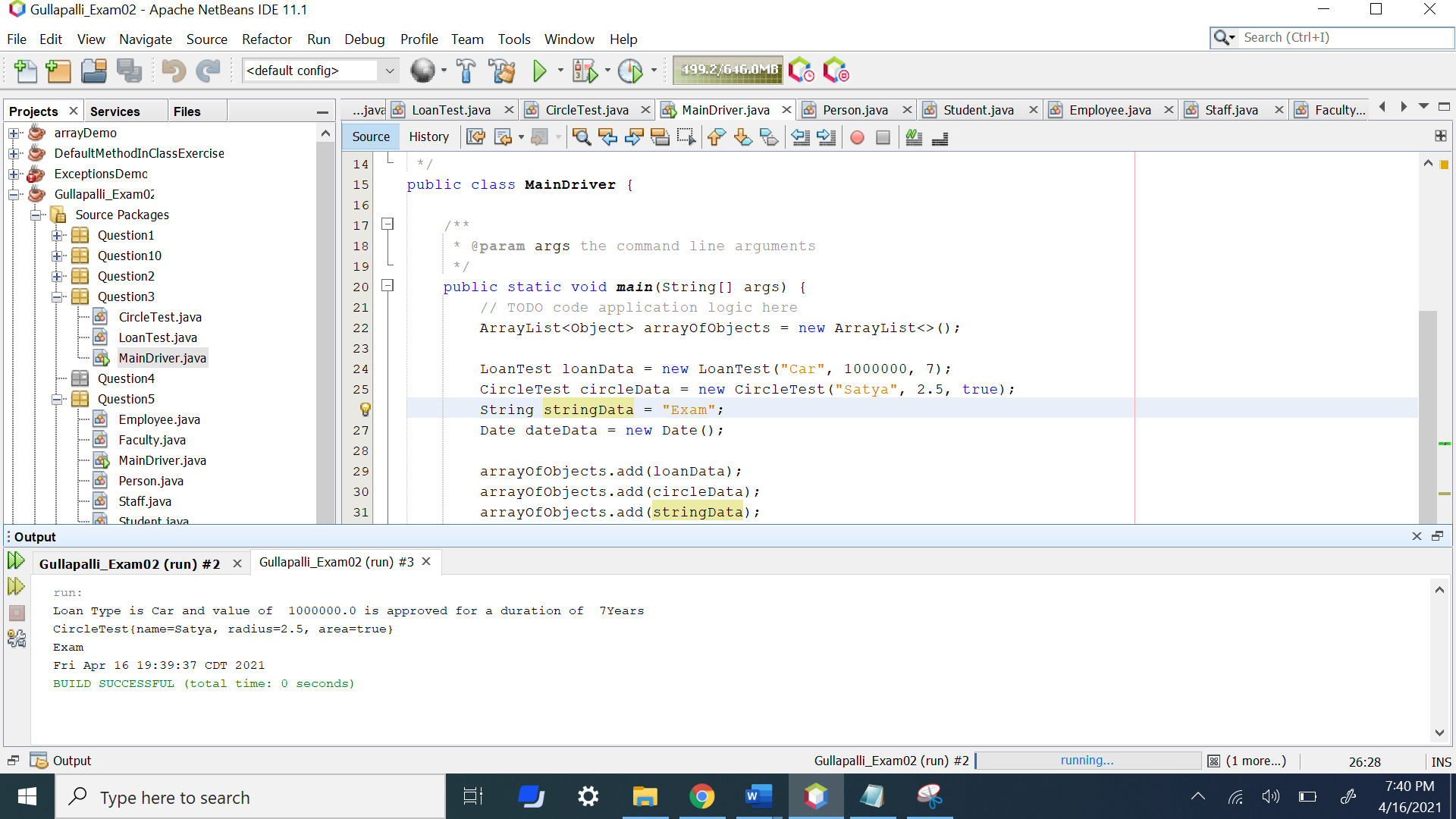
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question3;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class LoanTest {  private String loanType;  private double loanAmount;  private int Duration;  public LoanTest(String loanType, double loanAmount, int Duration) {  this.loanType = loanType;  this.loanAmount = loanAmount;  this.Duration = Duration;  }  @Override  public String toString() {  return "Loan Type is " + loanType + " and value of " + loanAmount  + " is approved for a duration of "  + Duration + "Years";  }  public String getLoanType() {  return loanType;  }  public void setLoanType(String loanType) {  this.loanType = loanType;  }  public double getLoanAmount() {  return loanAmount;  **}**  public void setLoanAmount(double loanAmount) {  this.loanAmount = loanAmount;  }  public int getDuration() {  return Duration;  }  public void setDuration(int Duration) {  this.Duration = Duration;  }  } |

MainDriver

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question3;  import java.util.ArrayList;  import java.util.Date;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class MainDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ArrayList<Object> arrayOfObjects = new ArrayList<>();  LoanTest loanData = new LoanTest("Car", 1000000, 7);  CircleTest circleData = new CircleTest("Satya", 2.5, true);  String stringData = "Exam";  Date dateData = new Date();  arrayOfObjects.add(loanData);  arrayOfObjects.add(circleData);  arrayOfObjects.add(stringData);  arrayOfObjects.add(dateData);  for (Object LCSD : arrayOfObjects) {  System.out.println(LCSD);  }  }  } |

Output **:-**

|  |
| --- |
| Loan Type is Car and value of 1000000.0 is approved for a duration of 7Years  CircleTest{name=Satya, radius=2.5, area=true}  Exam  Fri Apr 16 19:39:37 CDT 2021 |



1. (15-Points) What is Inheritance, Polymorphism and Late binding polymorphism? Explain and demonstrate with examples. Provide executable code screenshots for examples.

4(A) Inheritance:

Inheritance can be defined as the process where one class acquires the properties (methods) of another.The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class). Keyword extends and it is used to inherit the properties.

Syntax:

class Super {

.....

.....

}

class Sub extends Super {

.....

.....

}

Polymorphism:

Polymorphism means single action can be performed in different ways

Two types of polymorphism :

1. Compile-time polymorphism

2. Runtime polymorphism

polymorphism can be performed by method overloading and method overriding.

1. Compile-time polymorphism: It is known as static polymorphism.

it can be achieved by function overloading.

Runtime polymorphism: It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

Method overriding: It occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

Q4\_Inheritance.java

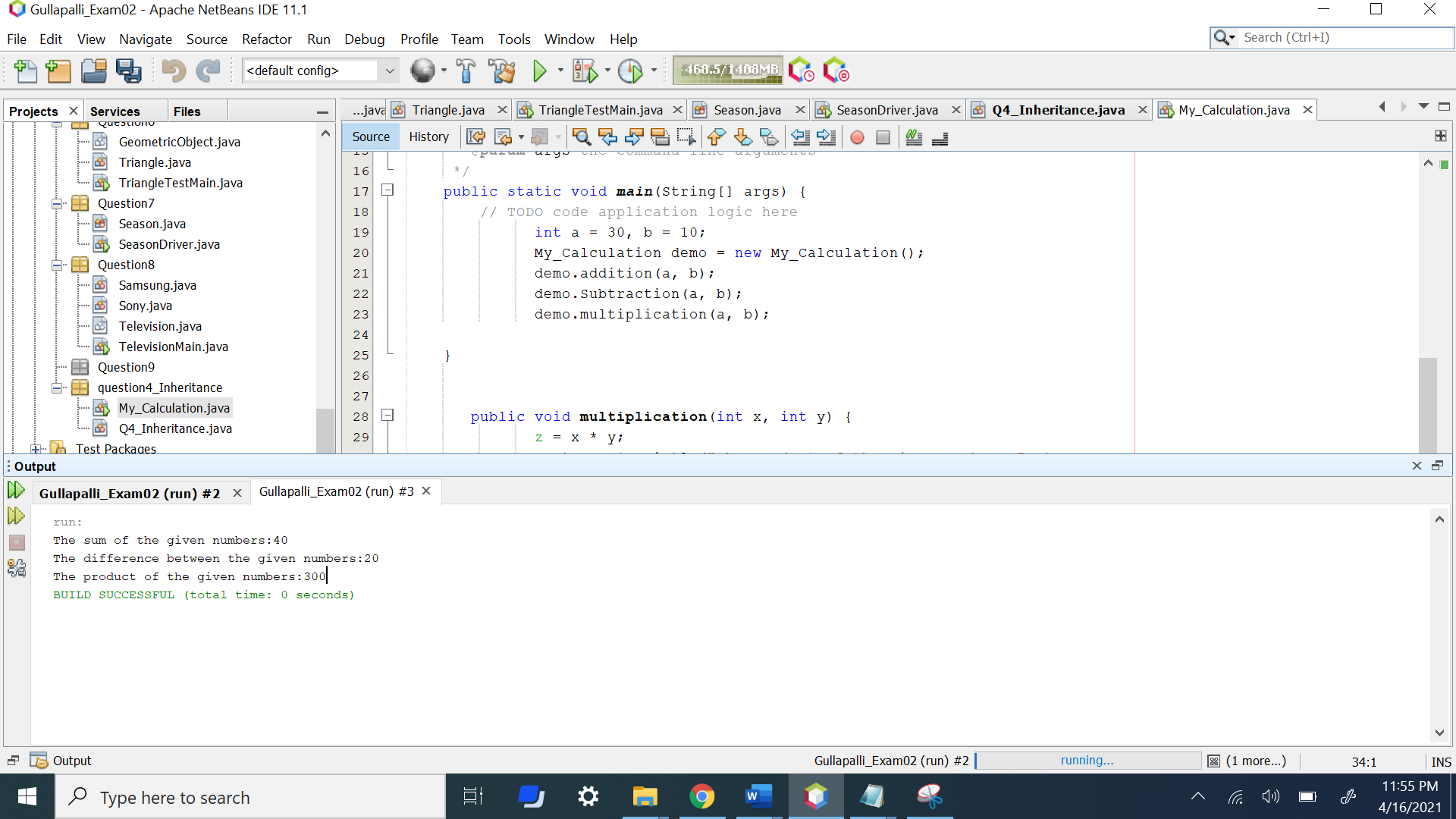
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question4\_Inheritance;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Q4\_Inheritance {    int z;    public void addition(int x, int y) {  z = x + y;  System.out.println("The sum of the given numbers:"+z);  }    public void Subtraction(int x, int y) {  z = x - y;  System.out.println("The difference between the given numbers:"+z);  }  } |

My\_Caluclation.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question4\_Inheritance;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class My\_Calculation extends Q4\_Inheritance {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int a = 30, b = 10;  My\_Calculation demo = new My\_Calculation();  demo.addition(a, b);  demo.Subtraction(a, b);  demo.multiplication(a, b);  }      public void multiplication(int x, int y) {  z = x \* y;  System.out.println("The product of the given numbers:"+z);  }  } |

Output

|  |
| --- |
| The sum of the given numbers:40  The difference between the given numbers:20  The product of the given numbers:300 |



Polymorphism

Testpoly.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Polymorphism;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Testpoly {    public void Shift(){    System.out.println("To day I went to the shift");  }  } |

ShiftChange.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Polymorphism;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class ShiftChange extends Testpoly {    public void Shift(){    System.out.println("I gave my shift to Suresh");  }  } |

PolymorphismDriver.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Polymorphism;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class PolymorphismDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here    ShiftChange sc = new ShiftChange();  sc.Shift();  }    } |

Output

|  |
| --- |
| I gave my shift to Suresh |

Late-binding

Shift.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Latebinding;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Shift {    public void MyShift(){    System.out.println("I went to my shift");  }  } |

ShiftSup.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Latebinding;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class ShiftSup extends Shift {    public void shiftswap(){    System.out.println("Naresh went to the shift in place of me");  }  } |

LatebindingDriver.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question4\_Latebinding;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class LatebindingDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here    ShiftSup sh = new ShiftSup();  sh.shiftswap();  }    } |

Output

|  |
| --- |
| Naresh went to the shift in place of me |

1. (10-Points) Design a class named **Person** and its two subclasses named **Student** and **Employee**. Make **Faculty** and **Staff** subclasses of **Employee**. A person has a name, address, phone number, and email address. A student has a grade and class status (Graduate). Define the status as a constant. An employee has an office, salary, and date hired. A faculty member has office hours and number of teaching subjects. A staff member has a title. Override the **toString** method in each class to display the class name and the person’s name.

Draw the UML diagram for the classes and implement them. Write a test program that creates a **Person**, **Student**, **Employee**, **Faculty**, and **Staff**, and invokes their **toString**() methods.

Note: All classes should have **toString()** Method.

Employee

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  import java.util.Date;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Employee extends Person {  private String office;  private double salary;  private Date dateHired;  public Employee(String office, double salary, Date dateHired, String name,  String address, long phoneNumber, String emailAddress) {  super(name, address, phoneNumber, emailAddress);  this.office = office;  this.salary = salary;  this.dateHired = dateHired;  }  @Override  public String toString() {  return "Person name from " + this.getClass().getName()  + " is " + super.getName();  }  } |

Faculty

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  import java.util.Date;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Faculty extends Employee {  private int officeHours;  private int numberTeachingSubjects;  public Faculty(int officeHours, int numberTeachingSubjects,  String office, double salary, Date dateHired, String name,  String address, long phoneNumber, String emailAddress) {  super(office, salary, dateHired, name, address, phoneNumber,  emailAddress);  this.officeHours = officeHours;  this.numberTeachingSubjects = numberTeachingSubjects;  }  @Override  public String toString() {  return "Person name from " + this.getClass().getName() + " is "  + super.getName();  }  } |

Person

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Person {  private String name;  private String address;  private long phoneNumber;  private String emailAddress;  public Person(String name, String address,  long phoneNumber, String emailAddress) {  this.name = name;  this.address = address;  this.phoneNumber = phoneNumber;  this.emailAddress = emailAddress;  }  public String getName() {  return name;  }  public String getAddress() {  return address;  }  public long getPhoneNumber() {  return phoneNumber;  }  public String getEmailAddress() {  return emailAddress;  }    @Override  public String toString() {  return "Person name from "+this.getClass().getName() +" class is " + name ;  }  } |

Staff

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  import java.util.Date;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Staff extends Employee {  private String title;  public Staff(String title, String office, double salary,  Date dateHired, String name, String address,  long phoneNumber, String emailAddress) {  super(office, salary, dateHired, name, address,  phoneNumber, emailAddress);  this.title = title;  }  @Override  public String toString() {  return "Person name from " + this.getClass().getName()  + " is " + super.getName();  }  } |

Student

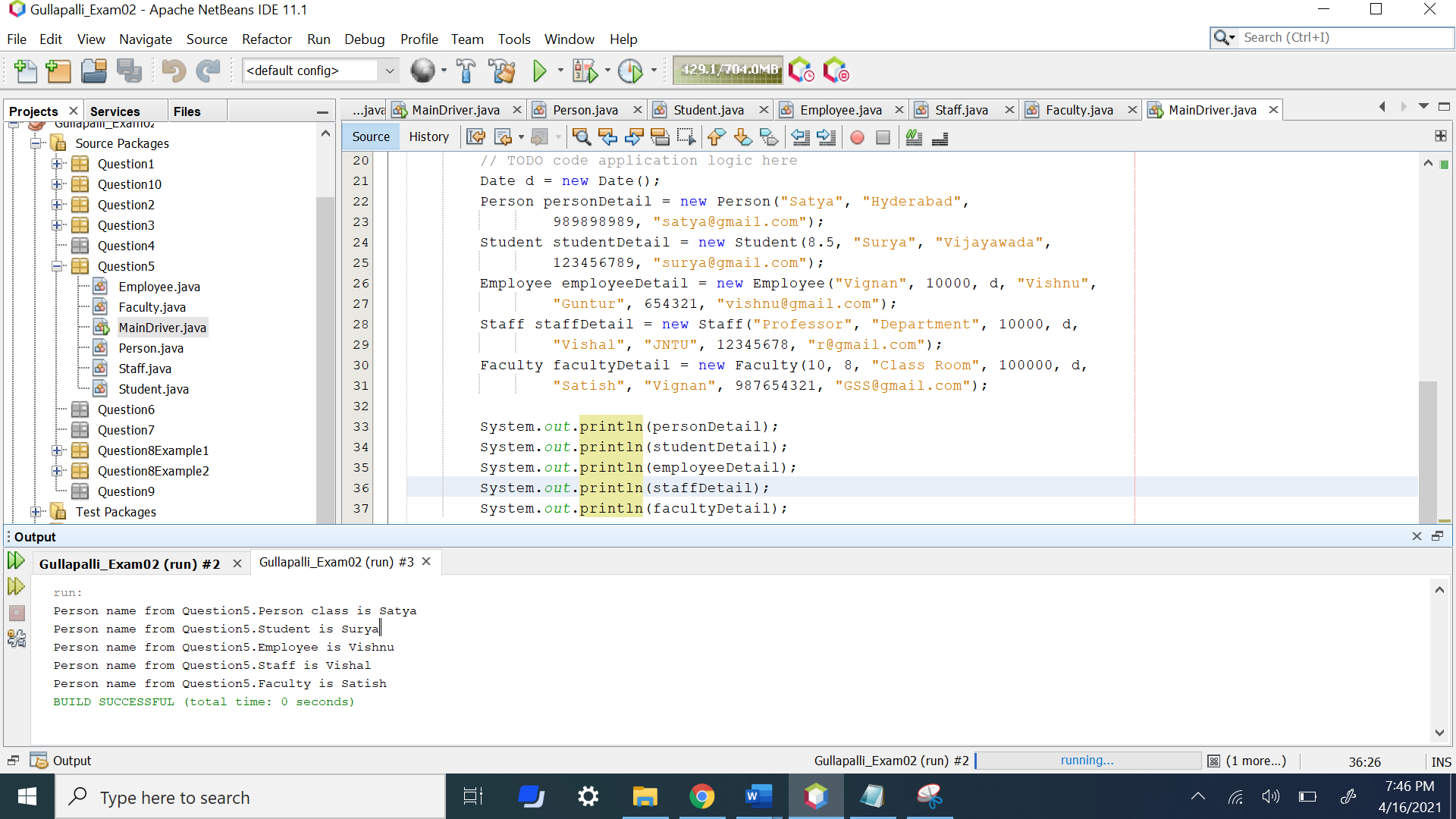
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Student extends Person {  private double grade;  public static final String CLASS\_STATUS = "Graduate";  public Student(double grade, String name, String address,  long phoneNumber, String emailAddress) {  super(name, address, phoneNumber, emailAddress);  this.grade = grade;  }  @Override  public String toString() {  return "Person name from " + this.getClass().getName() + " is "  + super.getName();  }  } |

MainDriver

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  import java.util.Date;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class MainDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Date d = new Date();  Person personDetail = new Person("Satya", "Hyderabad",  989898989, "satya@gmail.com");  Student studentDetail = new Student(8.5, "Surya", "Vijayawada",  123456789, "surya@gmail.com");  Employee employeeDetail = new Employee("Vignan", 10000, d, "Vishnu",  "Guntur", 654321, "vishnu@gmail.com");  Staff staffDetail = new Staff("Professor", "Department", 10000, d,  "Vishal", "JNTU", 12345678, "r@gmail.com");  Faculty facultyDetail = new Faculty(10, 8, "Class Room", 100000, d,  "Satish", "Vignan", 987654321, "GSS@gmail.com");  System.out.println(personDetail);  System.out.println(studentDetail);  System.out.println(employeeDetail);  System.out.println(staffDetail);  System.out.println(facultyDetail);  }  } |

Output

|  |
| --- |
| Person name from Question5.Person class is Satya  Person name from Question5.Student is Surya  Person name from Question5.Employee is Vishnu  Person name from Question5.Staff is Vishal  Person name from Question5.Faculty is Satish |



1. (10-Points) Design a new **Triangle** class that extends the abstract **GeometricObject** class. Draw the UML diagram for the classes **Triangle** and **GeometricObject** and then implement the **Triangle** class. Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled. The program should create a **Triangle** object with these sides and set the color and filled properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not. Provide screenshot of executable code with input and output.

GeometricObject.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question6;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public abstract class GeometricObject {  private String color = "while";  private boolean filled;  private java.util.Date dateCreated;  /\*\* Construct a default geometric object \*/  protected GeometricObject() {  dateCreated = new java.util.Date();  }  /\*\* Construct a geometric object with color and filled value \*/  protected GeometricObject(String color, boolean filled) {  dateCreated = new java.util.Date();  this.color = color;  this.filled = filled;  }  /\*\* Return color \*/  public String getColor() {  return color;  }  /\*\* Set a new color \*/  public void setColor(String color) {  this.color = color;  }  /\*\* Return filled. Since filled is boolean,  \* the get method is named isFilled \*/  public boolean isFilled() {  return filled;  }  /\*\* Set a new filled \*/  public void setFilled(boolean filled) {  this.filled = filled;  }  /\*\* Get dateCreated \*/  public java.util.Date getDateCreated() {  return dateCreated;  }  @Override  public String toString() {  return "created on " + dateCreated + "\ncolor: " + color +  " and filled: " + filled;  }  /\*\* Abstract method getArea \*/  public abstract double getArea();  /\*\* Abstract method getPerimeter \*/  public abstract double getPerimeter();  } |

Triangle.java

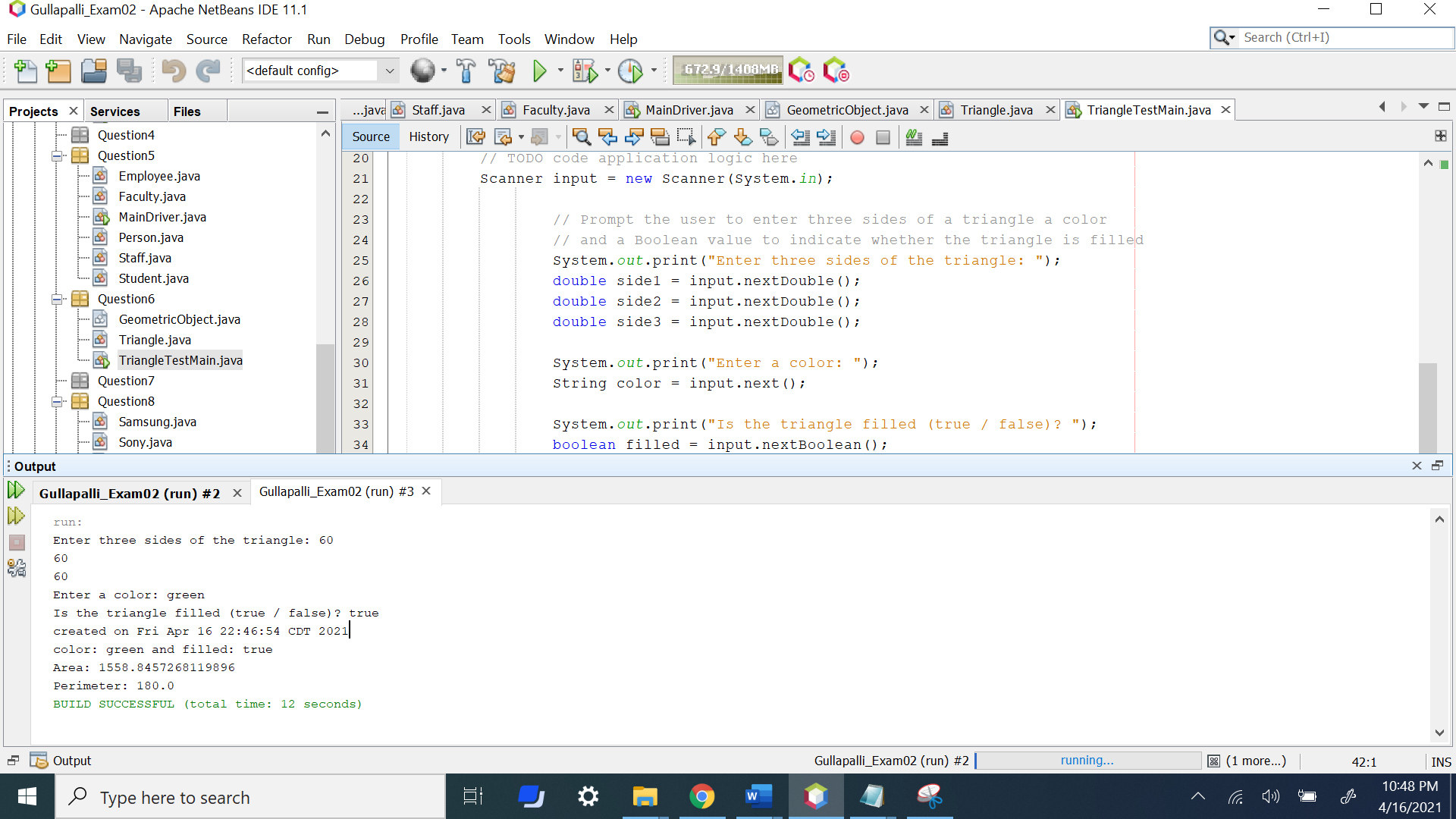
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question6;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Triangle extends GeometricObject {  private double side1;  private double side2;  private double side3;  public Triangle(){  }  public Triangle(double side1, double side2, double side3) {  this.side1 = side1;  this.side2 = side2;  this.side3 = side3;  }  public Triangle(double side1, double side2, double side3,  String color, boolean filled) {  this(side1, side2, side3);  setColor(color);  setFilled(filled);  }  /\*\* Return side1 \*/  public double getSide1() {  return side1;  }  /\*\* Set side1 to a new length \*/  public void setSide1(double side1) {  this.side1 = side1;  }  /\*\* Return side2 \*/  public double getSide2() {  return side2;  }  /\*\* Set side2 to a new lentgh \*/  public void setSide2(double side2) {  this.side2 = side2;  }  /\*\* Return side3 \*/  public double getSide3() {  return side3;  }  /\*\* Set side3 to a new length \*/  public void setSide3(double side3) {  this.side3 = side3;  }  @Override /\*\* Return area of this Triangle \*/  public double getArea() {  double s = (side1 + side2 + side3) / 2;  return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));  }  @Override /\*\* Return perimeter of this triangle \*/  public double getPerimeter() {  return side1 + side2 + side3;  }  @Override /\*\* Return a string description of the object \*/  public String toString() {  return super.toString() + "\nArea: " + getArea() +  "\nPerimeter: " + getPerimeter();  }  } |

TriangleTestMain.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question6;  import java.util.Scanner;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class TriangleTestMain {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner input = new Scanner(System.in);  // Prompt the user to enter three sides of a triangle a color  // and a Boolean value to indicate whether the triangle is filled  System.out.print("Enter three sides of the triangle: ");  double side1 = input.nextDouble();  double side2 = input.nextDouble();  double side3 = input.nextDouble();  System.out.print("Enter a color: ");  String color = input.next();  System.out.print("Is the triangle filled (true / false)? ");  boolean filled = input.nextBoolean();  // Create a Triangle  Triangle = new Triangle(side1, side2, side3, color, filled);  System.out.println(triangle);  }  } |

Output

|  |
| --- |
| Enter three sides of the triangle: 60  60  60  Enter a color: green  Is the triangle filled (true / false)? true  created on Fri Apr 16 22:46:54 CDT 2021  color: green and filled: true  Area: 1558.8457268119896  Perimeter: 180.0 |



1. (10-Points) What is an Enum in Java? Explain and demonstrate with some examples. Provide executable code screenshots for examples.

Enum in Java:

Enum in Java, it is a data type which contains a fixed set of constants. Enums are used to create our own data type like classes Java Enum internally inherits the Enum class, so it cannot inherit any other class, but it can implement many interfaces.

Enum Season

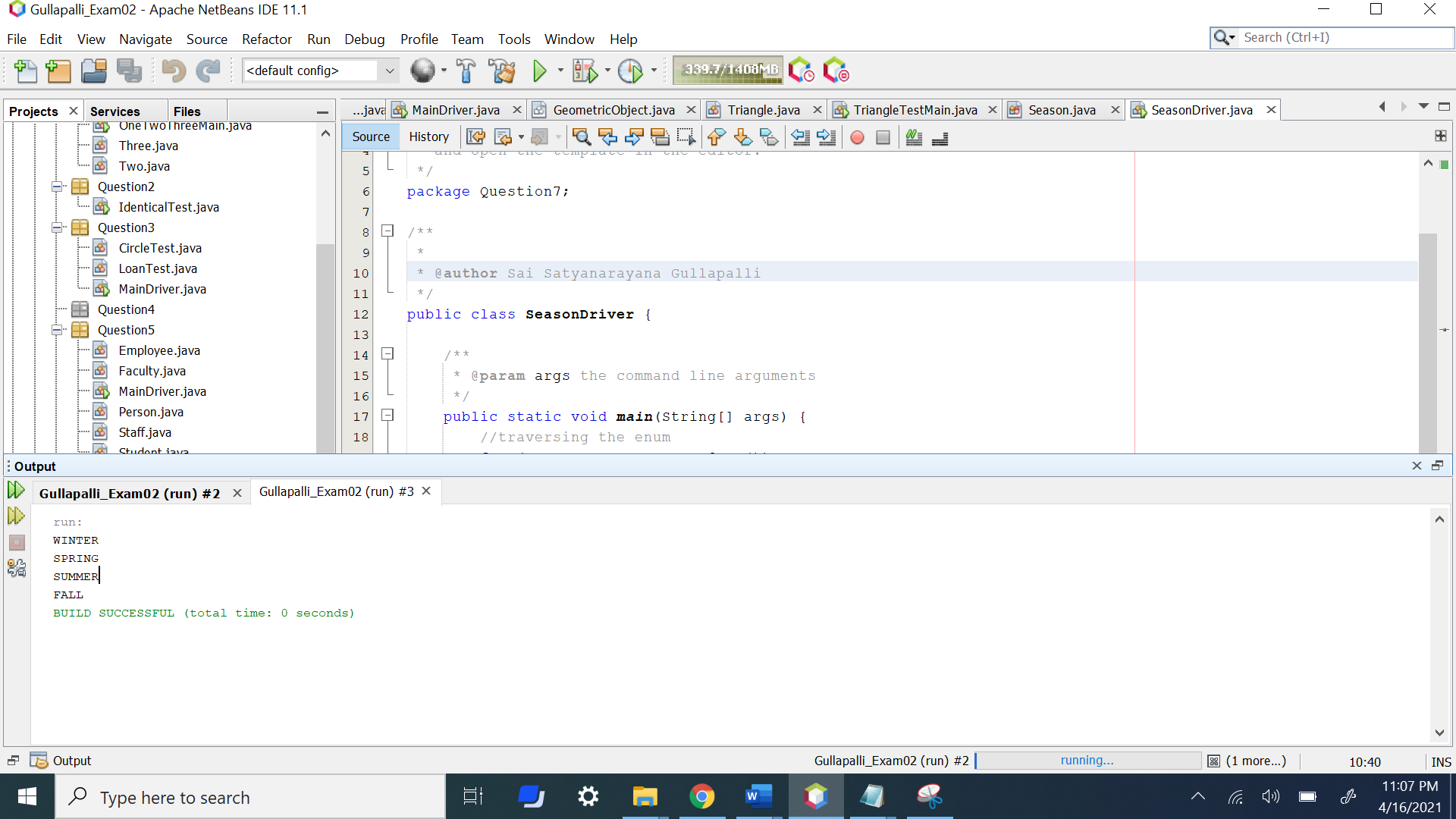
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question7;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public enum Season {  WINTER, SPRING, SUMMER, FALL  } |

SeasonDriver

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question7;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class SeasonDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  //traversing the enum  for (Season s : Season.values())  System.out.println(s);  }} |

Output

|  |
| --- |
| WINTER  SPRING  SUMMER  FALL |



1. (10-points) Define the term abstract class in java? Explain and demonstrate with some examples. Provide executable code screenshots for examples.

8(A) Abstract class in java:

An abstract class is a class that is declared abstract — it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

A class which is declared with the abstract keyword is known as an abstract class in Java.

Two types:

1. abstract methods

2. non-abstract methods

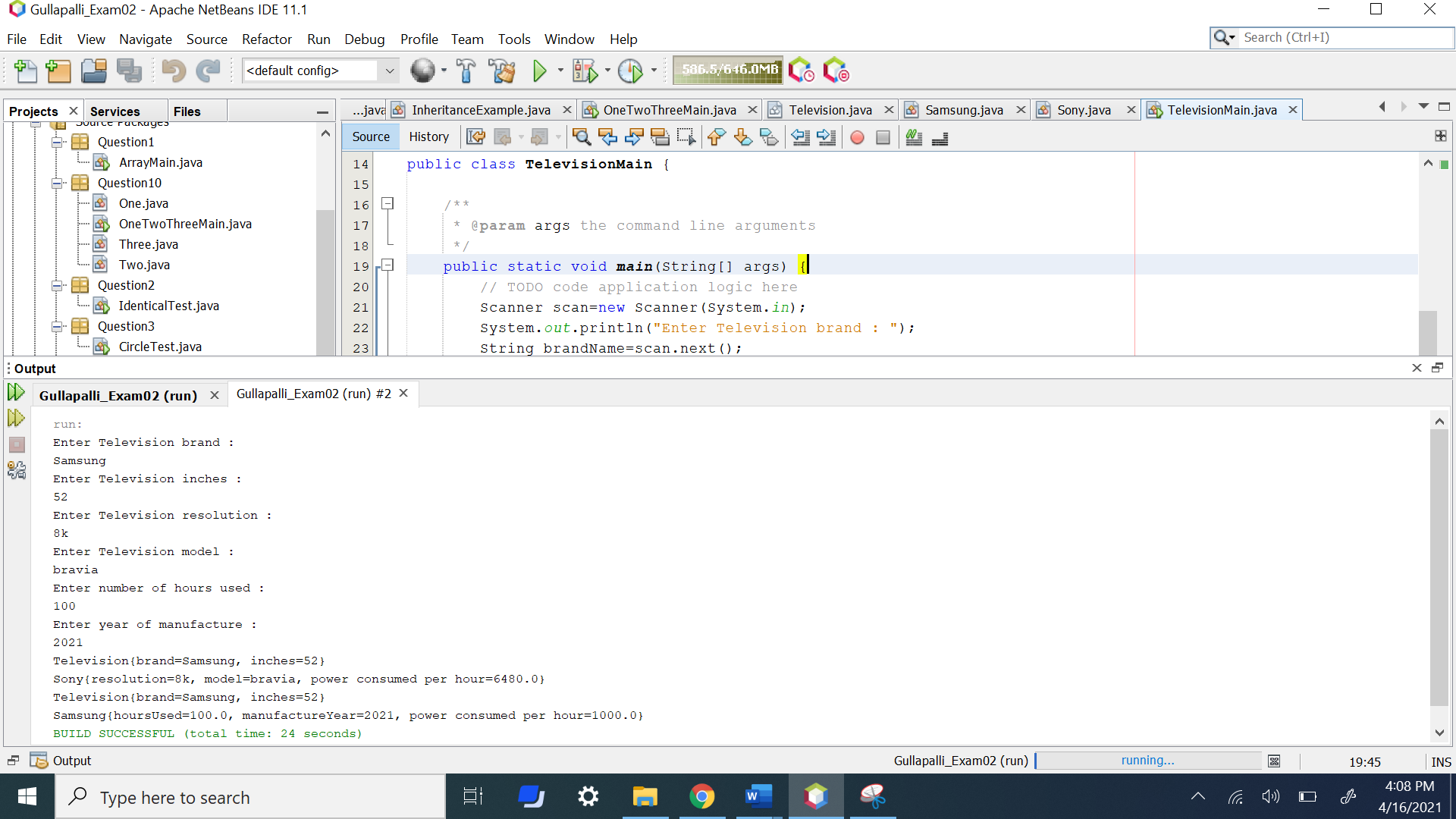
8(A) Example 1

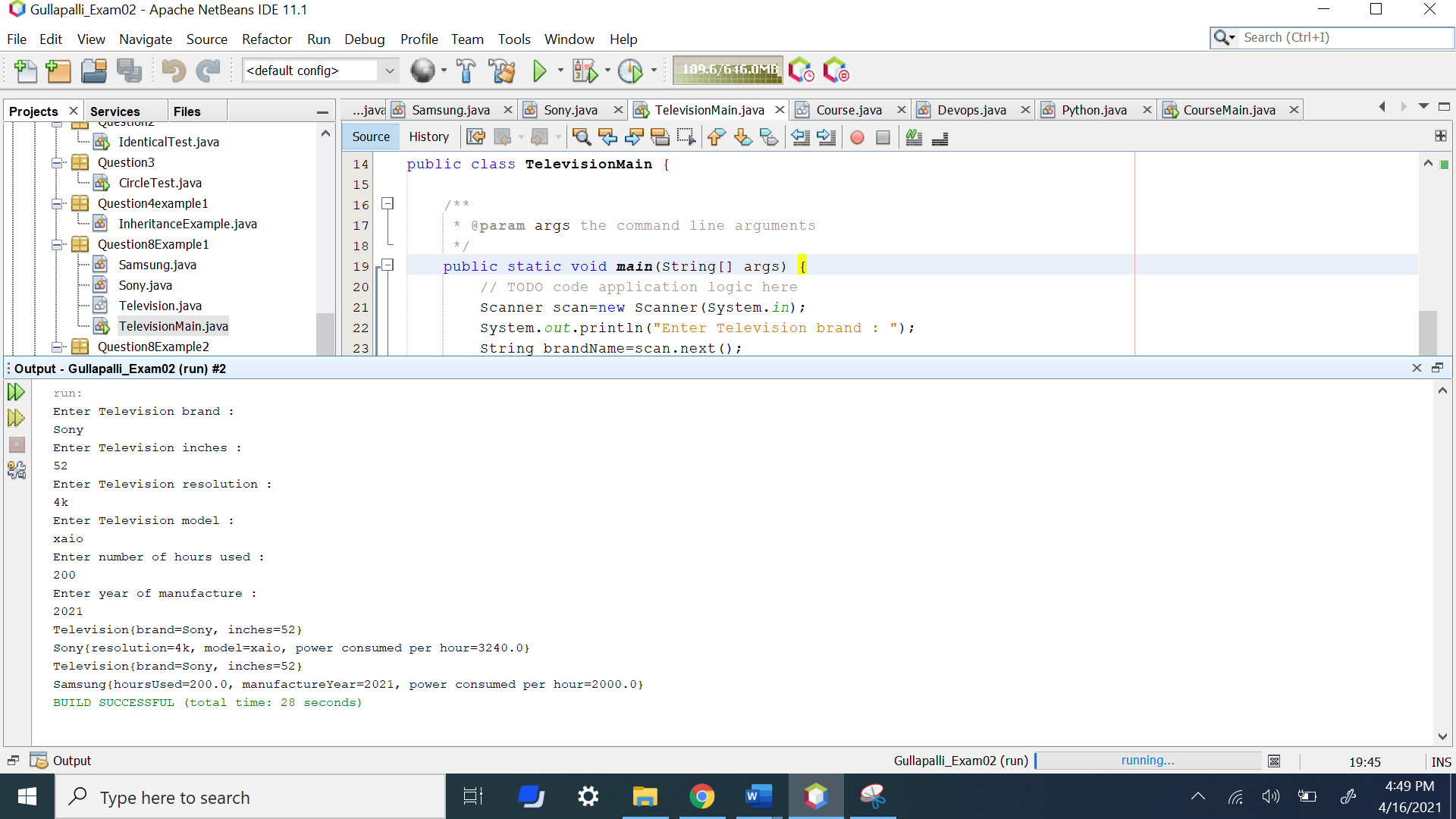
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question8Example1;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public abstract class Television {  private String brand;  private int inches;  public Television(String brand, int inches) {  this.brand = brand;  this.inches = inches;  }  public String getBrand() {  return brand;  }  public void setBrand(String brand) {  this.brand = brand;  }  public int getInches() {  return inches;  }  public void setInches(int inches) {  this.inches = inches;  }  @Override  public String toString() {  return "Television{" + "brand=" + brand + ", inches=" + inches + '}';  }  public abstract double powerconsumption();      }  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question8Example1;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Samsung extends Television {  private double hoursUsed;  private int manufactureYear;  public Samsung(double hoursUsed, int manufactureYear, String brand, int inches) {  super(brand, inches);  this.hoursUsed = hoursUsed;  this.manufactureYear = manufactureYear;  }  public double getHoursUsed() {  return hoursUsed;  }  public void setHoursUsed(double hoursUsed) {  this.hoursUsed = hoursUsed;  }  public int getManufactureYear() {  return manufactureYear;  }  public void setManufactureYear(int manufactureYear) {  this.manufactureYear = manufactureYear;  }  public double powerconsumption(){  double consumed=hoursUsed\*10;  return consumed;  }  @Override  public String toString() {  return super.toString()+"\nSamsung{" + "hoursUsed=" + hoursUsed + ", manufactureYear=" + manufactureYear + ", power consumed per hour="+powerconsumption()+'}';  }    }  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question8Example1;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class Sony extends Television{  private String resolution;  private String model;  public Sony(String resolution, String model, String brand, int inches) {  super(brand, inches);  this.resolution = resolution;  this.model = model;  }  public String getResolution() {  return resolution;  }  public void setResolution(String resolution) {  this.resolution = resolution;  }  public String getModel() {  return model;  }  public void setModel(String model) {  this.model = model;  }  public double powerconsumption(){  double consumed=0.0;  if(resolution.equalsIgnoreCase("4k")){  consumed= 1080\*3;  }  else if(resolution.equalsIgnoreCase("8k")){  consumed=2160\*3;  }  return consumed;    }  @Override  public String toString() {  return super.toString()+"\nSony{" + "resolution=" + resolution + ", model=" + model + ", power consumed per hour="+powerconsumption()+ '}';  }      }  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question8Example1;  import java.util.Scanner;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class TelevisionMain {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner scan=new Scanner(System.in);  System.out.println("Enter Television brand : ");  String brandName=scan.next();  System.out.println("Enter Television inches : ");  int size=scan.nextInt();  System.out.println("Enter Television resolution : ");  String screenResolution=scan.next();  System.out.println("Enter Television model : ");  String tvModel=scan.next();  System.out.println("Enter number of hours used : ");  double usedHours=scan.nextDouble();  System.out.println("Enter year of manufacture : ");  int year = scan.nextInt();  Sony sonyTv =new Sony(screenResolution,tvModel,brandName,size);  Samsung samsungTv = new Samsung(usedHours,year,brandName,size);  System.out.println(sonyTv.toString());  System.out.println(samsungTv.toString());    }    } |

Output :

|  |
| --- |
| Enter Television brand :  Samsung  Enter Television inches :  52  Enter Television resolution :  8k  Enter Television model :  bravia  Enter number of hours used :  100  Enter year of manufacture :  2021  Television{brand=Samsung, inches=52}  Sony{resolution=8k, model=bravia, power consumed per hour=6480.0}  Television{brand=Samsung, inches=52}  Samsung{hoursUsed=100.0, manufactureYear=2021, power consumed per hour=1000.0}  Enter Television brand :  Sony  Enter Television inches :  52  Enter Television resolution :  4k  Enter Television model :  xaio  Enter number of hours used :  200  Enter year of manufacture :  2021  Television{brand=Sony, inches=52}  Sony{resolution=4k, model=xaio, power consumed per hour=3240.0}  Television{brand=Sony, inches=52}  Samsung{hoursUsed=200.0, manufactureYear=2021, power consumed per hour=2000.0} |

|  |
| --- |
|  |
|  |





1. (10-points) Define the term interface in java? Explain and demonstrate with some examples. Provide executable code screenshots for examples.

9(A) Interface: It is a reference type in Java and it is similar to class.

It contains static constants and abstract methods.   
Syntax:

interface <interface\_name>{

}

Readable.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question9;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public interface Readable {  public abstract void read();  } |

TextMessage.java

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question9;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class TextMessage implements Readable {    @Override  public void read() {  System.err.println("Read implemented");  }  } |

InterfaceDriver.java

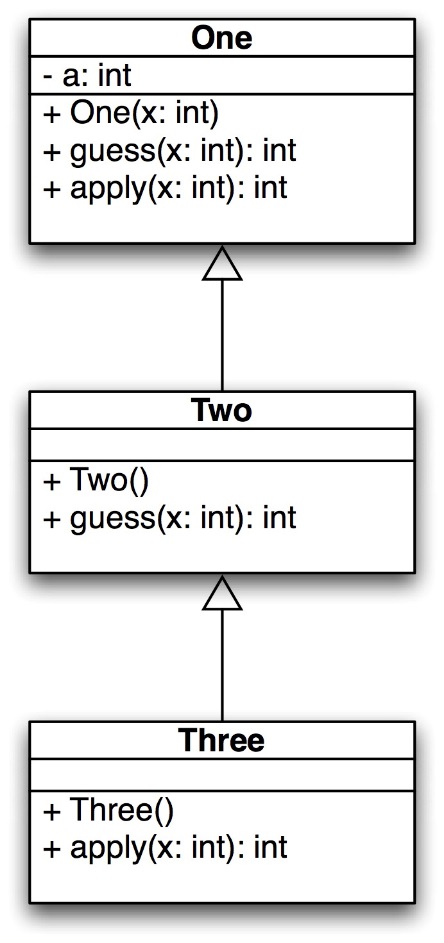
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question9;  /\*\*  \*  \* @author Sai Satyanarayana Gullapalli  \*/  public class InterfaceDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here    TextMessage tt = new TextMessage();  tt.read();  }    } |

Output

|  |
| --- |
| Read implemented |

1. (15-Points) Consider the following code for three classes One, Two, and Three. (A UML diagram is included for your convenience.)

|  |
| --- |
| public class One {  private int a;  public One(int in){  a = in;  }  public int guess (int x){  System.out.println("One guess " + x);  return a + x;  }    public int apply (int x){  System.out.println("One guess " + x);  return guess(x + 3);  }  } // end class One  public class Two extends One {  public Two(){  super(11);  }  public int guess(int x){  System.out.println("Two guess " + x);  return super.guess(x)+10;  }    } // end class Two  public class Three extends Two {  public int apply(int x){  System.out.println("Three apply " + x);  return -10;  }  } // end class Three |



What is the output of the following code? Explain it.

|  |
| --- |
| public static void main(String[] args) {  One hippo = new Three();  System.out.println(hippo.guess(4));  System.out.println(hippo.apply(12));  One lion = new One(-1);  System.out.println(lion.guess(5));  System.out.println(lion.apply(6));  } // end |

10(A) Polymorphism in Java is a concept by which we can perform a single action in different ways. So polymorphism means many forms. There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

Output :

|  |
| --- |
| Two guess 4  One guess 4  25  Three apply 12  -10  One guess 5  4  One guess 6  One guess 9  8 |

|  |
| --- |
|  |

