**Final Exam 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 . 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.

**Final Exam OBJECT-ORIENTED PROG 01FA20 150 pts**

1. (20-Points) Define the terms abstract classes and interfaces. What are the similarities and differences between abstract classes and interfaces? Why interfaces are preferred over abstract classes? Explain and demonstrate with examples.

Answer

**Abstract Classes:** A class that is declared using keyword **Abstract** is known as

Abstract class.

**Interfaces:** It is a completely "abstract class" that is used to group related methods with empty bodies (that is used to specify a behavior that classes must implement)

**Differences:**

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| It can have abstract and non-abstract methods. | It can have only abstract method |
| It does not support multiple inheritance | I supports multiple inheritance |
| It can have final ,non final static ,non static  variables | It can have only static and final variables. |
| It can provide the implementation of interface. | It can't provide the implementation of abstract class. |
| The abstract keyword is used to declare abstract class. | The interface keyword is used to declare an interface. |
| An abstract class can be extended using the keyword "extends". | An interface can be implemented using keyword "implements". |
| A Java abstract class can have class members like private, protected, etc. | Members of a Java interface are public by default |
| Example:  public abstract class drawable{  public abstract void draw();  } | Example:  public interface Drawable{  void draw();  } |

**Similarities of interfaces and abstract class:**

Interfaces can not be instantiated. Same way, you can not instantiate abstract class.

That means you can not create an object of interface or abstract class.

**Reason why interfaces are preferred over abstract classes**

An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it. And whereas a class can extend only one abstract class, it can take advantage of multiple interfaces.

**Abstract class Example**

**Explanation-**

Bank is super class and axis-midwest are sub classes.

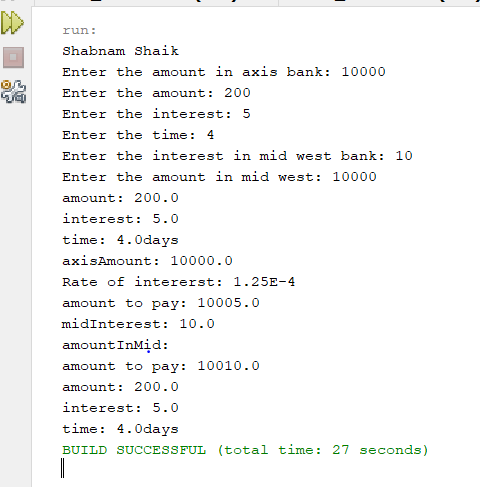
I calculated the rate of interest in bank and amount to pay is declared for abstract classes.

Amount to pay is calculated in both sub classes separately rate of interest is overrided in both classes.

When tostring method is executed, the amount to pay and rate of interest is calculated for both sub classes.

|  |
| --- |
| /\*  \* 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 question08\_Abstract1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public abstract class Bank {  private final double amount;  private final double interest;  private final double time;  public Bank(double amount, double interest, double time) {  this.amount = amount;  this.interest = interest;  this.time = time;  }  public double getAmount() {  return amount;  }  public double getInterest() {  return interest;  }  public double getTime() {  return time;  }  public double rateOfInterest() {  return interest / (amount \* time);  }  /\*\*  \*  \* @return  \*/  public abstract double amountToPay();  @Override  public String toString() {  return "amount: " + amount + "\ninterest: " + interest + "\ntime: " + time + "days";  }  }  /\*  \* 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 question08\_Abstract1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Axis extends Bank {  private final double axisAmount;  public Axis(double axisAmount, double amount, double interest, double time) {  super(amount, interest, time);  this.axisAmount = axisAmount;  }  public double getAxisAmount() {  return axisAmount;  }  /\*\*  \*  \* @return  \*/  @Override  public double rateOfInterest() {  return super.getInterest() / (axisAmount \* super.getTime());  }  /\*\*  \*  \* @return  \*/  @Override  public double amountToPay() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return axisAmount + super.getInterest();  }  @Override  public String toString() {  return super.toString() + "\naxisAmount: " + axisAmount  + "\nRate of intererst: " + rateOfInterest()  + "\namount to pay: " + amountToPay();  }  }  /\*  \* 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 question08\_Abstract1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class MidWest extends Bank {  private final double midInterest;  private final double amountInMid;  public MidWest(double midInterest, double amountInMid, double amount, double interest, double time) {  super(amount, interest, time);  this.midInterest = midInterest;  this.amountInMid = amountInMid;  }  public double getMidInterest() {  return midInterest;  }  public double getAmountInMid() {  return amountInMid;  }  /\*\*  \*  \* @return  \*/  @Override  public double rateOfInterest() {  return midInterest / (amountInMid \* super.getTime());  }  /\*\*  \*  \* @return  \*/  @Override  public double amountToPay() {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return amountInMid + midInterest;  }  @Override  public String toString() {  return "midInterest: " + midInterest + "\namountInMid: " + amountInMid  + "\n" + super.toString();  }  }  /\*  \* 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 question08\_Abstract1;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class BankDriver {  /\*\*  \* @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.print("Shabnam Shaik");  System.out.print("Enter the amount in axis bank: ");  double axisAmount = scan.nextDouble();  System.out.print("Enter the amount: ");  double amount = scan.nextDouble();  System.out.print("Enter the interest: ");  double interest = scan.nextDouble();  System.out.print("Enter the time: ");  double time = scan.nextDouble();  System.out.print("Enter the interest in mid west bank: ");  double midInterest = scan.nextDouble();  System.out.print("Enter the amount in mid west: ");  double amountInMid = scan.nextDouble();  Axis ax = new Axis(axisAmount, amount, interest, time);  MidWest mid = new MidWest(midInterest, amountInMid, amount, interest, time);  System.out.println(ax.toString());  System.out.println(mid.toString());  }  } |

**Output**



**Interface class Example**

**Explanation**

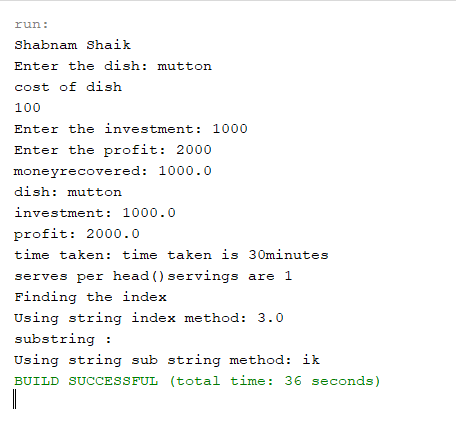
Biryani is class and other two interfaces are where I calculate the time taken and servings per head.

In servings I took the calculations for index and in timetaken I did for substring.

In biryani class, I did the main methods required for servings and time taken for the particular dish required.

|  |
| --- |
| /\*  \* 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 question09\_Intereface1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public interface Servings {  default double String(String sh) {  int str = sh.indexOf(“b”);  return str;  }  /\*\*  \*  \* @return  \*/  String servesperhead();  }  /\*  \* 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 question09\_Intereface1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public interface Time {  String timetaken();  default String myst(String name) {  String str = name.substring(3);  return str;  }  }  /\*  \* 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 question09\_Intereface1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Biryani implements Time, Servings {  private final String dish;  private final double investment;  private final double profit;  public Biryani(String dish, double investment, double profit) {  this.dish = dish;  this.investment = investment;  this.profit = profit;  }  public String getDish() {  return dish;  }  public double getInvestment() {  return investment;  }  public double getProfit() {  return profit;  }  public double moneyRecovered() {  return profit – investment;  }  @Override  public String timetaken() {  String str = “ “;  if (dish.equals(“mutton”)) {  str = “time taken is 30minutes”;  }  if (dish.equals(“chicken”)) {  str = “time taken is 20minutes”;  } else if (dish.equals(“panneer”)) {  str = “time taken is 10minutes”;  }  return str;  }  /\*\*  \*  \* @return  \*/  @Override  public String servesperhead() {  String str = “ “;  if (dish.equals(“mutton”)) {  str = (“servings are 1”);  }  if (dish.equals(“chicken”)) {  str = (“servings are 2”);  }  if (dish.equals(“panneer”)) {  str = (“servings are 3”);  }  return str;  }  /\*\*  \*  \* @param sh  \* @return  \*/  @Override  public double String(String sh) {  System.out.print(“Using string index method: “);  int str = (sh.indexOf(“b”));  return str;  }  /\*\*  \*  \* @param name  \* @return  \*/  @Override  public String myst(String name) {  System.out.print(“Using string sub string method: “);  String str = name.substring(3);  return str;  }  @Override  public String toString() {  return “dish: “ + dish + “\ninvestment: “ + investment + “\nprofit: “ + profit  + “\ntime taken: “ + timetaken() + “\nserves per head()” + servesperhead();  }  }  /\*  \* 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 question09\_Intereface1;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class BiryaniDriver {  /\*\*  \* @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.print(“Shabnam Shaik”);  System.out.print(“Enter the dish: “);  String dish = scan.next();  System.out.println(“cost of dish”);  double cost = scan.nextDouble();  System.out.print(“Enter the investment: “);  double investment = scan.nextDouble();  System.out.print(“Enter the profit: “);  double profit = scan.nextDouble();  Biryani b = new Biryani(dish, investment, profit);  System.out.println(“moneyrecovered: “ + b.moneyRecovered());  System.out.println(b.toString());  System.out.println(“Finding the index”);  System.out.println(b.String(“Shabnam”));  System.out.println(“substring : “);  System.out.println(b.myst(“shaik”));  }  } |

**Output**



1. (10-Points) Design an interface named Colorable with a void method named howToColor(). Every class of a colorable object must implement the Colorable interface. Design a class named Square that extends GeometricObject and implements Colorable Implement howToColor to display the message Color all four sides.

Draw a UML diagram that involves Colorable, Square, and GeometricObject. Write a test program that creates an array of five GeometricObjects. For each object in the array, display its area and invoke its howToColor method if it is colorable.

Answer

**Explanation**

In this GeometricObject class, taken a variable to find the length which is of type integer and declared variable is a. In square interface, extending to geometricobject implementing colorable.

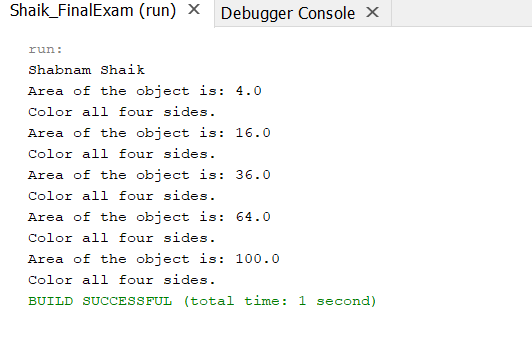
In square again, extracting the length integer and also declaring a new method for how to color.

In colorable interface implementing on method how to color.

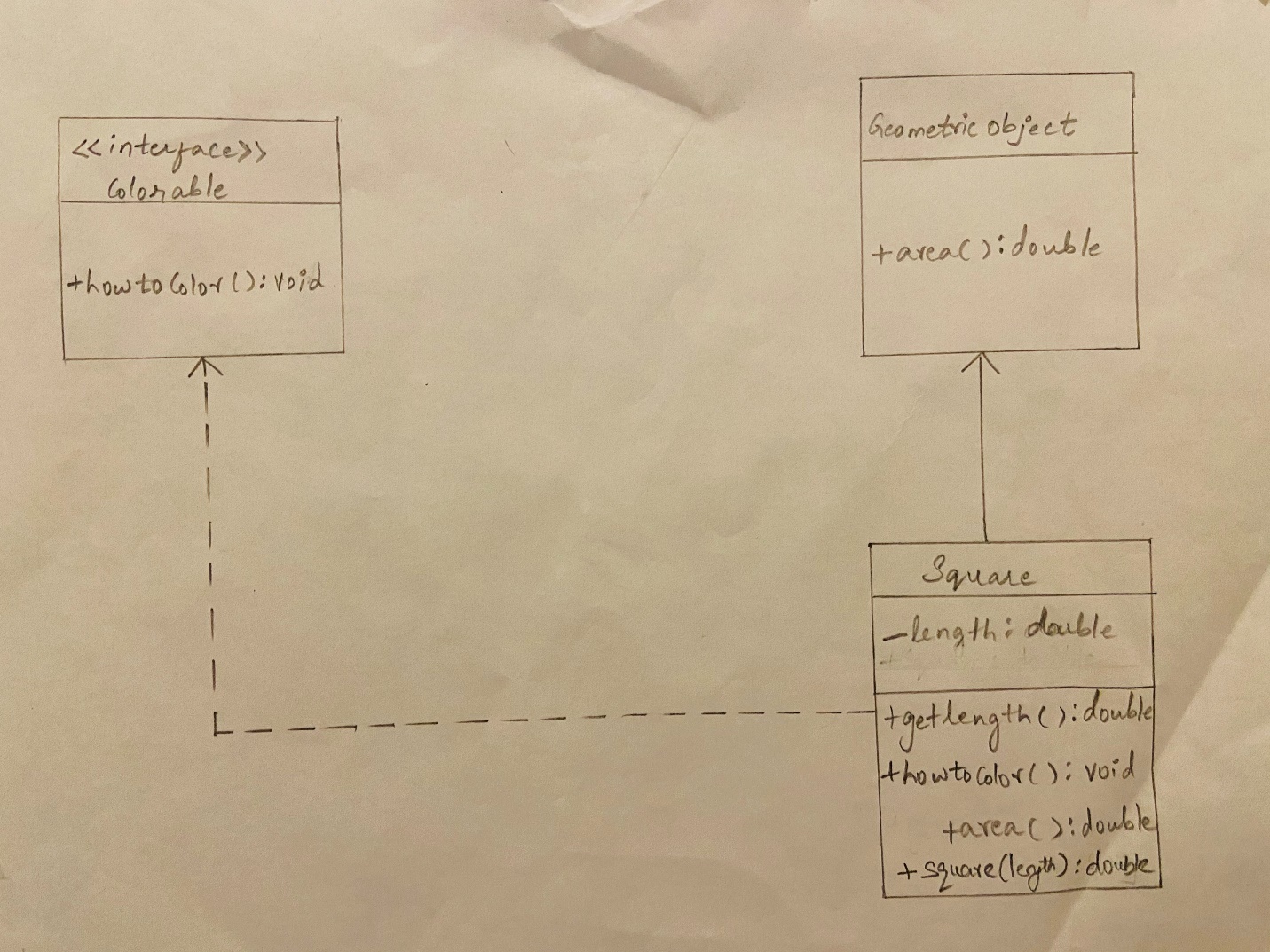
In driver class giving an array list for square and giving objects naming one,four,six,eight,ten and finding the calculated area along with checking if it is colorable or not.

|  |
| --- |
| /\*  \* 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 question02;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public interface Colorable {  void howToColor();  }  /\*  \* 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 question02;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public abstract class GeometricObject {  public abstract double area();  }  /\*  \* 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 question02;  import java.util.ArrayList;  import java.util.List;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class GeometricObjectDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  GeometricObject[] object = new Square[5];  GeometricObject one = new Square(2);  GeometricObject four = new Square(4);  GeometricObject six = new Square(6);  GeometricObject eight = new Square(8);  GeometricObject ten = new Square(10);  object[0] = one;  object[1] = four;  object[2] = six;  object[3] = eight;  object[4] = ten;  System.out.println("Shabnam Shaik");  for (GeometricObject g : object) {  System.out.println("Area of the object is: " + g.area());  if (g instanceof Colorable) {  Colorable co = (Colorable) g;  co.howToColor();  }  }  }  }  /\*  \* 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 question02;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Square extends GeometricObject implements Colorable {  private double length;  public Square(double length) {  this.length = length;  }  public double getLength() {  return length;  }  public void setLength(double length) {  this.length = length;  }  /\*\*  \*  \*/  @Override  public void howToColor() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  System.out.println("Color all four sides.");  }  @Override  public double area() {  return length \* length;  }  } |

**Output**



UML



1. (10-Points) What is casting? What are different types of casting? Explain and demonstrate with examples.

Answer

**Casting** is a method or process that converts a data type into another data type in both ways manually and automatically.

or

It is when you assign a value of one primitive data type to another type.

There are 2 types of casting

**Widening casting**: Widening, also known as upcasting, converting a smaller type to a larger type size. byte -> short -> char -> int -> long -> float -> double. Widening takes place when a smaller primitive type value is automatically accommodated in a larger/wider primitive data type. Widening also takes place when a reference variable of a subclass is automatically accommodated in a reference variable of its superclass.

**Narrowing Casting:** Narrowing, also known as down casting/casting, is a conversion that is explicitly performed in the following situations, converting a higher datatype to a lower datatype is known as narrowing. In this case the casting/conversion is not done automatically, you need to convert explicitly using the cast operator “()” explicitly. Therefore, it is known as explicit type casting. In this case both datatypes need not be compatible with each other. Narrowing a wider/bigger primitive type value to a smaller primitive type value. Narrowing a superclass reference to a subclass reference, during inheritance.

**Examples**

**Widening Casting**

Explanation:

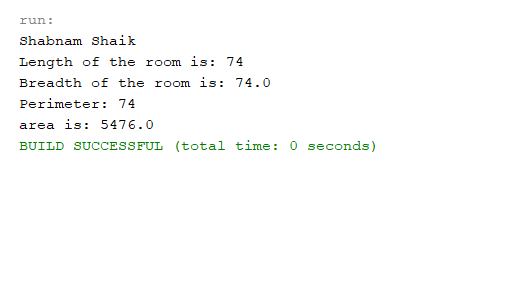
In room class given, room length-room breadth-perimeter in integer,double and long datatypes.

Finding out the area in double by calculating room length and room breadth.

In driver class, since widening casting gives smaller length to larger length, the output is given the same.

|  |
| --- |
| /\*  \* 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 question03Widening;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Room {  private int RoomLength;  private double RoomBreadth;  private long perimeter;  public Room(int RoomLength, double RoomBreadth, long perimeter) {  this.RoomLength = RoomLength;  this.RoomBreadth = RoomBreadth;  this.perimeter = perimeter;  }  public int getRoomLength() {  return RoomLength;  }  public void setRoomLength(int RoomLength) {  this.RoomLength = RoomLength;  }  public double getRoomBreadth() {  return RoomBreadth;  }  public void setRoomBreadth(double RoomBreadth) {  this.RoomBreadth = RoomBreadth;  }  public long getPerimeter() {  return perimeter;  }  public void setPerimeter(long perimeter) {  this.perimeter = perimeter;  }  public double getarea() {  double area = RoomLength \* RoomBreadth;  return area;  }  @Override  public String toString() {  return "Length of the room is: " + RoomLength + "\n" + "Breadth of the room is: " + RoomBreadth + "\nPerimeter: " + perimeter  + "\narea is: " + getarea();  }  }  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class RoomDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int RoomLength = 74;  long perimeter = RoomLength;  double RoomBreadth = perimeter;  Room r = new Room(RoomLength, RoomBreadth, perimeter);  System.out.println("Shabnam Shaik");  System.out.println(r.toString());  }  } |

Output



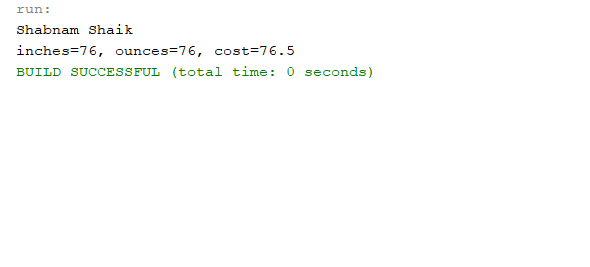
**Narrowing Casting**

Explanation

Given cost of cosmetics by finding it through inches ounces and cost with long, integer and double datatypes. Finding the cost by using method and then in driver class declaring so that the narrowing casting is done by printing from larger length to smaller length.

|  |
| --- |
| /\*  \* 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 question03Narrow;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Cosmetics {  private long inches;  private int ounces;  private double cost;  public Cosmetics(long inches, int ounces, double cost) {  this.inches = inches;  this.ounces = ounces;  this.cost = cost;  }  public long getInches() {  return inches;  }  public void setInches(long inches) {  this.inches = inches;  }  public int getOunces() {  return ounces;  }  public void setOunces(int ounces) {  this.ounces = ounces;  }  public double getCost() {  cost = ounces \* 24.3;  return cost;  }  @Override  public String toString() {  return "inches=" + inches + ", ounces=" + ounces + ", cost=" + cost;  }  }  /\*  \* 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 question03Narrow;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class CosmeticsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  double cost = 76.5;  long inches = (long) cost;  int ounces = (int) inches;  Cosmetics cos = new Cosmetics(inches, ounces, cost);  System.out.println("Shabnam Shaik");  System.out.println(cos.toString());  }  } |

Output



1. (15-Points) Suppose that Fruit, Apple, Orange, GoldenDelicious, and McIntosh are defined in the following inheritance hierarchy:

Fruit

Orange

Apple

GoldenDelicious

McIntosh

Assume that the following code is given:

Fruit fruit = new GoldenDelicious();

Orange orange = new Orange();

Answer the following questions and explain why these Statements are legal or illegal.

1. Is fruit instanceof Fruit?

**Answer:**

Yes,Instance of subclass is also an instance of super class.

1. Is fruit instanceof Orange?
2. **Answer:**

No, it super class instance can’t always be sub-class.

1. Is fruit instanceof Apple?
2. **Answer:**

No,We know that GoldenDelicious is the sub class of Apple.Instance of sub class is an instance of super class.

1. Is fruit instanceof GoldenDelicious?
2. **Answer:**

Yes,fruit is an instance of GoldenDelicious.

1. Is fruit instanceof McIntosh?

**Answer:**

No,as fruit is not an instance of McIntosh since it not a superclass of GoldenDelicious(Apple is superclass).fruit contains an instance if GoldenDelicious.

1. Is orange instanceof Orange?

**Answer:**

Yes,orange is instance of Orange.

1. Is orange instanceof Fruit?

**Answer:**

Yes,Orange object is an instance of Orange. Orange is a sub class of Fruit so orange instanceof Fruit.

1. Is orange instanceof Apple?

**Answer:**

No, here Apple is not an super class of Orange(object orange contains instance of class Orange).

1. Suppose the method makeAppleCider is defined in the Apple class. Can fruit invoke this method? Can orange invoke this method?

**Answer:**

Fruit can invoke method orange but orange cannot as orange is not an instance of Apple whereas fruit is an instance of Apple.

j. Suppose the method makeOrangeJuice is defined in the Orange class. Can orange invoke this method? Can fruit invoke this method?

**Answer:**

The object orange is an instance of the class Orange therefore, orange can invoke makeOrangeJuice method whereas fruit is not an instance of Orange so it cannot.

k.Is the statement Orange p = new Apple() legal?

**Answer:**

The object orange is an instance of the class Orange therefore, orange can invoke makeOrangeJuice method whereas fruit is not an instance of Orange so it cannot.

**Answer:**

l.Is the statement McIntosh p = new Apple() legal?

Illegal Statement, instance of Apple cannot be assigned to object of orange class because Apple is not a sub class of Orange.

1. Is the statement Apple p = new McIntosh() legal?
2. **Answer:**

Legal Statement, Apple is the super-class for class McIntosh and assignment of the instance of a sub-class to an object of the super class is possibl

1. (10-Points) Define a class named ComparableCircle that extends Circle and implements Comparable. Draw the UML diagram and implement the compareTo method to compare the circles on the basis of area. Write a test class to find the larger of two instances of ComparableCircle objects.

Answer

Explanation

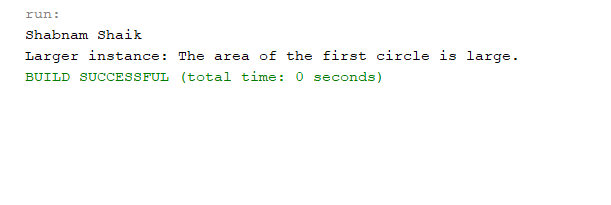
In circle class taken an variable named radius, given the constructor,getter and setter methods. Added a new method named getArea of type double, then written the toString method for radius.

In ComparableCircle it extends to Circle implements Comparable<ComparableCircle>. Then written the constructor for the radius where given the toString for superclass area and for radius of superclass.

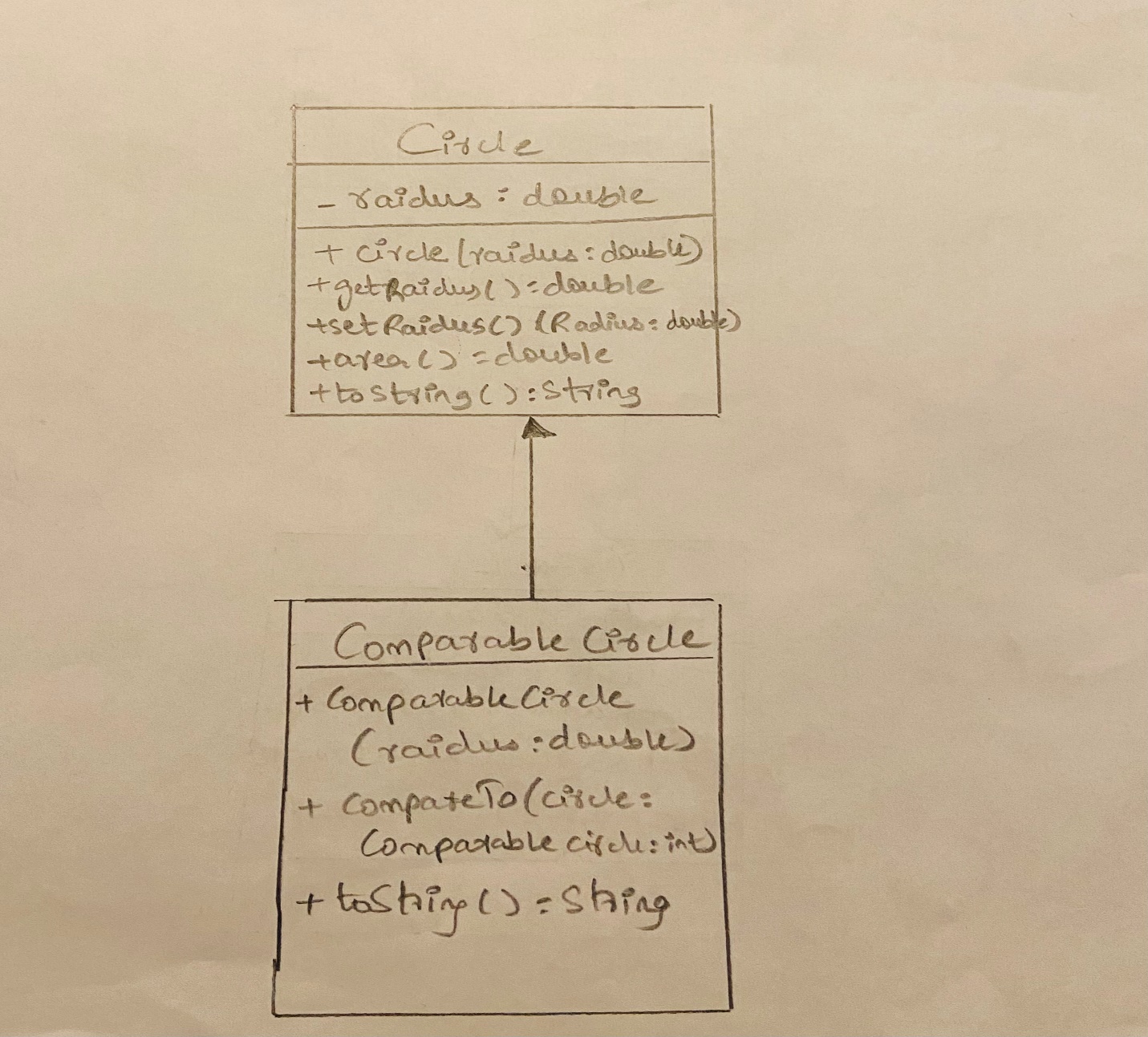
In driver class given two objects named circle1 and circle2 and given the print statement to print the largest instance of the circle. And used the switch case to compare the circle1 and circle2 instances to find the largest.

|  |
| --- |
| /\*  \* 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 question05;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Circle {  private double radius;  public Circle(double radius) {  this.radius = radius;  }  public double getRadius() {  return radius;  }  public void setRadius(double radius) {  this.radius = radius;  }  public double area() {  return Math.PI \* radius \* radius;  }  @Override  public String toString() {  return "radius: " + radius;  }  }  /\*  \* 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 question05;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class ComparableCircle extends Circle implements Comparable<ComparableCircle> {  public ComparableCircle(double radius) {  super(radius);  }  @Override  public String toString() {  return "radius: " + super.getRadius();  }  public int compareTo(ComparableCircle circle) {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return Double.compare(super.area(), circle.area());  }  }  /\*  \* 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 question05;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class ComparableCircleDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ComparableCircle circle1 = new ComparableCircle(26.5);  ComparableCircle circle2 = new ComparableCircle(13.4);  System.out.println("Larger instance: " + findLargest(circle1, circle2));  }  private static String findLargest(ComparableCircle circle1, ComparableCircle circle2) {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  String sha = " ";  if (circle1.compareTo(circle2) > 0) {  sha = "The area of ​​the first circle is large.";  } else if (circle1.compareTo(circle2) < 0) {  sha = "The area of ​​the second circle is large.";  } else {  sha = "The area of ​​the two circles is the same";  }  return sha;  }  } |

Output



UML:



1. (15-Points) What is an exception? What are checked and unchecked exceptions? Explain and demonstrate with examples.

Answer

Exception :

It is an event that interrupts the normal flow of the program execution.

Checked Exception:

They are checked at compile time.

It means if any program or method is showing checked exceptions then should be handled by try-catch block or it should be declared using throws keyword otherwise it shows compile time error.

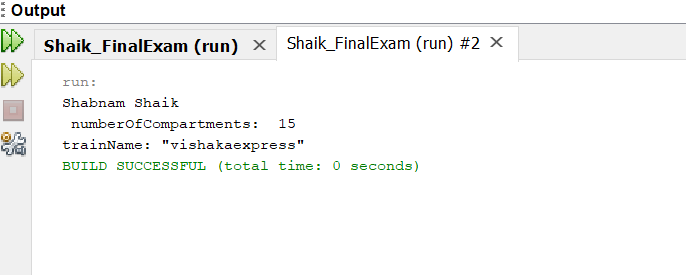
Example for checked exception

Explanation

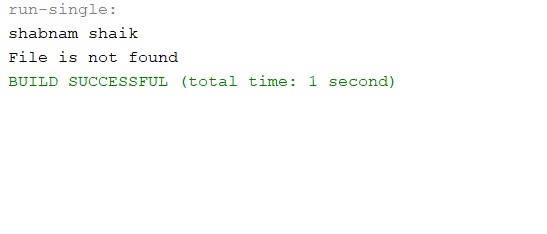
In class train, declared two variables number of comparments and train name.Declared constructor, setter and getter methods. Declaring two methods train name and number of comparments. In class driver, using scanner scan reading inputs for number of compartments and train name. If the input text file is given wrong, then it prints file not found.

|  |
| --- |
| /  \* 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\_checked;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Train {  private int numberOfCompartments;  private String trainName;  public Train(int numberOfCompartments, String trainName) {  this.numberOfCompartments = numberOfCompartments;  this.trainName = trainName;  }  public int getNumberOfCompartments() {  return numberOfCompartments;  }  public String getTrainName() {  return trainName;  }  public void setNumberOfCompartments(int numberOfCompartments) {  this.numberOfCompartments = numberOfCompartments;  }  public void setTrainName(String trainName) {  this.trainName = trainName;  }  @Override  public String toString() {  return " numberOfCompartments: " + numberOfCompartments + "\ntrainName: " + trainName;  }  }  /\*  \* 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\_checked;  import java.io.File;  import java.io.FileNotFoundException;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class TrainDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws FileNotFoundException {  // TODO code application logic here  Scanner scan = new Scanner(new File("input.txt"));  System.out.println("Shabnam Shaik");  int compartments = scan.nextInt();  String name = scan.next();  try {  Train train = new Train(compartments, name);  System.out.println(train.toString());  } catch (Exception e) {  System.out.println("file is not found");  }  }  } |

OUTPUT1:



Output 2:



**Unchecked Expection:**

Unchecked exceptions are not checked at compile time. It means if your program is throwing an unchecked exception and even if you didn’t handle/declare that exception, the program won’t give a compilation error.

This exception occurs due to the incorrect data provided by the user.

All Unchecked exceptions are direct subclasses of *RuntimeException* class.

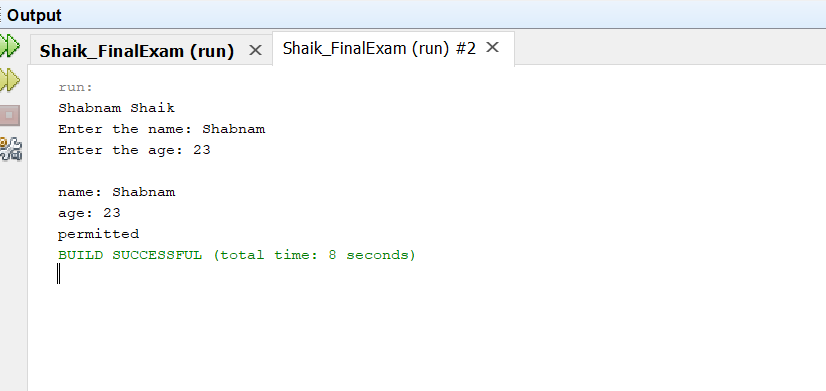
Example

Explanation:

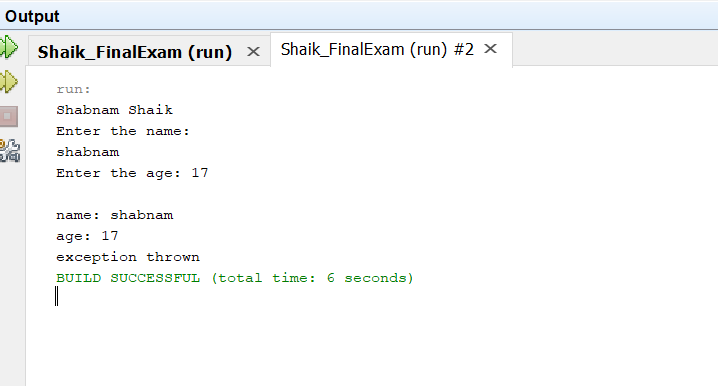
In class, declared two variables name and age. Declared getter setter and constructor. In age method throws expection when age is read as less than 18. It prints not permitted, else prints permitted. In driver class, reading the inputs and the expection is caught but using catch keyword where it prints expecton thrown.

|  |
| --- |
| /\*  \* 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\_unchecked;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Person {  private String name;  private int age;  public Person(String name, int age) {  this.name = name;  this.age = age;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  public String age() {  String s = " ";  if (age < 18) {  throw new IllegalArgumentException("not permitted");  } else {  s = ("permitted");  }  return s;  }  @Override  public String toString() {  return "\nname: " + name + "\nage: " + age;  }  }  /\*  \* 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\_unchecked;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class PersonDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws IllegalArgumentException {  // TODO code application logic here  Scanner scan = new Scanner(System.in);  System.out.println("Shabnam Shaik");  System.out.print("Enter the name: ");  String name = scan.next();  try {  System.out.print("Enter the age: ");  int age = scan.nextInt();  Person per = new Person(name, age);  System.out.println(per.toString());  System.out.println(per.age());  } catch (Exception e) {  System.out.println("exception thrown");  }  }  } |

OUTPUT1:



OUTPUT2:



1. (10-Points) Write a program that meets the following requirements:

* Creates an array with 100 randomly chosen integers.
* Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message Out of Bounds.

Answer

**Explanation**

1.Declaring an array size using length

2.Scanning elements in array

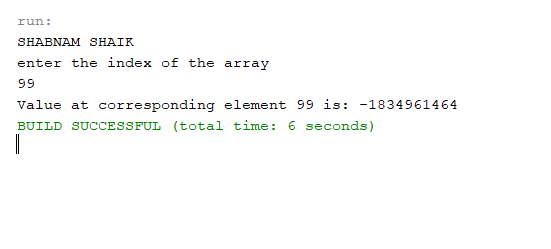
3.Scanning the required index value to be displayed

4.Using scanner class, entering what ever index we want, if the index is less than 100, then it prints the position else prints out of bounds.

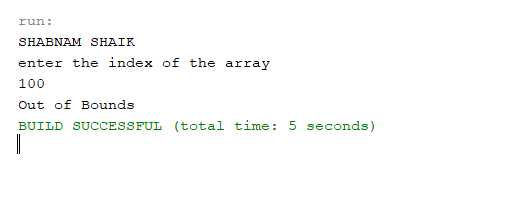
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question07;  import java.util.Random;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class ArrayDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Random rd = new Random();  int[] arr = new int[100];  for (int i = 0; i < arr.length; i++) {  arr[i] = rd.nextInt();  }  Scanner scan = new Scanner(System.in);  System.out.println("SHABNAM SHAIK");  System.out.println("enter the index of the array ");  int userInput = scan.nextInt();  if (userInput >= 0 && userInput < 100) {  System.out.println("Value at corresponding element " + userInput + " is: " + arr[userInput]);  } else {  System.out.println("Out of Bounds");  }  scan.close();  }  } |

**Output**

1.Within index



2.Out of index



1. (10-Points) What is the purpose of declaring exceptions? How do you declare an exception, and where? Can you declare multiple exceptions in a method header? Explain and demonstrate with examples.

**Exceptions** separate error handling code from regular code. They propagate errors up the call stack and these classes, group and differentiate error types.

The **purpose** of declaring exceptions is to tell the Java runtime system what can go wrong..

We can **declare exception**, using the throws keyword in the method declaration, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature. We can throw an **exception**, either a newly instantiated one or an exception that you just caught, by using the throw keyword.

We can **declare multiple exceptions**, by adding list of the exceptions, separated by commas after throws.

Example1

Explanation

In driver class, array size given as 10, and index are given from 1 to 5.

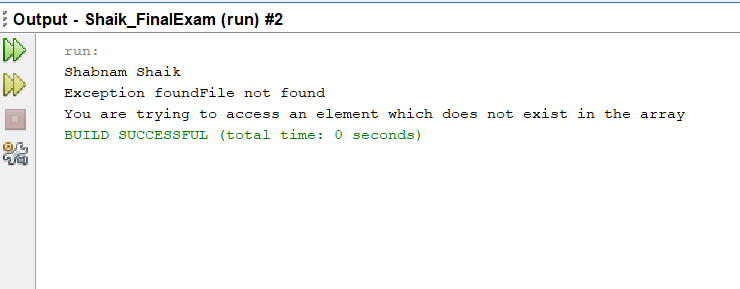
If the input file given is wrong then, it throws exception as file not found, using try and catch keywords.

Similary, given the index as 7, then it throws array index out of bounds, as the index declaration at the beginning given was only till 5.

Therefore, the above are multiple exceptions using try keyword.

|  |
| --- |
| /\*  \* 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\_example1;  import java.util.Scanner;  import java.io.File;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class EDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws Exception, IOException {  // TODO code application logic here  {  int x = 10, res;  int arr[] = {1, 2, 3, 4, 5};  try {  res = x / 0;  } catch (Exception e) {  System.out.println("Shabnam Shaik");  System.out.print("Exception found");  }  try {  File file = new File(hi.txt");  FileReader fr = new FileReader(file);  } catch (Exception FileNotFoundException) {  System.out.print("File not found\n");  }  try {  System.out.print(arr[7]);  } catch (Exception ArrayIndexOutOfBound) {  System.out.println("You are trying to access an element which does not exist in the array");  }  }  }  } |

OUTPUT



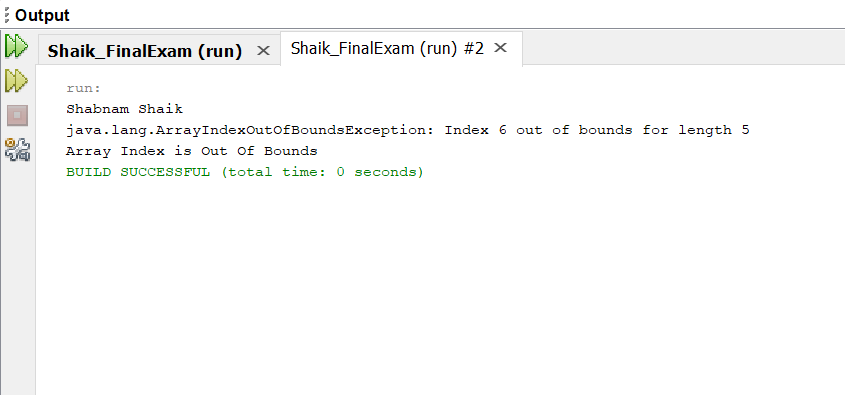
Example2

Explanation

In this I have used the try catch block , in try I have taken an array of length 5. And assigned the a[6]=9. If the index is not found within the bounds od the given length then it prints the Array Index is Out Of Bounds.

|  |
| --- |
| /\*  \* 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\_example2;  import java.io.File;  import java.io.FileNotFoundException;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class MDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws ArrayIndexOutOfBoundsException {  try {  int a[] = new int[5];  a[6] = 9;  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("Shabnam Shaik");  System.out.println(e + "\nArray Index is Out Of Bounds");  }  }  } |

OUTPUT



1. (10-Points) What is the keyword throw used for? What is the keyword throws used for? Can you throw multiple exceptions in one throw statement? Explain with examples.

Answer:

The **throws** keyword is used to declare which exceptions can be thrown from a method, while the **throw** keyword is used to explicitly throw an exception within a method or block of code. This keyword is used in methods signature and declares which exceptions can be thrown from a method.When an exception is thrown, the flow of program execution transfers from the try block to the catch block. We use the throw keyword within a method.

We use a separate try block for each statement that could throw an exception or use one try block for multiple statements that might throw **multiple exceptions.**

Example for throw exception

Explanation

In class discount, finding out the discount percentage for which I declared the variable discountPercentage in type double, also declared getter method.

In driver class using Scanner class, scanning the input value of percentage.

If at all the % is between 90-100, prints more than 90% - if the % is between 80-90, prints More than 80% - 90% , if at all it is between 80-70, prints More than 70% - 80%, else prints less than 70%.

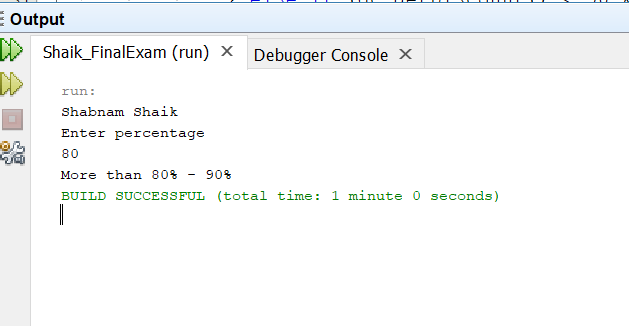
By using throw exception, if percentage is given less than 0 and greater than 100, it prints Invalid Discout Percentage.

This is catched after passing through all the if statements, at the end of the if statement by keyword catch(Exception ex)

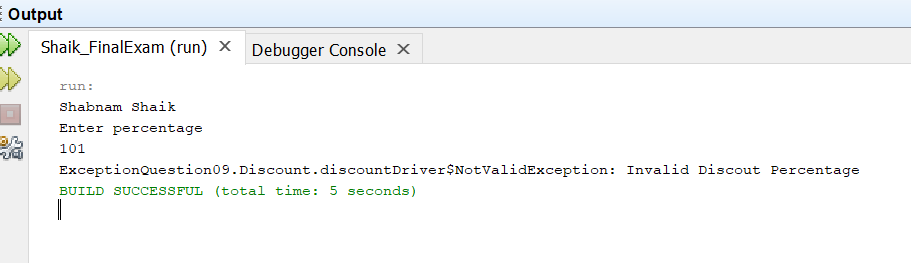
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question09.Discount;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Discount {  private double discountPercentage;  public Discount(double discountPercentage) {  this.discountPercentage = discountPercentage;  }  public double getDiscount() {  return discountPercentage;  }  }  /\*  \* 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 Question09.Discount;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class discountDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Shabnam Shaik");  System.out.println("Enter percentage");  double perc = sc.nextDouble();  try {  Discount dc = new Discount(perc);  if (dc.getDiscount() < 0 || dc.getDiscount() > 100) {  throw new NotValidException("Invalid Discout Percentage");  }  if (dc.getDiscount() >= 90 && dc.getDiscount() <= 100) {  System.out.println("More than 90% ");  } else if (dc.getDiscount() < 90 && dc.getDiscount() >= 80) {  System.out.println("More than 80% - 90% ");  } else if (dc.getDiscount() < 80 && dc.getDiscount() >= 70) {  System.out.println("More than 70% - 80% ");  } else {  System.out.println("Less than 70%");  }  } catch (Exception ex) {  System.out.println("Exception" + ex);  }  }  private static class NotValidException extends Exception {  public NotValidException() {  }  public NotValidException(String msg) {  super(msg);  }  }  } |

Output

1.Without throw Exception



2.With throw Exception



Example2 for throws exception

Explanation

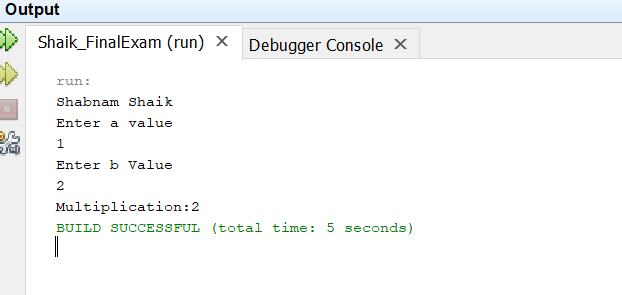
In main class, finding out the multiplication by multiplying a & b, where I scan these inputs using scanner class. If b is zero then using throw exception, it prints “b value cannot be 0”.

In this, try and catch keywords are used to catch the exceptions.

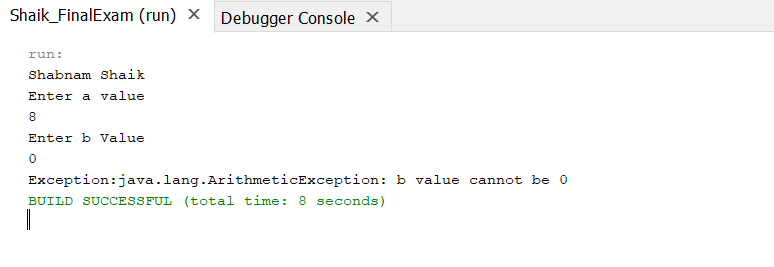
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question09;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class ThrowsDriver {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Enter a value");  int a = sc.nextInt();  System.out.println("Enter b Value");  int b = sc.nextInt();  if (b == 0) {  try {  throw new ArthmeticException("b value cannot be 0");  } catch (ArthmeticException ex) {  System.out.println("Exception:" + ex);  }  } else {  System.out.println("Multiplication:" + (a \* b));  }  }  } |

Output

1.Without throws exception



2.With throws exception



1. (15-Points) What is a recursive method? What is an infinite recursion? Explain and demonstrate with examples. Implement the search (element) in a list using recursion.

**Answer**

**Recursion** is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

If a recursion never reaches a base case, it will go on making recursive calls forever and the program will never terminate. This is known as **infinite recursion**, and it is generally not considered a good idea.

Example1 for infinite recursion:

Explanation

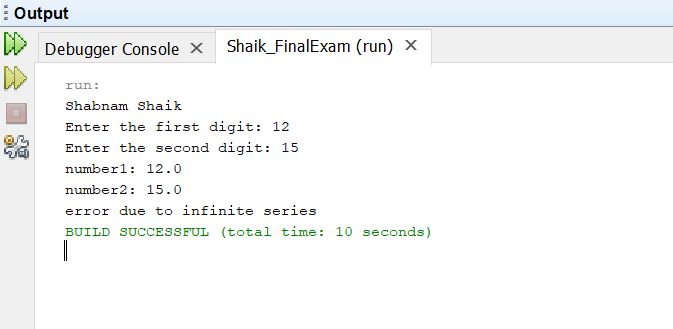
In main class declaring two digits and declaring getter and setter methods for digit1 and digit2.

In method series, if digit1 is equal to zero and also if digit2 is equal to zero, then it returns zero, else, it takes math power function to calculate the series calculation.

In driver class, using scanner class, reading the two inputs, and whatever the value maybe, it takes the output as infinite recursion as using by try and catch method we give this in print statements according to the required output.

|  |
| --- |
| /\*  \* 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\_example1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class infiniteRecursion {  private double digit1;  private double digit2;  public infiniteRecursion(double digit1, double digit2) {  this.digit1 = digit1;  this.digit2 = digit2;  }  public double getDigit1() {  return digit1;  }  public void setDigit1(double digit1) {  this.digit1 = digit1;  }  public double getDigit2() {  return digit2;  }  public void setDigit2(double digit2) {  this.digit2 = digit2;  }  public double series(int n) {  if (digit1 == 0 || digit2 == 0) {  return 0;  } else {  return series((int) ((int) Math.log(n) + (Math.pow(-2, (digit1 + digit2)))));  }  }  @Override  public String toString() {  return "number1: " + digit1 + "\nnumber2: " + digit2;  }  }  /\*  \* 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\_example1;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class InfiniteDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner sha = new Scanner(System.in);  System.out.println("Shabnam Shaik");  System.out.print("Enter the first digit: ");  double digit1 = sha.nextDouble();  System.out.print("Enter the second digit: ");  double digit2 = sha.nextDouble();  try {  infiniteRecursion inrecurs = new infiniteRecursion(digit1, digit2);  System.out.println(inrecurs.toString());  System.out.println(inrecurs.series(3));  } catch (StackOverflowError st) {  System.out.println("error due to infinite series");  }  }  } |

Output



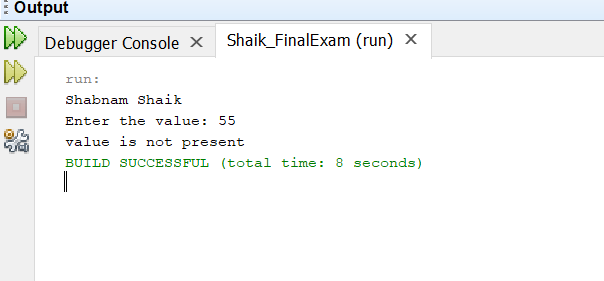
Example2 for recursive method

Explanation

In driver class, using scanner class reading the input value, if that input value matches with the already declared array list, then it prints, value is present, else it prints, value is not present.If there is no recursive search given then there is no throw new UnsupportedOperationException declared.

|  |
| --- |
| /\*  \* 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\_example2;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner scan = new Scanner(System.in);  int list[] = new int[]{101, 78, 54, 67, 89};  System.out.println("Shabnam Shaik");  System.out.print("Enter the value: ");  int value = scan.nextInt();  int h = recursiveSearch(list, 0, list.length - 1, value);  if (h != -1) {  System.out.println("value is present"  );  } else {  System.out.println("value is not present");  }  }  private static int recursiveSearch(int[] list, int s, int t, int r) {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  if (t < s) {  return -1;  }  if (list[s] == r) {  return s;  }  if (list[t] == r) {  return t;  }  return recursiveSearch(list, s + 1, t - 1, r);  }  } |

Output



1. (10-Points) Write a java program that illustrates how equals() and hashCode() methods work? Explain your code in comments.

Answer

**equals() method**

In java equals() method is used to compare equality of two Objects. The equality can be compared in two ways:2

Shallow comparison: The default implementation of equals method is defined in Java.lang.Object class which simply checks if two Object references (say x and y) refer to the same Object. i.e. It checks if x == y. Since Object class has no data members that define its state, it is also known as shallow comparison.

Deep Comparison: Suppose a class provides its own implementation of equals() method in order to compare the Objects of that class w.r.t state of the Objects. That means data members (i.e. fields) of Objects are to be compared with one another. Such Comparison based on data members is known as deep comparison.

**hashCode() method**

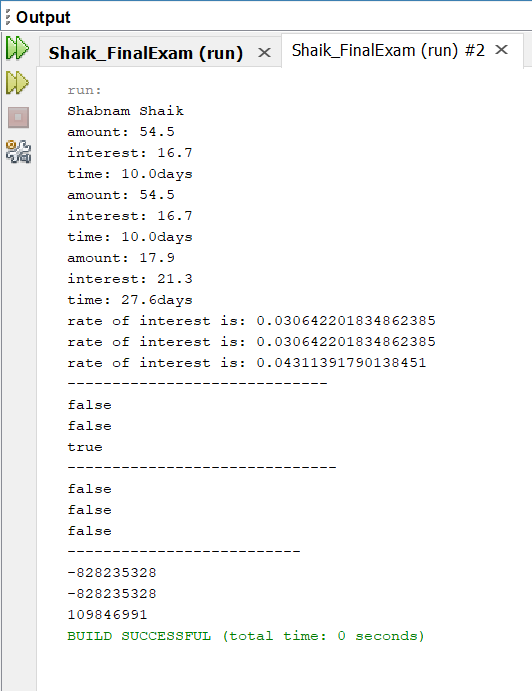
It returns the hashcode value as an Integer. Hashcode value is mostly used in hashing based collections like HashMap, HashSet, HashTable….etc. This method must be overridden in every class which overrides equals() method.

Code with explanation in comments

Example for equals() method:

|  |
| --- |
| /\*  \* 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 question11\_example1;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Bank {  private final double amount;  private final double interest;  private final double time;  /\*\*  \*  \* @param amount needed will be given  \* @param interest the rate of interest is given  \* @param time the total time or duration is taken  \*/  public Bank(double amount, double interest, double time) {  this.amount = amount;  this.interest = interest;  this.time = time;  }  /\*\*  \*  \* @return will get the ampunt  \*/  public double getAmount() {  return amount;  }  /\*\*  \*  \* @return will set interest  \*/  public double getInterest() {  return interest;  }  /\*\*  \*  \* @return will get the time  \*/  public double getTime() {  return time;  }  /\*\*  \*  \* @return will get the rate of interest  \*/  public double rateOfInterest() {  return interest / (amount \* time);  }  /\*\*  \* hashcode is overridden  \* @return  \*/  @Override  public int hashCode() {  int hash = 5;  hash = 12 \* hash + (int) (Double.doubleToLongBits(this.amount) ^ (Double.doubleToLongBits(this.amount) >>> 32));  hash = 12 \* hash + (int) (Double.doubleToLongBits(this.interest) ^ (Double.doubleToLongBits(this.interest) >>> 32));  hash = 12 \* hash + (int) (Double.doubleToLongBits(this.time) ^ (Double.doubleToLongBits(this.time) >>> 32));  return hash;  }  /\*\*  \*  \* @param obj if else statement is given to check the equalto statement  \* @return  \*/  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Bank other = (Bank) obj;  if (Double.doubleToLongBits(this.amount) != Double.doubleToLongBits(other.amount)) {  return false;  }  if (Double.doubleToLongBits(this.interest) != Double.doubleToLongBits(other.interest)) {  return false;  }  if (Double.doubleToLongBits(this.time) != Double.doubleToLongBits(other.time)) {  return false;  }  return true;  }  /\*\*  \*  \* @return toString method is overridden  \*/  @Override  public String toString() {  return "amount: " + amount + "\ninterest: " + interest + "\ntime: " + time + "days";  }  }  /\*  \* 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 question11\_example1;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class BankDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  /\*\*  \* amount ,interest ,time for three banks are given  \*/  Bank b1 = new Bank(54.5, 16.7, 10);  Bank b2 = new Bank(54.5, 16.7, 10);  Bank b3 = new Bank(17.9, 21.3, 27.6);  System.out.println("Shabnam Shaik");  /\*\*\*  \* all the values are called by the to.string  \*/  System.out.println(b1.toString());  System.out.println(b2.toString());  System.out.println(b3.toString());  System.out.println("rate of interest is: " + b1.rateOfInterest());  System.out.println("rate of interest is: " + b2.rateOfInterest());  System.out.println("rate of interest is: " + b3.rateOfInterest());  System.out.println("-----------------------------");  System.out.println(b1.equals(b3));  System.out.println(b2.equals(b3));  System.out.println(b2.equals(b1));  System.out.println("------------------------------");  System.out.println(b1 == b3);  System.out.println(b2 == b3);  System.out.println(b1 == b2);  System.out.println("--------------------------");  System.out.println(b1.hashCode());  System.out.println(b2.hashCode());  System.out.println(b3.hashCode());  }  } |

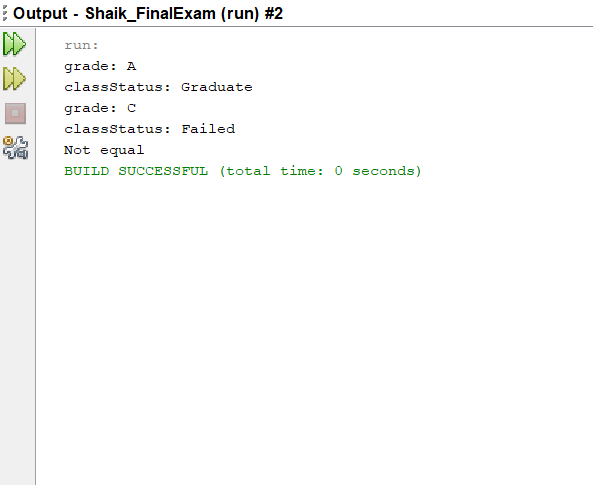
OUTPUT



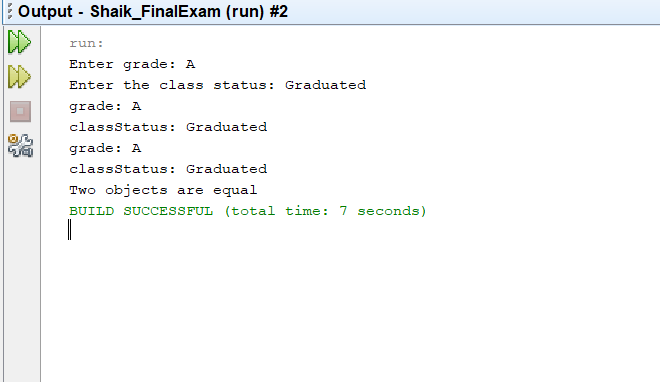
Example for hashCode() method:

|  |
| --- |
| /\*  \* 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 question11\_example2;  import java.util.Objects;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Student {  private String grade;  private String classStatus;  /\*\*  \*  \* @param grade student grade is given  \* @param classStatus of the student is given  \*/  public Student(String grade, String classStatus) {  this.grade = grade;  this.classStatus = classStatus;  }  /\*\*  \*  \* @return will get grade  \*/  public String getGrade() {  return grade;  }  /\*\*  \*  \* @param grade will set  \*/  public void setGrade(String grade) {  this.grade = grade;  }  /\*\*  \*  \* @return will get the classStatus  \*/  public String getClassStatus() {  return classStatus;  }  /\*\*  \*  \* @param classStatus is set  \*/  public void setClassStatus(String classStatus) {  this.classStatus = classStatus;  }  /\*\*  \* hashcode is overridden  \*  \* @return  \*/  @Override  public int hashCode() {  int hash = 5;  hash = 89 \* hash + Objects.hashCode(this.grade);  hash = 89 \* hash + Objects.hashCode(this.classStatus);  return hash;  }  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Student other = (Student) obj;  if (!Objects.equals(this.grade, other.grade)) {  return false;  }  if (!Objects.equals(this.classStatus, other.classStatus)) {  return false;  }  return true;  }  /\*\*  \*  \* @return toString method is overridden  \*/  @Override  public String toString() {  return "grade: " + grade + "\nclassStatus: " + classStatus;  }  }  /\*  \* 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 question11\_example2;  import java.util.Scanner;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  /\*\*  \* Two Student s1 and s2 are created and declared  \*/  Student s1 = new Student("A", "Graduate");  Student s2 = new Student("C", "Failed");  System.out.println(s1.toString());  System.out.println(s2.toString());  /\*\*  \* if else statement is taken and checks if both are equal and prints  \* the outputs accordingly  \*/  if (s1.hashCode() == s2.hashCode()) {  if (s1.equals(s2)) {  System.out.println("Two objects are equal ");  } else {  System.out.println("two objects are not equal");  }  } else {  System.out.println("Not equal ");  }  }  } |

Output when not equal:



Output when equal:



1. (15-Points) Design Employee class and Employee driver class as follows:
2. **Employee Class implements Comparable<Employee**>

* Data fields named empId, empName and empSalary
* A constructor with parameters, listed in the same order as above.
* Create getter methods for all the parameters.
* A toString method that prints the empId, empName and empSalary. There should be one space between each value output.
* Because Employee implements the Comparable interface, you must also implement the compareTo method as defined by the Comparable interface. Define this method in such a way that the natural ordering of employees will be by id number, in ascending order.

1. **EmployeeDriver Class**

* Begin by filling an ArrayList with at least 5 employees. Add employees in random order – not by id number, not by name, and not by salary. The original list should not be in order by any of these attributes.
* Use an enhanced for loop to print the original list.
* Call the one-parameter sort method of the Collections class to sort the list by its natural order (empId number) and then print the list again.
* Call the two-parameter sort method of the Collections class, supplying a new Comparator<Employee> that sorts by salary. Print the list again.
* Call the two-parameter sort method of the Collections class, supplying a new Comparator<Employee> that sorts by name. Print the list again.

Answer

Explanation

In Employee class implements Comparable<Employee> where the variables named empId, employeeName, empSalary taken. Then written the constructor for all the variables and given the getter and setter methods for them.

Written the toString methods those variables. Then overridden the compareTo method for natural ordering of the empId.

In driver class declared an array list named employee List and added the five employee details into the list , then used the print statement to print the details in order .If employee is equals to ‘s’ , the ‘s’ is printed .

Then using sorting method, if emp1 salary is less than the emp2 it returns the -1, if it is equal it returns the 0, else +1.

Then according to the sorting, it prints the salary. Then comparison between the employees takes for the names and by sorting the names are printed. Then employee ‘s’ of employee details is printed.

|  |
| --- |
| /\*  \* 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 question12;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class Employee implements Comparable<Employee> {  private int empId;  private String employeeName;  private double empSalary;  public Employee(int empId, String employeeName, double empSalary) {  this.empId = empId;  this.employeeName = employeeName;  this.empSalary = empSalary;  }  public int getEmpId() {  return empId;  }  public String getEmployeeName() {  return employeeName;  }  public double getEmpSalary() {  return empSalary;  }  @Override  public String toString() {  return "empId:" + empId + " " + "employeeName:" + employeeName + " " + "empSalary:" + empSalary;  }  @Override  public int compareTo(Employee emp) {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  int cmp = ((Integer) this.empId).compareTo(emp.empId);  return cmp;  }  }  /\*  \* 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 question12;  import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  /\*\*  \*  \* @author Shabnam Shaik  \*/  public class EmployeeDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  // TODO code application logic here  ArrayList<Employee> employee = new ArrayList<Employee>();  employee.add(new Employee(15, "mastanvali shaik", 10113.67));  employee.add(new Employee(10, "razia shaik", 90087.98));  employee.add(new Employee(03, "saajidah shaik", 80097.76));  employee.add(new Employee(07, "syed shaik", 11234.05));  employee.add(new Employee(11, "chandini shaik", 13421.54));  System.out.println("SHABNAM SHAIK");  System.out.println("Employees in actual order:");  for (Employee s : employee) {  System.out.println(s);  }  System.out.println();  // Sort the employees into natural order by emp id  Collections.sort(employee);  System.out.println("Sorting by employee id: ");  for (Employee s : employee) {  System.out.println(s);  }  System.out.println();  Collections.sort(employee, new Comparator<Employee>() {  public int compare(Employee e1, Employee e2) {  if (e1.getEmpSalary() < e2.getEmpSalary()) {  return -1;  } else if (e1.getEmpSalary() == e2.getEmpSalary()) {  return 0;  } else {  return +1;  }  }  });  System.out.println("Sorted by Salary in increasing order: ");  for (Employee s : employee) {  System.out.println(s);  }  System.out.println();  Collections.sort(employee, new Comparator<Employee>() {  public int compare(Employee s1, Employee s2) {  return (s1.getEmployeeName()).compareTo(s2.getEmployeeName());  }  });  System.out.println("Sorted Employees by name: ");  for (Employee s : employee) {  System.out.println(s);  }  System.out.println();  }  } |

Output

