**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**

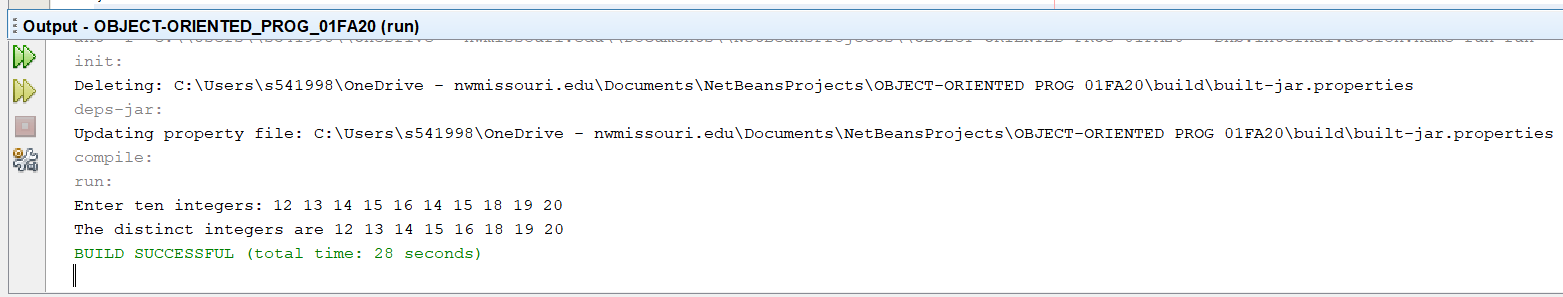
1. (5-Points) (1D-Array -) Write a method that removes the duplicate elements from an array list of integers using the following header:

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 |

|  |
| --- |
| /\*  \* 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;  import java.util.ArrayList;  import java.util.Scanner;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class tryToRemoveDuplicate {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ArrayList<Integer> manojArray = new ArrayList<>();  Scanner scan = new Scanner(System.in);  System.out.print("Enter ten integers: ");  for (int i = 0; i < 10; i++) {  manojArray.add(scan.nextInt());  }  removeDuplicate(manojArray);  }  public static void removeDuplicate(ArrayList<Integer> list) {  ArrayList<Integer> resultArray = new ArrayList<>();  for (Integer number : list) {  if (!(resultArray.contains(number))) {  resultArray.add(number);  }  }  System.out.print("The distinct integers are ");  for (Integer distinct : resultArray) {  System.out.print(distinct + " ");  }  System.out.println();  }  } |



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:

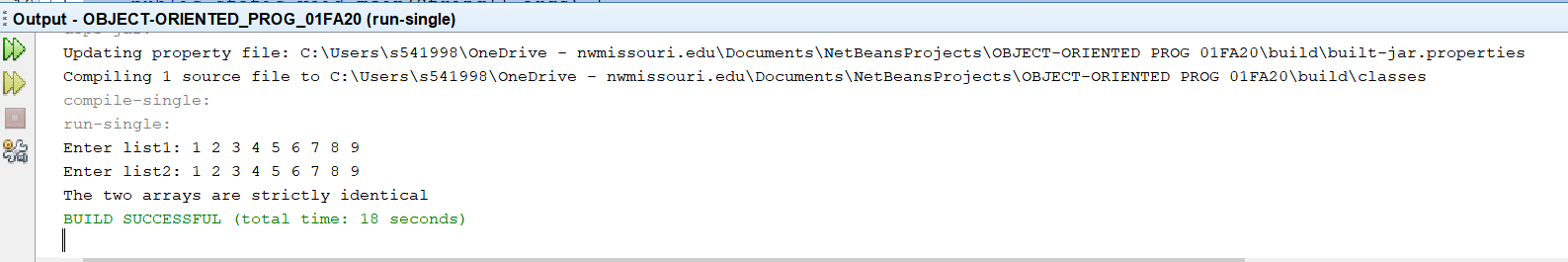
public static boolean equals(int[][] m1, int[][] m2)

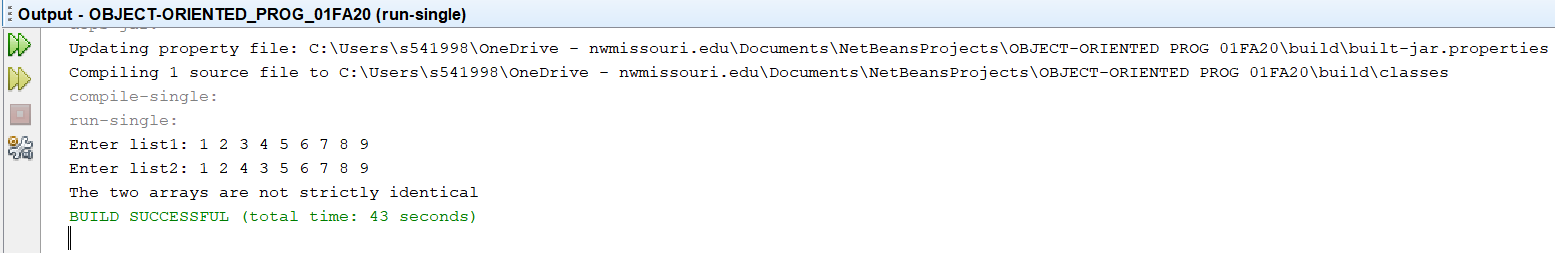
Write a test program that prompts the user to enter two 3 \* 3 arrays of integers and displays whether the two are strictly identical. Provide screenshot of executable code with input and output. Here are the sample runs.

|  |
| --- |
| 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 |

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class identiacalOrNotIdentical {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  // TODO code application logic here  int[][] m1 = new int[3][3];  int[][] m2 = new int[3][3];  System.out.print("Enter list1: ");  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 3; j++) {  m1[i][j] = scan.nextInt();  }  }  System.out.print("Enter list2: ");  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 3; j++) {  m2[i][j] = scan.nextInt();  }  }  boolean equalsOrNot = equals(m1, m2);  if (equalsOrNot) {  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) {  for (int i = 0; i < m1.length; i++) {  for (int j = 0; j < m2.length; j++) {  if (!(m1[i][j] == m2[i][j])) {  return false;  }  }  }  }  return true;  }  } |





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.java**

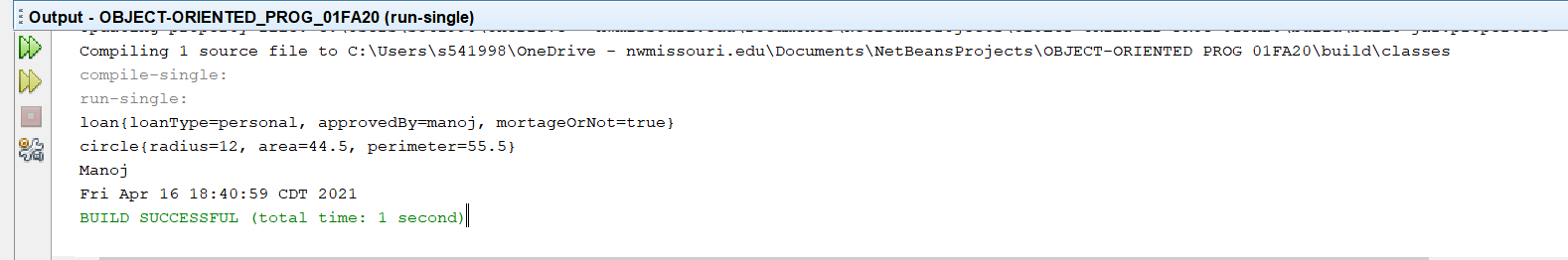
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question3;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class circle {  private int radius;  private double area;  private double perimeter;  public circle(int radius, double area, double perimeter) {  this.radius = radius;  this.area = area;  this.perimeter = perimeter;  }  @Override  public String toString() {  return "circle{" + "radius=" + radius + ", area=" + area + ", perimeter=" + perimeter + '}';  }  } |

**loan.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 question3;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class loan {  private String loanType;  private String approvedBy;  private boolean mortgageOrNot;  public loan(String loanType, String approvedBy, boolean mortageOrNot) {  this.loanType = loanType;  this.approvedBy = approvedBy;  this.mortgageOrNot = mortageOrNot;  }  @Override  public String toString() {  return "loan{" + "loanType=" + loanType + ", approvedBy=" + approvedBy + ", mortageOrNot=" + mortgageOrNot + '}';  }  } |

**Drivercl.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 question3;  import java.util.ArrayList;  import java.util.Date;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class drivercl {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ArrayList<Object> objects = new ArrayList<>();  loan loanObject = new loan("personal", "manoj", true);  circle circleObject = new circle(12, 44.5, 55.5);  String stringObj = "Manoj";  Date dateObj = new Date();  objects.add(loanObject);  objects.add(circleObject);  objects.add(stringObj);  objects.add(dateObj);  for (Object LCSD : objects) {  System.out.println(LCSD);  }  }  } |

****

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

**Inheritance:**

Inheritance is defined a class to derive the properties and characteristics from other classes where we can extract features from the base class and implement it into other derived classes.

**Polymorphism:**

It is the ability of an object to take many forms where polymorphism uses these methods to perform different tasks. polymorphism in Java occurs when there are one or more classes or objects related to each other by inheritance.

**Late Binding polymorphism:**

This generally occurs after the polymorphic substitution. Once an object of instance parent which holds the child, instance calls any method. Then it is defined as late binding polymorphism.

Examples :

Here we created parent class called bikes which has subclasses like bullet and yamaharx100.We inherit features from the parent class and override few methods. This is example of inheritance.

In the driver class while creating objects we hold the child object in the parent object and we call the methods using that object which is Late binding polymorphism and which checks whether the calling class has that method or not if there is no method then the JVM with determine from which class the method has to be called. This because of inheritance. We are invoking the toString method of parent using super in the child classes.

**Bikes.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;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class bikes {  private String category;  private double price;  public bikes(String category, double price) {  this.category = category;  this.price = price;  }  @Override  public String toString() {  return "bikes{" + "category=" + category + ", price=" + price + '}';  }  } |

**Bullet.java**

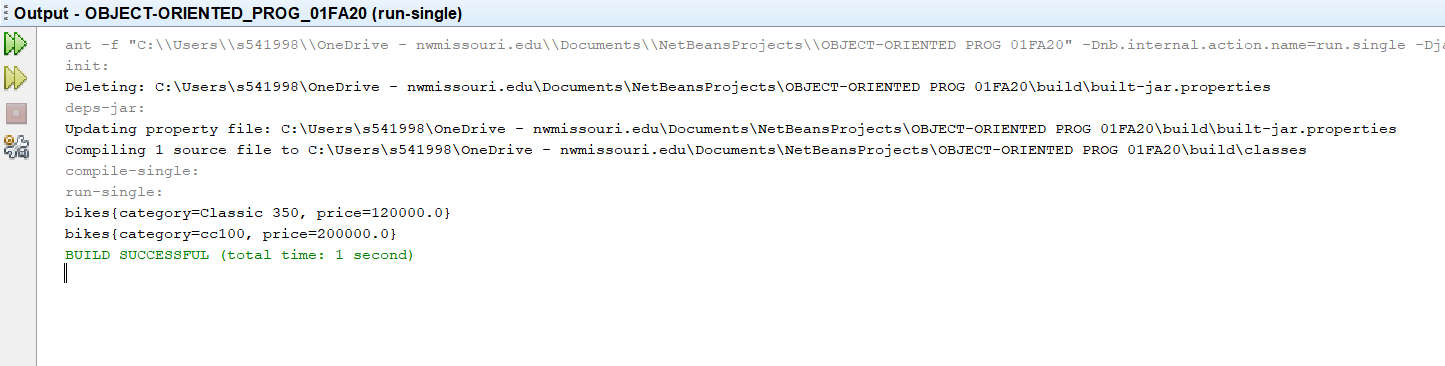
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question4;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class bullet extends bikes {  private String model;  public bullet(String model, String category, double price) {  super(category, price);  this.model = model;  }  @Override  public String toString() {  return "bullet{" + "model=" + model + '}';  }  } |

**Yamaharx100.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;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class yamaharx100 {  private String cc;  public yamaharx100(String cc) {  this.cc = cc;  }  @Override  public String toString() {  return "yamaharx100{" + "cc=" + cc + '}';  }  } |

**Bikedriver.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;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class bikesDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  bikes bullet = new bikes("Classic 350", 120000);  bikes yamaharx100 = new bikes("cc100", 200000);  System.out.println(bullet);  System.out.println(yamaharx100);  }  } |

****

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.

**Person.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 question5;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  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 from " + this.getClass().getName() + " class is " + name;  }  } |

**Student.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 question5;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class Student extends Person {  public static final String CLASS\_STATUS = "Graduate";  private final double grade;  public Student(double grade, String name, String address,  long phoneNumber, String emailAddress) {  super(name, address, phoneNumber, emailAddress);  this.grade = grade;  }  } |

**Employee.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 question5;  import java.util.Date;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  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();  }  } |

**Staff.java**

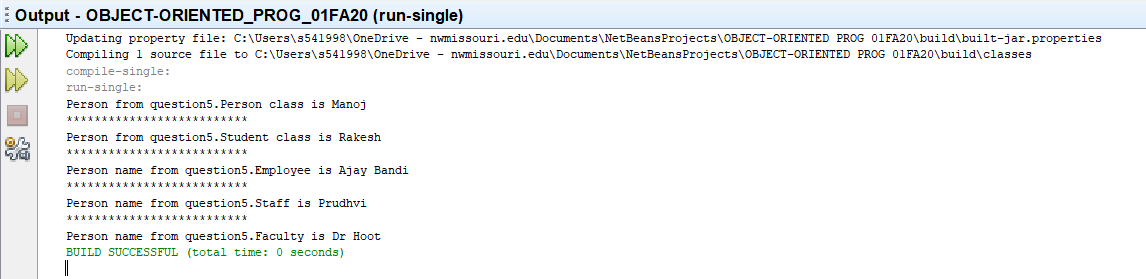
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question5;  import java.util.Date;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class 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();  }  } |

**Faculty.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 question5;  import java.util.Date;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  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();  }  } |

**driverperson.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 question5;  import java.util.Date;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class driverperson {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Date D = new Date();  Person person1 = new Person("Manoj", "Maryville",  224528386, "man@gmail.com");  Student student1 = new Student(9.4, "Rakesh", "Maryville",  999999999, "racks@gmail.com");  Employee employee1 = new Employee("Northwest", 20000, D, "Ajay Bandi",  "Maryville", 111111, "ab@gmail.com");  Staff staff1 = new Staff("Assistant", "Assisatance", 4000, D,  "Prudhvi", "MAryville", 191919191, "r@gmail.com");  Faculty faculty1 = new Faculty(12, 3, "Colden Hall", 100000, D,  "Dr Hoot", "Northwest", 979797979, "bb@gmail.com");  System.out.println(person1);  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(student1);  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(employee1);  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(staff1);  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(faculty1);  }  } |



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**

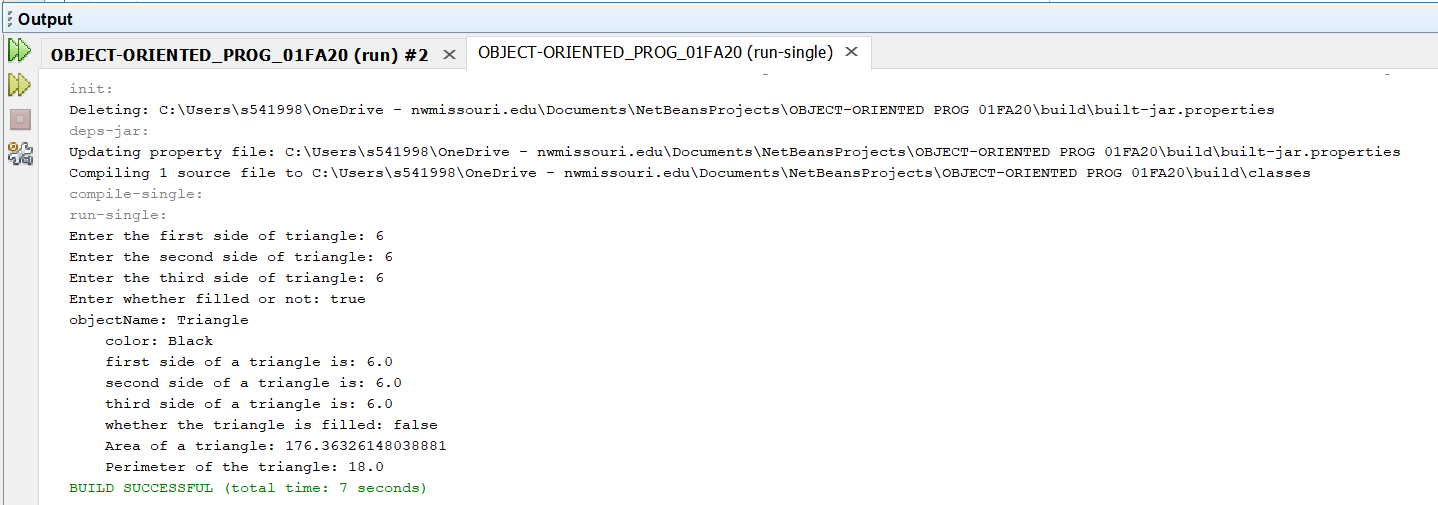
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question6;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public abstract class GeometricObject {  private final String objectName;  private final String color;  public GeometricObject(String objectName, String color) {  this.objectName = objectName;  this.color = color;  }  public String getObjectName() {  return objectName;  }  public String getColor() {  return color;  }  public abstract boolean isFilled();  public abstract double areaOfTriangle();  public abstract double perimeterOfTriangle();  @Override  public String toString() {  return "objectName: " + objectName + "\n color: " + color;  }  } |

**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 Manoj Nuvvala  \*/  public class Triangle extends GeometricObject {  private final double s1;  private final double s2;  private final double s3;  private String filling;  public Triangle(double s1, double s2, double s3, String filling, String objectName, String color) {  super(objectName, color);  this.s1 = s1;  this.s2 = s2;  this.s3 = s3;  this.filling = filling;  }  public double getTr1() {  return s1;  }  public double getTr2() {  return s2;  }  public double getTr3() {  return s3;  }  public String getFilling() {  return filling;  }  @Override  public boolean isFilled() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  if (filling.equals("filled")) {  return true;  } else {  return false;  }  }  @Override  public double perimeterOfTriangle() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  double perimeter = s1 + s2 + s3;  return perimeter;  }  @Override  public double areaOfTriangle() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  double x = (perimeterOfTriangle() \* ((perimeterOfTriangle() - s1) \* (perimeterOfTriangle() - s2) \* (perimeterOfTriangle() - s3)));  double area = Math.sqrt(x);  return area;  }  @Override  public String toString() {  return super.toString() + "\n first side of a triangle is: " + s1 + "\n second side of a triangle is: " + s2  + "\n third side of a triangle is: " + s3  + "\n whether the triangle is filled: " + isFilled()  + "\n Area of a triangle: " + areaOfTriangle()  + "\n Perimeter of the triangle: " + perimeterOfTriangle();  }  } |

**triangledriver.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 Manoj Nuvvala  \*/  public class triangledriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner nacs = new Scanner(System.in);  System.out.print("Enter the first side of triangle: ");  double s1 = nacs.nextDouble();  System.out.print("Enter the second side of triangle: ");  double s2 = nacs.nextDouble();  System.out.print("Enter the third side of triangle: ");  double s3 = nacs.nextDouble();  System.out.print("Enter whether filled or not: ");  String filling = nacs.next();  GeometricObject geoOb = new Triangle(s1, s2, s3, filling, "Triangle", "Black");  System.out.println(geoOb.toString());  }  } |



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

An Enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

Example: compass directions :-NORTH, SOUTH, EAST, and WEST

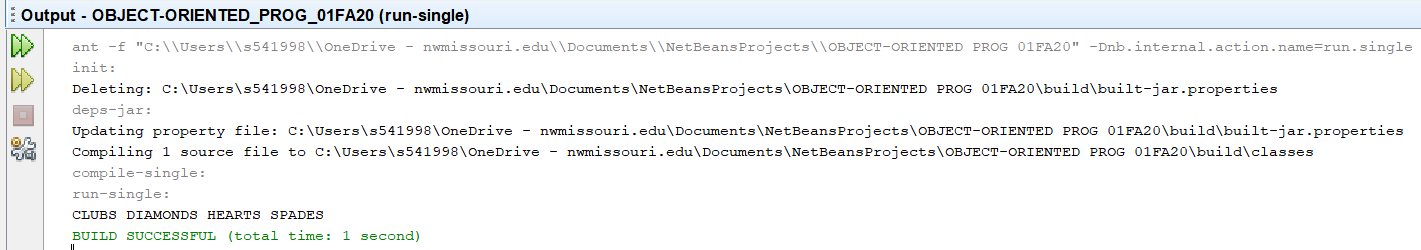
Here we took cards of specific constants in Enum class and calling them in the driver class

**Cards.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 question7;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public enum cards {  HEARTS, CLUBS, DIAMONDS, SPADES  } |

**cardsDriver.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 question7;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class cardsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println(cards.CLUBS + " " + cards.DIAMONDS + " " + cards.HEARTS + " " + cards.SPADES);  }  } |



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

Abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class) and contains all methods in the class which can be inherited in other classes.

Example :

**AbstractExampleCourse.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 question8;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public abstract class AbstractExampleCourse {  private String courseName;  private int courseCode;  public AbstractExampleCourse(String courseName, int courseCode) {  this.courseName = courseName;  this.courseCode = courseCode;  }  public String getCourseName() {  return courseName;  }  public int getCourseCode() {  return courseCode;  }  public void setCourseName(String courseName) {  this.courseName = courseName;  }  public void setCourseCode(int courseCode) {  this.courseCode = courseCode;  }  @Override  public String toString() {  return "abstractexample{" + "courseName=" + courseName + ", courseCode=" + courseCode + '}';  }  } |

**Python.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 question8;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class Python extends AbstractExampleCourse {  private int courseCredits;  private String toolUsed;  public Python(int courseCredits, String toolUsed, String courseName, int courseCode) {  super(courseName, courseCode);  this.courseCredits = courseCredits;  this.toolUsed = toolUsed;  }  public int getCourseCredits() {  return courseCredits;  }  public void setCourseCredits(int courseCredits) {  this.courseCredits = courseCredits;  }  public String getToolUsed() {  return toolUsed;  }  public void setToolUsed(String toolUsed) {  this.toolUsed = toolUsed;  }  public double courseCost() {  double cost = 0.0;  cost = courseCredits \* 120;  return cost;  }  @Override  public String toString() {  return super.toString() + "\nDevops{" + "courseCredits=" + courseCredits + ", toolUsed=" + toolUsed + ", course fee=" + courseCost() + '}';  }  } |

**pythonDriver.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 question8;  import java.util.Scanner;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class pythonDriver {  /\*\*  \* @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 the Course Name : ");  String name = scan.next();  System.out.println("Enter the Course code : ");  int code = scan.nextInt();  System.out.println("Enter the course Credits : ");  int credits = scan.nextInt();  System.out.println("Enter course professor Name : ");  String professor = scan.next();  System.out.println("Enter the tool used : ");  String tool = scan.next();  Python pythonobj = new Python(credits, professor, name, code);  System.out.println(pythonobj.toString());  }  } |



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

An **Interface in Java** programming is defined as an abstract type used to specify the behavior of a class. A Java interface contains static constants and abstract methods. A class can implement multiple interfaces.

Example:

**birds.java**

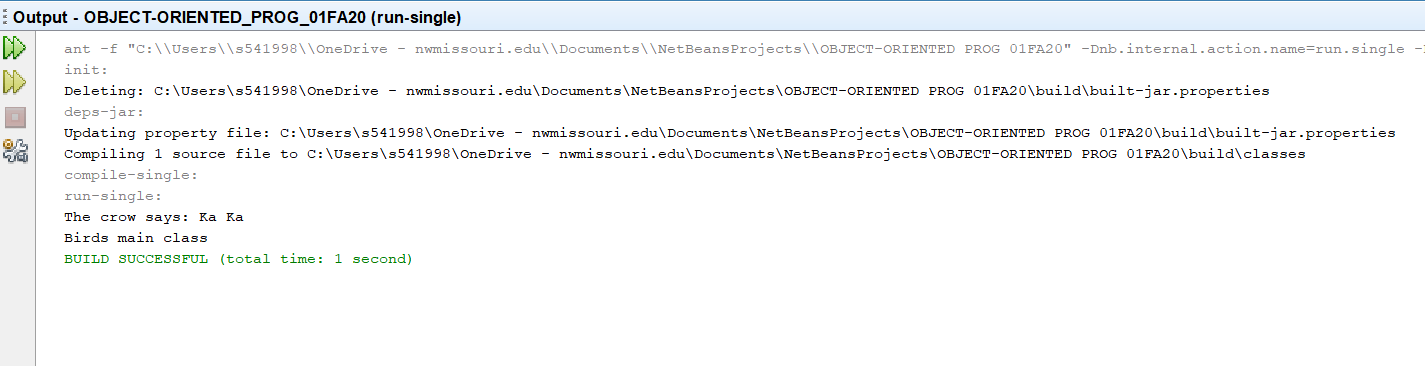
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question9;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  abstract class birds {  // Abstract method (does not have a body)  public abstract void birdsSound();  // Regular method  public void original() {  System.out.println("Birds main class");  }  } |

**crows.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 Manoj Nuvvala  \*/  public class crow extends birds {  public void birdsSound() {  System.out.println("The crow says: Ka Ka");  }  } |

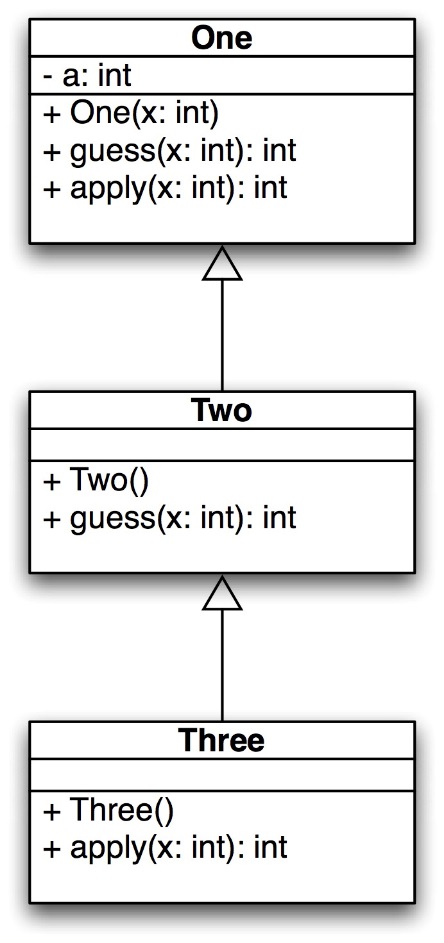
**Birdsdriver.java**

|  |
| --- |
| /\*  \* 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 Manoj Nuvvala  \*/  public class birdsdriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  crow mycrow = new crow(); // Create a Pig object  mycrow.birdsSound();  mycrow.original();  }  } |



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 |

**One.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 question10;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  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);  }  } |

**Two.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 question10;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  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;  }  } |

**Three.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 question10;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class three extends two {  public int apply(int x) {  System.out.println("Three apply " + x);  return -10;  }  } |

**Driver.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 question10;  /\*\*  \*  \* @author Manoj Nuvvala  \*/  public class driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  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));  }  } |



In the driver class the statements

One hippo=new Three(); is called Polymorphic substitution since the parent object holding the instance of child object i.e., Three subclass.

Whenever a method is called using the hippo object the methods inside the subclass Three are called. This is called Late binding Polymorphism.

Here hippo.guess(4) is called in the print statement. Now the JVM checks whether there is a method named guess inside Three class since the hippo is created for Three subclass. We don’t see any guess method inside the Three class so it figures out and calls method form its super class that is Two (This is because Inheritance).

Yes, Two has guess method but the definition guess in Two calls guess from its super which is One again.

Here the constructor is also invoked by setting the value of a to 11 from the Two class. Now it returns the a+x where a is 11 and x=4 which is added to 10 in the return statement of guess method in Two class and returns 25.

The output for the first statement is :

Two guess 4

One guess 4

25

Similarly, the apply method in the statement hippo.apply(12) checks in the Three class which is present and it is returning the value -10.

The output will be :

Three apply 12

-10

The next statement is creating an instance(lion) for One class with parameter -1. When methods are called over the lion object the methods inside the One class are invoked.

Since the One class has guess and apply methods in it. These methods are invoked.

The output is straight with a=-1 and x=5 which returns 4

Similarly, the One class has apply method which is again calling guess with x added to 3 and returns

9 since a=-1 and x=9.