**Exam 2 OBJECT-ORIENTED PROG 01FA20 75 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 |

|  |
| --- |
| Class RemoveDuplicates  package karepe\_exam2.question1;  import java.util.ArrayList;  import java.util.Scanner;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class RemoveDuplicates {  /\*\*  \*  \*/  public static ArrayList<Integer> numbers = new ArrayList<>();  /\*\*  \*  \* @param args  \*/  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.print("Enter ten integers: ");  int i = 0;  //Adding values to ArrayList  while (i < 10) {  numbers.add(scan.nextInt());  i++;  }  removeDuplicate(numbers);  System.out.print("The distinct integers are");  for (int num : numbers) {  System.out.print(" " + num);  }  scan.close();  }  /\*\*  \*  \* @param list  \*/  public static void removeDuplicate(ArrayList<Integer> list) {  // Create a new ArrayList  ArrayList<Integer> updatedList = new ArrayList<Integer>();  // Loop through the original list  for (int i : list) {  // If this element is not present in updatedList then add it  if (!updatedList.contains(i)) {  updatedList.add(i);  }  }  numbers = updatedList;  }  **Output**    } |

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 |

|  |
| --- |
| Class Compare2DArray  package karepe\_exam2.question2;  import java.util.Scanner;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Compare2DArray {  /\*\*  \*  \* @param args  \*/  public static void main(String[] args) {  int[][] m1 = new int[3][3];  int[][] m2 = new int[3][3];  Scanner scan = new Scanner(System.in);  System.out.print("Enter list1: ");  for (int i = 0; i < m1.length; i++) {  for (int j = 0; j < m1[0].length; j++) {  m1[i][j] = scan.nextInt();  }  }  System.out.print("Enter list2: ");  for (int i = 0; i < m2.length; i++) {  for (int j = 0; j < m2[0].length; j++) {  m2[i][j] = scan.nextInt();  }  }  scan.close();  if (equals(m1, m2)) {  System.out.println("The two arrays are strictly identical");  } else {  System.out.println("The two arrays are not strictly identical");  }  }  /\*\*  \*  \* @param m1  \* @param m2  \* @return  \*/  public static boolean equals(int[][] m1, int[][] m2) {  if (m1.length != m2.length || m1[0].length != m2[0].length) {  return false;  } else {  for (int i = 0; i < m1.length; i++) {  for (int j = 0; j < m1[0].length; j++) {  if (m1[i][j] != m2[i][j]) {  return false;  }  }  }  }  return true;  }  }  Output : 1    Output : 2 |

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

|  |
| --- |
| Circle Class  package karepe\_exam2.question3;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Circle {  double radius;  /\*\*  \* @return the radius  \*/  public double getRadius() {  return radius;  }  /\*\*  \* @param radius the radius to set  \*/  public void setRadius(double radius) {  this.radius = radius;  }  /\*\*  \* @param radius  \*/  public Circle(double radius) {  this.radius = radius;  }  /\*  \* (non-Javadoc)  \*  \* @see java.lang.Object#toString()  \*/  @Override  public String toString() {  return String.format("Circle :\n" + " Radius = %.2fcms\n" + " Area = %.2fcm\u00b2\n"  + " Circumference = %.2fcms\n", radius, Math.PI \* radius \* radius, 2 \* Math.PI \* radius);  }  }  Loan Class  package karepe\_exam2.question3;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Loan {  double principal;  int time;  double rate;  /\*\*  \* @param principal  \* @param time  \* @param rate  \*/  public Loan(double principal, int time, double rate) {  this.principal = principal;  this.time = time;  this.rate = rate;  }  /\*\*  \* @return the principal  \*/  public double getPrincipal() {  return principal;  }  /\*\*  \* @param principal the principal to set  \*/  public void setPrincipal(double principal) {  this.principal = principal;  }  /\*\*  \* @return the time  \*/  public double getTime() {  return time;  }  /\*\*  \* @param time the time to set  \*/  public void setTime(int time) {  this.time = time;  }  /\*\*  \* @return the rate  \*/  public double getRate() {  return rate;  }  /\*\*  \* @param rate the rate to set  \*/  public void setRate(double rate) {  this.rate = rate;  }  /\*\*  \* @return the interest  \*/  public double calcInterest() {  return principal \* time \* rate / 100;  }  @Override  public String toString() {  return "Loan :\n" + " Amount = $" + principal + "\n Time = " + time  + "months\n Rate = " + rate + "%\n Interest = $" + calcInterest();  }  }  Driver Class  package karepe\_exam2.question3;  import java.util.ArrayList;  import java.util.Calendar;  import java.util.Date;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Driver {  /\*\*  \*  \* @param args  \*/  public static void main(String[] args) {  ArrayList<Object> list = new ArrayList<>();  Loan loan = new Loan(100, 6, 5);  list.add(loan);  Date date = Calendar.getInstance().getTime();  list.add(date);  String name = "Nithya Karepe";  list.add(name);  Circle circle = new Circle(5);  list.add(circle);  for (Object o : list) {  System.out.println(o.toString());  }  }  }  Output : |

1. (10-Points) What is Inheritance, Polymorphism and Late binding polymorphism? Explain and demonstrate with examples. Provide executable code screenshots for examples.

|  |
| --- |
| Inheritance   1. Inheritance is a process where one class acquires the properties (methods and fields) of another. 2. Extends is the keyword used to inherit the properties of a super class. 3. The super is a parent class where the other class which inherit the properties of parent class are called sub classes. 4. Syntax   class Super {  .....  .....  }  class Sub extends Super {  .....  .....  }  Here super class is the parent class, sub class inherits the properties of super class.  Polymorphism   1. Polymorphism is the capability of an object to take many forms. 2. The use of polymorphism in OOP is when a parent class reference is used to refer to a child class object. 3. Any Java object that can pass more than one IS-A test is polymorphic.   Late Binding Polymorphism   1. In this, the compiler doesn’t decide the method to be called. Overriding is an example of dynamic binding. 2. In overriding both parent and child classes have the same method. 3. The ability to override methods coupled with the run-time determination of which method to invoke is referred to as late binding polymorphism.   Examples  Gadget  package karepe\_exam2.question4;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Gadget {  public String brand;  public double price;  /\*\*  \* @return the brand  \*/  public String getBrand() {  return brand;  }  /\*\*  \* @param brand the brand to set  \*/  public void setBrand(String brand) {  this.brand = brand;  }  /\*\*  \* @return the price  \*/  public double getPrice() {  return price;  }  /\*\*  \* @param price the price to set  \*/  public void setPrice(double price) {  this.price = price;  }  /\*\*  \* @return the category  \*/  public String getCategory() {  return null;  }  /\*\*  \* @param brand  \* @param price  \*/  public Gadget(String brand, double price) {  super();  this.brand = brand;  this.price = price;  }  /\*\*  \*  \*/  public Gadget() {  }  @Override  public String toString() {  return "Brand= " + brand + "\nPrice= " + price + "\nCategory: " + getCategory();  }  } Watch  package karepe\_exam2.question4;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Watch extends Gadget {  boolean isSmartWatch = false;    public String color;  public String material;  public boolean isSmartWatch() {  if (super.getBrand().equalsIgnoreCase("apple")) {  setSmartWatch(true);  return true;  }  return isSmartWatch;  }  /\*\*  \* @param isSmartWatch the isSmartWatch to set  \*/  public void setSmartWatch(boolean isSmartWatch) {  this.isSmartWatch = isSmartWatch;  }  /\*\*  \* @return the color  \*/  public String getColor() {  return color;  }  /\*\*  \* @param color the color to set  \*/  public void setColor(String color) {  this.color = color;  }  /\*\*  \* @return the material  \*/  public String getMaterial() {  return material;  }  /\*\*  \* @param material the material to set  \*/  public void setMaterial(String material) {  this.material = material;  }  /\*\*  \* @param brand  \* @param price  \* @param color  \* @param material  \*/  public Watch(String brand, double price, String color, String material) {  super(brand, price);  this.color = color;  this.material = material;  }  /\*\*  \* @param price  \* @param color  \* @param material  \*/  public Watch(String color, String material, double price) {  super.price = price;  this.color = color;  this.material = material;  }  /\*\*  \*  \* @return  \*/  @Override  public String getCategory() {  return "Watch";  }  @Override  public String toString() {  return super.toString() + "\nMaterial: " + getMaterial() + "\nisSmartWatch: " + isSmartWatch() + "\nColor: "  + getColor();  }  }  Laptop  package karepe\_exam2.question4;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Laptop extends Gadget {  /\*\*  \*  \*/  public int screenSize;  /\*\*  \*  \*/  public String processor;  /\*\*  \* @return the screenSize  \*/  public int getScreenSize() {  return screenSize;  }  /\*\*  \* @param screenSize the screenSize to set  \*/  public void setScreenSize(int screenSize) {  this.screenSize = screenSize;  }  /\*\*  \* @return the processor  \*/  public String getProcessor() {  return processor;  }  /\*\*  \* @param processor the processor to set  \*/  public void setProcessor(String processor) {  this.processor = processor;  }  /\*\*  \* @param brand  \* @param price  \* @param screenSize  \* @param processor  \*/  public Laptop(String brand, double price, int screenSize, String processor) {  super(brand, price);  this.screenSize = screenSize;  this.processor = processor;  }  /\*\*  \*  \* @return  \*/  @Override  public String getCategory() {  return "Laptop";  }  @Override  public String toString() {  return super.toString() + "\nScreenSize: " + getScreenSize() + " inches\n" + "Processor: " + getProcessor();  }  }  AppleWatch  package karepe\_exam2.question4;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class AppleWatch extends Watch {  /\*\*  \*  \* @param price  \* @param color  \* @param material  \*/  public AppleWatch(double price, String color, String material) {  super("Apple", price, color, material);  }  }  GadgetDriver  package karepe\_exam2.question4;  import java.util.ArrayList;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class GadgetDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Gadget g1 = new Watch("Fossil", 225.00, "Red", "Stainless Steel");  Gadget g2 = new AppleWatch(425.00, "Black", "Stainless Steel");  Gadget g3 = new Laptop("Dell", 1345.00, 13, "Intel Core i5");  ArrayList<Gadget> list = new ArrayList<>();  list.add(g1);  list.add(g2);  list.add(g3);  for (Gadget g : list) {  System.out.println("\n" + g.toString());  }  }  }  Output :    In our example we make use of the below classes  Gadget  Watch  Laptop  AppleWatch  Here both Watch & Laptop extend the class Gadget or we can say Watch & Laptop classes both inherit from class Gadget.  By default all objects inherit the Object class in java.  AppleWatch extends the class Watch or we can say AppleWatch class inherits from class Watch.  Using Inheritance methods from the super class can be accessed by sub class objects. Hence visible methods or variables from Gadget class can be accessed by all the below classes  Watch  Laptop  The above is an example for hierarchical inheritance.  Same for visible methods or variables from Watch class can be accessed by AppleWatch class.  In the above case AppleWatch extends Watch & Watch extends Gadget so AppleWatch can also access visible methods or variables from Gadget class. This is known as Mutlilevel inheritance.  We use super keyword to access parent variables or methods.  Within Laptop Constructor we use super(brand, price) to set values for brand and price and create object.  Within Watch constructor we use super.price = price to set price  Within AppleWatch we use super("Apple", price, color, material)to set brand as Apple and other corresponding values for Watch  Polymorphism:  We use polymorphism to use inherited methods differently across different classes  For Example:  We override getCategory method from Gadget class to return Watch from Watch class and its sub classes and the same method return  Laptop from Laptop class.  This allows us to customize methods and functionality.  We also override the toString methods in Watch and Laptop classes to override the default toString from Object class and also the parent Gadget class.  Polymorphism can be both compile time and late binding polymorphism.  Methods having same name but different number of arguments or method signatures fall under compile time polymorphism.  Overriding inherited methods at run time is call late binding polymorphism  In our GadgetDriver class we use Polymorphic substitution while adding objects of Watch, Apple Watch and Laptop all into an ArrayList<Gadget>  While printing all of these objects the toString method follows late binding polymorphism. |

1. (15-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.

|  |
| --- |
| Person Class  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Person {  public String name;  public String address;  public String phoneNumber;  public String emailAddress;  /\*\*  \* @param name  \* @param address  \* @param phoneNumber  \* @param emailAddress  \*/  public Person(String name, String address, String phoneNumber, String emailAddress) {  super();  this.name = name;  this.address = address;  this.phoneNumber = phoneNumber;  this.emailAddress = emailAddress;  }  /\*\*  \* @return the name  \*/  public String getName() {  return name;  }  /\*\*  \* @param name the name to set  \*/  public void setName(String name) {  this.name = name;  }  /\*\*  \* @return the address  \*/  public String getAddress() {  return address;  }  /\*\*  \* @param address the address to set  \*/  public void setAddress(String address) {  this.address = address;  }  /\*\*  \* @return the phoneNumber  \*/  public String getPhoneNumber() {  return phoneNumber;  }  /\*\*  \* @param phoneNumber the phoneNumber to set  \*/  public void setPhoneNumber(String phoneNumber) {  this.phoneNumber = phoneNumber;  }  /\*\*  \* @return the emailAddress  \*/  public String getEmailAddress() {  return emailAddress;  }  /\*\*  \* @param emailAddress the emailAddress to set  \*/  public void setEmailAddress(String emailAddress) {  this.emailAddress = emailAddress;  }  @Override  public String toString() {  return String.format("Name: %s\nAddress: %s\nPhoneNumber: %s\nEmailAddress: %s\n", getName(), getAddress(),  getPhoneNumber(), getEmailAddress());  }  }  Employee Class  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Employee extends Person {  public String office;  public double salary;  public String date\_hired;  /\*\*  \* @return the office  \*/  public String getOffice() {  return office;  }  /\*\*  \* @param office the office to set  \*/  public void setOffice(String office) {  this.office = office;  }  /\*\*  \* @return the salary  \*/  public double getSalary() {  return salary;  }  /\*\*  \* @param salary the salary to set  \*/  public void setSalary(double salary) {  this.salary = salary;  }  /\*\*  \* @return the date\_hired  \*/  public String getDate\_hired() {  return date\_hired;  }  /\*\*  \* @param date\_hired the date\_hired to set  \*/  public void setDate\_hired(String date\_hired) {  this.date\_hired = date\_hired;  }  /\*\*  \* @param name  \* @param address  \* @param phoneNumber  \* @param emailAddress  \* @param office  \* @param salary  \* @param date\_hired  \*/  public Employee(String name, String address, String phoneNumber, String emailAddress, String office, double salary, String date\_hired) {  super(name, address, phoneNumber, emailAddress);  this.office = office;  this.salary = salary;  this.date\_hired = date\_hired;  }  /\*\*  \* @param office  \* @param salary  \* @param date\_hired  \* @param person  \*/  public Employee(String office, double salary, String date\_hired, Person person) {  super(person.getName(), person.getAddress(), person.getPhoneNumber(), person.getEmailAddress());  this.office = office;  this.salary = salary;  this.date\_hired = date\_hired;  }  @Override  public String toString() {  return super.toString() + String.format("Office: %s\nSalary: %.2f\nDate\_hired: %s\n", getOffice(), getSalary(),  getDate\_hired());  }  }  Student Class  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Student extends Person {  public double grade;  public static final String STATUS = "Graduate";  /\*\*  \* @param name  \* @param address  \* @param phoneNumber  \* @param emailAddress  \* @param grade  \*/  public Student(String name, String address, String phoneNumber, String emailAddress, double grade) {  super(name, address, phoneNumber, emailAddress);  this.grade = grade;  }  /\*\*  \* @return the grade  \*/  public double getGrade() {  return grade;  }  /\*\*  \* @param grade the grade to set  \*/  public void setGrade(double grade) {  this.grade = grade;  }  @Override  public String toString() {  return super.toString() + String.format("Grade: %s\nStatus: %s\n", getGrade(), STATUS);  }  }  Faculty Class  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Faculty extends Employee {  public String officeHours;  public int numberOfTeachingSubjects;  /\*\*  \* @return the officeHours  \*/  public String getOfficeHours() {  return officeHours;  }  /\*\*  \* @param officeHours the officeHours to set  \*/  public void setOfficeHours(String officeHours) {  this.officeHours = officeHours;  }  /\*\*  \* @return the numberOfTeachingSubjects  \*/  public int getNumberOfTeachingSubjects() {  return numberOfTeachingSubjects;  }  /\*\*  \* @param numberOfTeachingSubjects the numberOfTeachingSubjects to set  \*/  public void setNumberOfTeachingSubjects(int numberOfTeachingSubjects) {  this.numberOfTeachingSubjects = numberOfTeachingSubjects;  }  /\*\*  \* @param name  \* @param address  \* @param phoneNumber  \* @param emailAddress  \* @param office  \* @param salary  \* @param date\_hired  \* @param officeHours  \* @param numberOfTeachingSubjects  \*/  public Faculty(String name, String address, String phoneNumber, String emailAddress, String office, double salary,  String date\_hired, String officeHours, int numberOfTeachingSubjects) {  super(name, address, phoneNumber, emailAddress, office, salary, date\_hired);  this.officeHours = officeHours;  this.numberOfTeachingSubjects = numberOfTeachingSubjects;  }  @Override  public String toString() {  return super.toString() + String.format("OfficeHours: %s\nNumberOfTeachingSubjects: %d\n", getOfficeHours(),  getNumberOfTeachingSubjects());  }  }  Staff Class  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Staff extends Employee {  public String title;  /\*\*  \* @return the title  \*/  public String getTitle() {  return title;  }  /\*\*  \* @param title the title to set  \*/  public void setTitle(String title) {  this.title = title;  }  /\*\*  \* @param name  \* @param address  \* @param phoneNumber  \* @param emailAddress  \* @param office  \* @param salary  \* @param date\_hired  \* @param title  \*/  public Staff(String name, String address, String phoneNumber, String emailAddress, String office, double salary,  String date\_hired, String title) {  super(name, address, phoneNumber, emailAddress, office, salary, date\_hired);  this.title = title;  }  @Override  public String toString() {  return super.toString() + String.format("Title: %s\n", getTitle());  }  }  TestDriverQ05  package karepe\_exam2.question5;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class TestDriverQ05 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Person person = new Person("Wilmes", "Quail Forest Drive 123G NC-28412", "657-876-9087","wilmes56@nwmissori.edu");  System.out.println(person.toString());  Student student = new Student("Nithya Karepe", "1115N College Drive Maryville Mo-64468", "132-456-6767", "s540109@nwmissouri.edu", 4.0);  System.out.println(student.toString());  Employee employee = new Employee("CsDpt-1109", 8790.00, "11/01/2012", person);  System.out.println(employee.toString());  Faculty faculty = new Faculty("Wilmes", "Quail Forest Drive 123G NC-28412", "657-876-9087","wilmes56@nwmissori.edu", "CsDpt-1109", 8790.00, "11/01/2012", "TR- 1:00pm-3:00pm", 2);  System.out.println(faculty.toString());  Staff staff = new Staff("Joann", "7350 McArdle rd Apt153 MO-64468", "637-076-9000", "joann@nwmissori.edu","Admin-1009", 8790.00, "11/01/2012", "Clerk");  System.out.println(staff.toString());  }  Output : |

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.

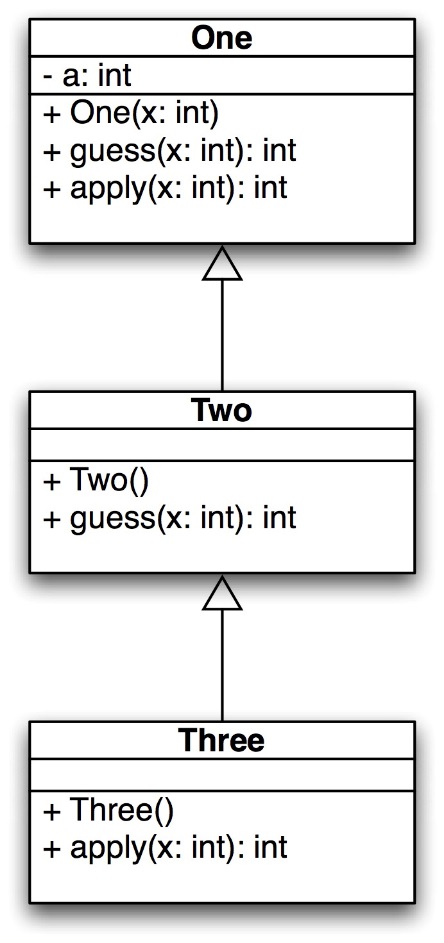
|  |
| --- |
| **GeometricObject Class**  package karepe\_exam2.question6;  /\*\*  \*  \* @author Nithya Karepe  \*/  public abstract class GeometricObject {  public String color;  public boolean filled = false;  /\*\*  \* @return the color  \*/  public String getColor() {  return color;  }  /\*\*  \* @param color the color to set  \*/  public void setColor(String color) {  this.color = color;  }  /\*\*  \* @return the filled  \*/  public boolean isFilled() {  return filled;  }  /\*\*  \* @param filled the filled to set  \*/  public void setFilled(boolean filled) {  this.filled = filled;  }  /\*\*  \*  \* @return  \*/  public abstract double getArea();  /\*\*  \*  \* @return  \*/  public abstract double getPerimeter();  /\*\*  \* @param color  \* @param filled  \*/  public GeometricObject(String color, boolean filled) {  this.color = color;  this.filled = filled;  }  }  **Triangle**  package karepe\_exam2.question6;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Triangle extends GeometricObject {  public double side1;  public double side2;  public double side3;  /\*\*  \* @return the side1  \*/  public double getSide1() {  return side1;  }  /\*\*  \* @param side1 the side1 to set  \*/  public void setSide1(double side1) {  this.side1 = side1;  }  /\*\*  \* @return the side2  \*/  public double getSide2() {  return side2;  }  /\*\*  \* @param side2 the side2 to set  \*/  public void setSide2(double side2) {  this.side2 = side2;  }  /\*\*  \* @return the side3  \*/  public double getSide3() {  return side3;  }  /\*\*  \* @param side3 the side3 to set  \*/  public void setSide3(double side3) {  this.side3 = side3;  }  /\*\*  \* @param color  \* @param filled  \* @param side1  \* @param side2  \* @param side3  \*/  public Triangle(String color, boolean filled, double side1, double side2, double side3) {  super(color, filled);  this.side1 = side1;  this.side2 = side2;  this.side3 = side3;  }  /\*\*  \*  \* @return  \*/  @Override  public double getArea() {  double s = getPerimeter() / 2;// semi perimeter  return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3)); // using heron's formula  }  /\*\*  \*  \* @return  \*/  @Override  public double getPerimeter() {  return getSide1() + getSide2() + getSide3();  }  @Override  public String toString() {  return "Triangle: " + "\n Side1 = " + side1 + "\n Side2 = " + side2 + "\n Side3 = "  + side3 + "\n Color = " + color + "\n Filled = " + filled + "\n Area = "  + getArea() + "\n Perimeter = " + getPerimeter();  }  }  **GeometricObjectDriver**  package karepe\_exam2.question6;  import java.util.Scanner;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class GeometricObjectDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  System.out.println("Enter the Values here: ");  System.out.print("Side1 : ");  Scanner scan = new Scanner(System.in);  double side1 = scan.nextDouble();  System.out.print("Side2 : ");  double side2 = scan.nextDouble();  System.out.print("Side3 : ");  double side3 = scan.nextDouble();  System.out.print("Color : ");  String color = scan.next();  System.out.print("Filled : ");  boolean filled = scan.nextBoolean();  scan.close();  GeometricObject triangle = new Triangle(color, filled, side1, side2, side3);  System.out.println(triangle.toString());  }  }  Output |

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

|  |
| --- |
| An enum is special class where it contains a group of**constants. It is a** Java type used to define collections of constants. An enum can contain constants, methods etc.  **Enum Example**  **public enum GradCourses {**  **Applied\_Computer\_Science,**  **Information\_Systems;**  **}**  We have a variable **GradCourses** of **enum type**, which is a collection of four constants **Applied\_Computer\_Science, Information\_Systems;**  **Executable Code with Explanation**  **enum CountryCode** :  package karepe\_exam2.question7;  /\*\*  \*  \* @author Nithya Karepe  \*/  public enum CountryCode {  US("United States"), UK("United Kingdom"), IN("India"), AU("Australia"), CA("Canada"), MX("Mexico"),  HK("Hong Kong");  private String value;  /\*\*  \* @param value  \*/  private CountryCode(String value) {  this.value = value;  }  /\*\*  \* @return the value  \*/  public String getValue() {  return value;  }  }  **enum PhoneCode**  package karepe\_exam2.question7;  /\*\*  \*  \* @author Nithya Karepe  \*/  public enum PhoneCode {  United\_States("+1"), United\_Kingdom("+44"), India("+91"), Australia("+61"), CA("+1"), MX("+52"),  HK("+852");  public String code;  private PhoneCode(String code) {  this.code = code;  }  public String getCode() {  return code;  }    }  **Driver Class**  package karepe\_exam2.question7;  /\*\*  \*  \* @author Nithya Karepe  \*/  public class Driver {  public static void main(String[] args) {  // Printing Country codes and corresponding values  CountryCode[] codes = CountryCode.values();  PhoneCode[] phonecodes = PhoneCode.values();  for (CountryCode code : codes) {  System.out.println("Country code : " + code.name() + " Country Name: " + code.getValue());  }  System.out.printf("I am Nithya Karepe. Currently living in %s. My home country is %s\n",  CountryCode.US.getValue(), CountryCode.IN.getValue());  for (PhoneCode phoneCode : phonecodes) {  System.out.println("Phone : " + phoneCode.name() + " " + phoneCode.getCode());  }  }  }  Output     1. In the below example we have consider enums classes and a driver class. 2. The enum classes are CountryCode class and PhoneCode class. 3. The driver class is named as Driver 4. The enum constants are generally represented in uppercase. 5. Collection of constants in CountryCode Class - US("United States"), UK("United Kingdom"),IN("India"), AU("Australia"), CA("Canada"), MX("Mexico"),   HK("Hong Kong")  Collection of constants in PhoneCode Class - United\_States("+1"), United\_Kingdom("+44"), India("+91"), Australia("+61"), CA("+1"), MX("+52"),HK("+852")   1. enum constructor type must be package-private or private access.We can’t invoke an enum constructor by ourselves. 2. These are used to get values from the enum classes, we have used enchanced for loop which helps us to print the values of each constant of enums.   CountryCode[] codes = CountryCode.values();  PhoneCode[] phonecodes = PhoneCode.values();  for (CountryCode code : codes)  for (PhoneCode phoneCode : phonecodes) |

1. (10-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 |

|  |
| --- |
| Output:  Two guess 4  One guess 4  25  Three apply 12  -10  One guess 5  4  One guess 6  One guess 9  8  Explanation:  The code here follows the concepts of Inheritance, Polymorphism.  Inheritance happens in the above code at  Class Three extends Two  Class Two extends One  Polymorphism happens in the above code when methods from class One are overridden in class Two and Two methods are overridden in class Three  One hippo = new Three();  The above code is an example of Polymorphic substitution. When hippo object is created using constructor of Three class since Three extends Two and Three doesn’t have its own constructor, constructor of Two is called.  Two extends One and constructor of Two calls super(11) which assigns the value of 11 to private variable a from class One.  System.out.println(hippo.guess(4));  When hippo.guess(4) is called, since class Three doesn’t have its own guess(int) method, it executes the method from Two class(this is Late Binding Polymorphism) which prints the below  Two guess 4  After printing it calls super.guess(4) which prints the below  One guess 4  And returns a+4 (a value is 11 from above) so returns 15  So finally the guess(4) returns 15+10 =25 and the below is printed  25  System.out.println(hippo.apply(12));  When hippo.apply(12) is called, since class Three has its own apply(int) method, it prints the below  Three apply 12  After printing it returns -10 which prints the below  -10  One lion = new One(-1);  When lion object is created using constructor of One class it assigns the value of -1 to private variable a from class One.  System.out.println(lion.guess(5));  When lion.guess(5) is called it prints below  One guess 5  After printing it returns a+5(here a value is -1) which prints the below  4  System.out.println(lion.apply(6));  When lion.apply(6) is called it prints below  One guess 6  After printing it returns guess(x+3)(here x value is 6) which prints the below  One guess 9  After printing it return a+x(here a =-1, x=9) which prints the below  8 |

GitHub : <https://github.com/KarepeN/fall2020Exam2>