**lFinal 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

Interface: The interface is a template for a class that can be used to build it. There are no concrete approaches in the gui (methods that have code). An interface's methods are all abstract methods.

It is not possible to create an interface. Groups that implement interfaces, on the other hand, can be instantiated. Instance variables are never used in interfaces, but public static final variables may be used.

Abstract**:**

Abstract class is a class that has the abstract keyword in its declaration. At least one abstract method, i.e. methods without a body, should be present in abstract groups. It may have a variety of concrete approaches. You may use abstract classes to construct blueprints for concrete classes. However, the abstract procedure must be implemented by the inheriting class. It is not possible to instantiate abstract groups.

Similarities:

1)Both abstract and interface are java basic object types.

2) Both can contain variables and methods.

3) Both be inherited using Inheritance (extends keyword for classes and implements keyword for interfaces)

Differences**:**

1) Interface can have only abstract methods. An abstract class can have abstract and non-abstract methods. From Java 8, it can have default and static methods also.

2) Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.

3) Abstract class can have final, non-final, static and non-static variables. The interface has only static and final variables.

4) An interface can extend another Java interface only; an abstract class can extend another Java class and implement multiple Java interfaces.

5) Java interface can be implemented using the keyword “implements” and an abstract class can be extended using the keyword “extends”.

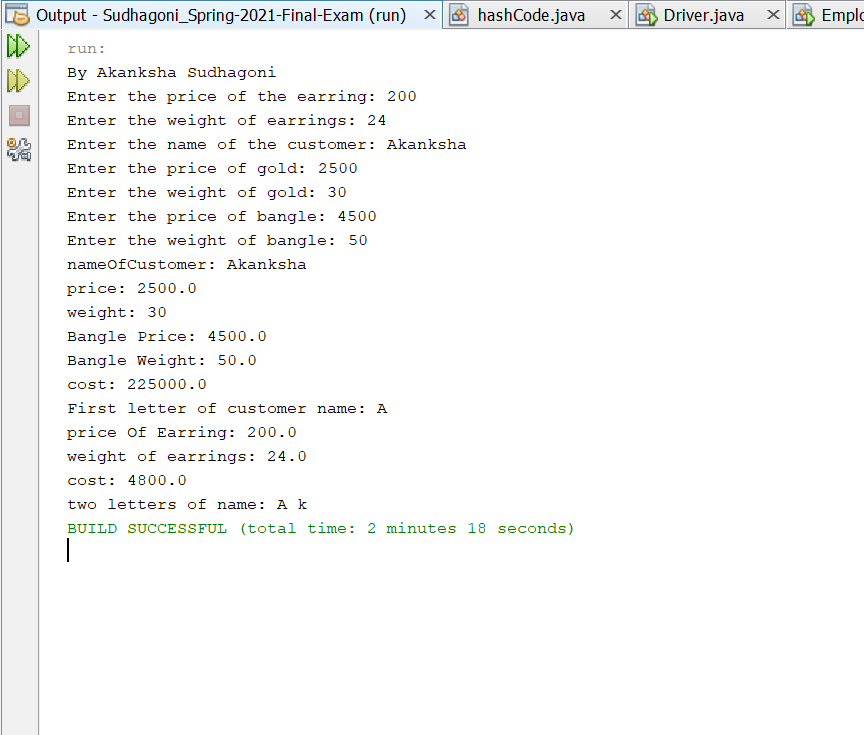
The key advantage for using an interface over an abstract class is that it avoids the diamond dilemma and allows for multiple inheritance. As a result, multiple inheritance is not allowed in Java classes. As a result, we use interfaces to achieve multiple inheritance

Explanation

An Abstract class Gold is created where all the attributes and methods are given and overridden in the toString method. And then two other classes called bangles and Earrings are created where the other methods are created and overridden which are even extended. Then a Driver class is created, and all these methods are called in the Driver class and also all the println are given in this class.

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question1;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public abstract class Gold {  private String nameOfCustomer;  private double price;  private int weight;  public Gold(String nameOfCustomer, double price, int weight) {  this.nameOfCustomer = nameOfCustomer;  this.price = price;  this.weight = weight;  }  public String getNameOfCustomer() {  return nameOfCustomer;  }  public double getPrice() {  return price;  }  public int getWeight() {  return weight;  }  public abstract double getCost();  public abstract String stringMethod();  @Override  public String toString() {  return "nameOfCustomer: " + nameOfCustomer + "\nprice: " + price + "\nweight: " + weight;  }  }  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question1;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class Bangles extends Gold {  private double BanglePrice;  private double BangleWeight;  public Bangles(double BanglePrice, double BangleWeight, String nameOfCustomer, double price, int weight) {  super(nameOfCustomer, price, weight);  this.BanglePrice = BanglePrice;  this.BangleWeight = BangleWeight;  }  public double getBanglePrice() {  return BanglePrice;  }  public double getBangleWeight() {  return BangleWeight;  }  @Override  public double getCost() {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  double cost = BanglePrice \* BangleWeight;  return cost;  }  @Override  public String stringMethod() {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return super.getNameOfCustomer().substring(0, 1);  }  @Override  public String toString() {  return super.toString() + "\nBangle Price: " + BanglePrice + "\nBangle Weight: " + BangleWeight  + "\ncost: " + getCost() + "\nFirst letter of customer name: " + stringMethod();  }  }  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question1;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class EarRings extends Gold {  private double priceOfEarring;  private double EarringsWeight;  public EarRings(double priceOfEarring, double EarringsWeight, String nameOfCustomer, double price, int weight) {  super(nameOfCustomer, price, weight);  this.priceOfEarring = priceOfEarring;  this.EarringsWeight = EarringsWeight;  }  public double getPriceOfEarring() {  return priceOfEarring;  }  public double getEarringsWeight() {  return EarringsWeight;  }  @Override  public double getCost() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return priceOfEarring \* EarringsWeight;  }  @Override  public String stringMethod() {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  return (super.getNameOfCustomer().substring(0, 1)) + " " + (super.getNameOfCustomer().substring(1, 2));  }  @Override  public String toString() {  return "price Of Earring: " + priceOfEarring + "\nweight of earrings: " + EarringsWeight  + "\ncost: " + getCost() + "\ntwo letters of name: " + stringMethod();  }  }  /\*  \* 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.Scanner;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class GoldDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner sc = new Scanner(System.in);  System.out.println("By Akanksha Sudhagoni");  System.out.print("Enter the price of the earring: ");  double priceOfEarring = sc.nextDouble();  System.out.print("Enter the weight of earrings: ");  double EarringsWeight = sc.nextDouble();  System.out.print("Enter the name of the customer: ");  String nameOfCustomer = sc.next();  System.out.print("Enter the price of gold: ");  double price = sc.nextDouble();  System.out.print("Enter the weight of gold: ");  int weight = sc.nextInt();  System.out.print("Enter the price of bangle: ");  double BanglePrice = sc.nextDouble();  System.out.print("Enter the weight of bangle: ");  double BangleWeight = sc.nextDouble();  Gold g = new EarRings(priceOfEarring, EarringsWeight, nameOfCustomer, price, weight);  Gold b = new Bangles(BanglePrice, BangleWeight, nameOfCustomer, price, weight);  System.out.println(b.toString());  System.out.println(g.toString());  }  } |

OUTPUT-

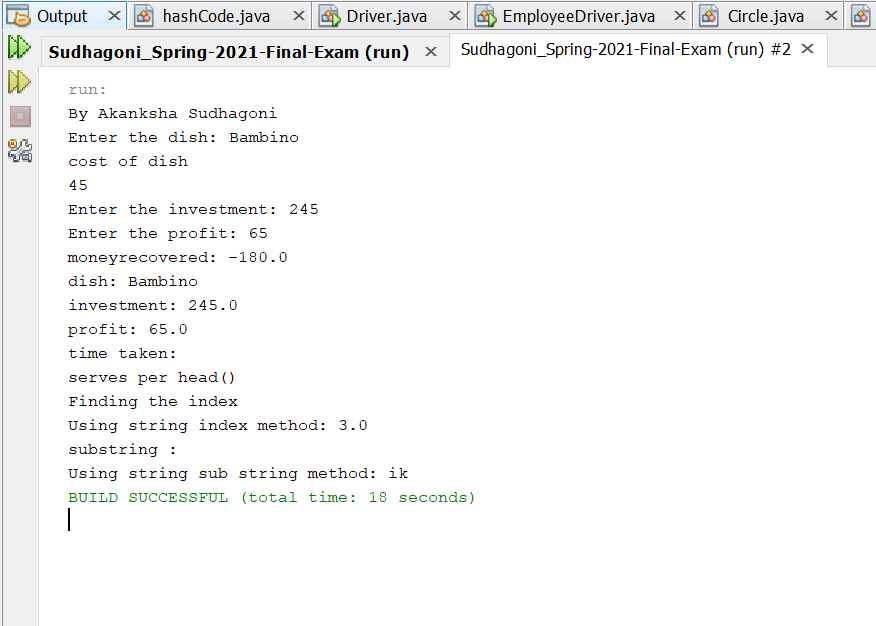


Explanation:

An interface classes Distributing and Time are created where public methods are given then a class bambino and then is created Time and Distributed are implemented all the other methods are given and overridden. Then a driver class is created where all these are called and the print statements are given as per the 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 Question01Interface01;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public interface Distributing {  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 Question01Interface01;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  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 Question01Interface01;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class Bambino implements Time, Distributing {  private final String dish;  private final double investment;  private final double profit;  public Bambino(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 Question01Interface01;  import java.util.Scanner;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class BambinoDriver {  /\*\*  \* @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(“By Akanksha Sudhagoni”);  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();  Bambino b = new Bambino(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("Akanksha"));  System.out.println("substring : ");  System.out.println(b.myst("Sudhagoni"));  }  } |

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.

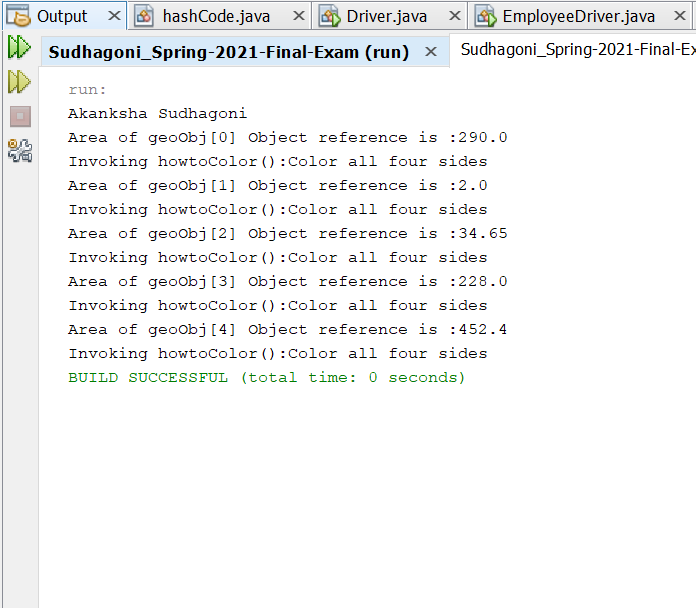
Draw a UML diagram that involves Colorable, Square, and GeometricObject. Write a test program that creates an array of five GeometricObjects. For each object in the array, display its area and invoke its howToColor method if it is colorable.

ANSWER

Explanation:

An interface Colorable is created with avoid method named howToColor().Then class GeomentricObject is created where and abstract double method getArea() is created. Then class Square extends GeomentricObject implements Colorable is created, here the attributes about the size1 and size2 are given and then finally a Driverclass named Driver is created where all these methods are called and the print Statements are also given.

|  |
| --- |
| /\*  \* 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 Question\_02;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public interface Colorable {  public 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 Question\_02;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public abstract class GeomentricObject {  public abstract double getArea();  public void howToColor() {  System.out.println("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 Question\_02;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class Square extends GeomentricObject implements Colorable {  private double side1, side2;  public Square(double side1, double side2) {  this.side1 = side1;  this.side2 = side2;  }  public double getSide1() {  return side1;  }  public double getSide2() {  return side2;  }  @Override  public double getArea() {  return getSide1() \* getSide2();  }  @Override  public void howToColor() {  System.out.println("Color all four sides");  }  }  /\*  \* 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 Question\_02;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Akanksha Sudhagoni");  GeomentricObject[] geoObj = new GeomentricObject[5];  geoObj[0] = new Square(10, 29);  geoObj[1] = new Square(1, 2);  geoObj[2] = new Square(3.3, 10.5);  geoObj[3] = new Square(12, 19);  geoObj[4] = new Square(13, 34.8);  for (int i = 0; i < geoObj.length; i++) {  System.out.println("Area of geoObj[" + i + "]" + " Object reference is :" + geoObj[i].getArea());  System.out.print("Invoking howtoColor():");  geoObj[i].howToColor();  }  }  } |

Output: 

Diagram

Description automatically generated

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

ANSWER

Assigning a value of one primitive data type to another type is Casting.

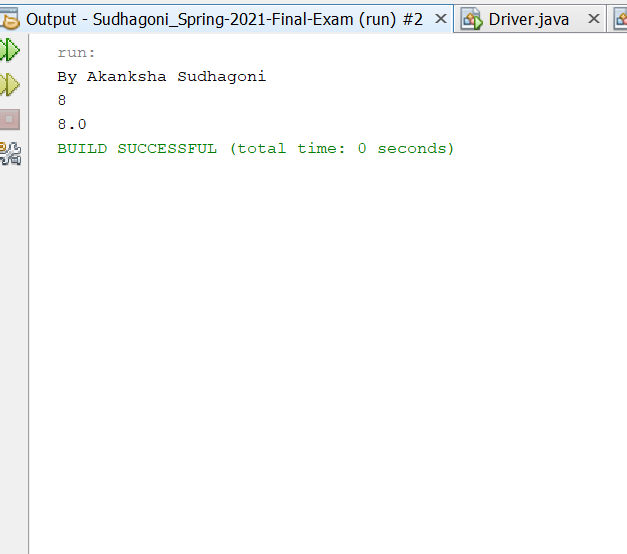
They are two types of casting:

**1.Widening Casting** (automatically) - 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.

Explanation: Here a driver class is created where I gave (int) myDouble in which converts smaller size variables to larger type. And the Double value is converted into a integer number.

|  |
| --- |
| /\*  \* 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\_wideningExample;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class DriverWidening {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int myInt = 8;  double myDouble = myInt;  System.out.println("By Akanksha Sudhagoni");  System.out.println(myInt);  System.out.println(myDouble);  }  } |

Output: 

1. **Narrowing Casting** (manually) -converting a larger type to a smaller size

type

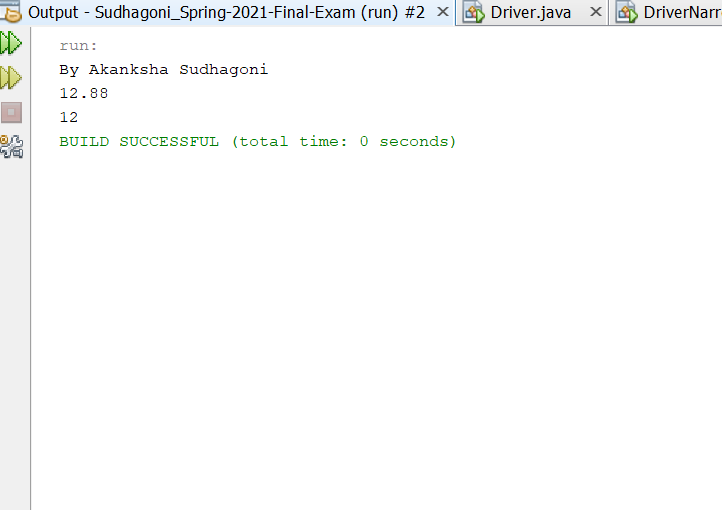
double -> float -> long -> int -> char -> short -> byte.

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.

Explanation: A Driver class is created named Narrowing, here a double value is converted into integer with giving (int) my Double. Which converts larger value into smaller value.

|  |
| --- |
| /\*  \* 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\_wideningExample;  /\*\*  \*  \* @author akanksha Sudhagoni  \*/  public class DriverNarrowing {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  double myDouble = 12.88;  int myInt = (int) myDouble;  System.out.println("By Akanksha Sudhagoni");  System.out.println(myDouble);  System.out.println(myInt);  }  } |

OUTPUT-



4.(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.

a. Is fruit instanceof Fruit?

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

b. Is fruit instanceof Orange?

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

c. Is fruit instanceof Apple?

Answer- We know that GoldenDelicious is the sub class of Apple.Instance of sub class is an instance of super class

d. Is fruit instanceof GoldenDelicious?

Answer- Yes,fruit is an instance of GoldenDelicious

e. 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

f. Is orange instanceof Orange?

Answer- Yes,orange is instance of Orange.

g. Is orange instanceof Fruit?

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

h. Is orange instanceof Apple?

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

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

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

Answer- Illegal Statement,super class(Apple) instance cannot be assigned to sub-class reference(McIntosh)

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

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 possible

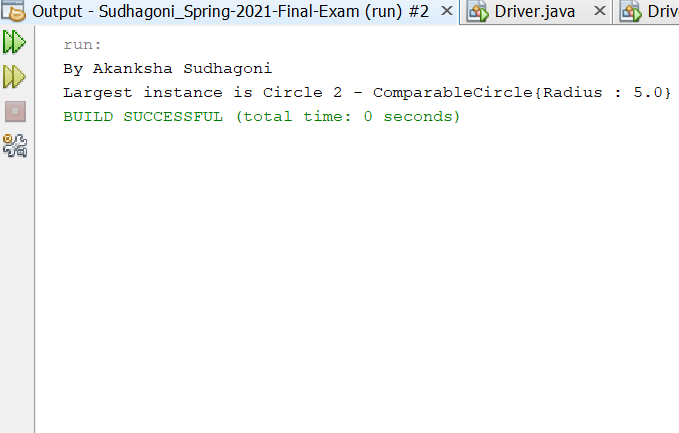
5.(10-Points) Define a class named ComparableCircle that extends Circle and implements Comparable. Draw the UML diagram and implement the compareTo method to compare the circles on the basis of area. Write a test class to find the larger of two instances of ComparableCircle objects.

Explanation:

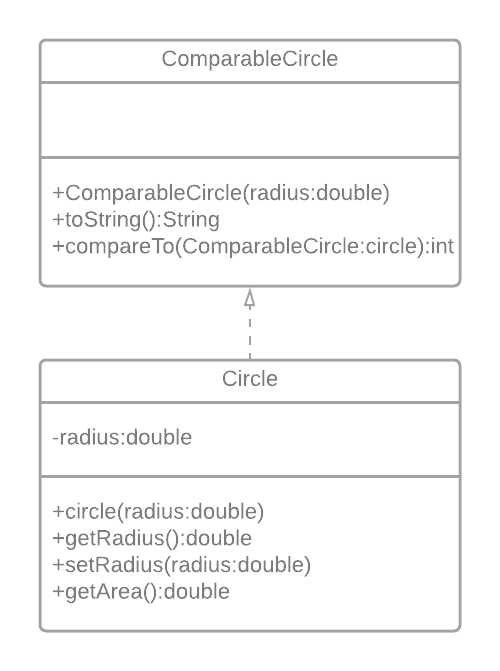
Firstly a class is created named circle where methods to get the area and the radius are given and then comparablecircle class is created where we compare the radius and area of the circles Then a driver class is created here all these methods are called and then the print statements are given to give us the 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 Question05;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  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 getArea() {  return Math.PI \* 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 Akanksha Sudhagoni  \*/  public class ComparableCircle extends Circle implements Comparable<ComparableCircle> {  ComparableCircle(double radius) {  super(radius);  }  @Override  public String toString() {  return "ComparableCircle{Radius : " + getRadius() + "}";  }  @Override  public int compareTo(ComparableCircle circle) {  return Double.compare(getArea(), circle.getArea());  }  }  /\*  \* 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 Akanksha Sudhagoni  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ComparableCircle circle1 = new ComparableCircle(2.0);  ComparableCircle circle2 = new ComparableCircle(5);  System.out.println("By Akanksha Sudhagoni");  System.out.println("Largest instance is " + findLargest(circle1, circle2));  }  public static String findLargest(ComparableCircle c1, ComparableCircle c2) {  switch (c1.compareTo(c2)) {  case -1:  return "Circle 2 - " + c2.toString();  default:  return "Circle 1 - " + c1.toString();  }  }  } |

OUTPUT-



UML:



6.(15-Points) What is an exception? What are checked and unchecked exceptions? Explain and demonstrate with examples.

ANSWER

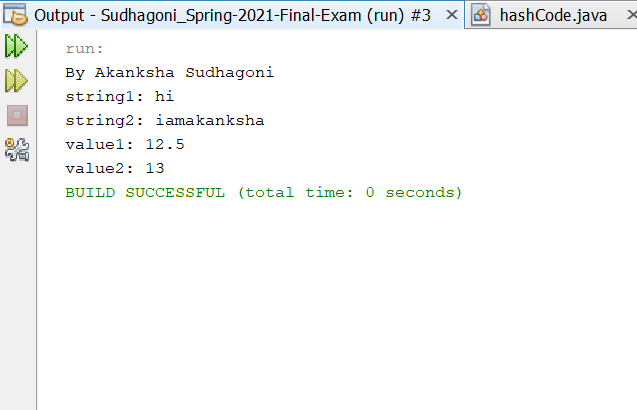
A special case is an occasion, which happens during the execution of a program, that disturbs the ordinary progression of the program's guidelines. At the point when a mistake happens inside a technique, the strategy makes an article and hands it off to the runtime framework. ... This square of code is called a special case controller.

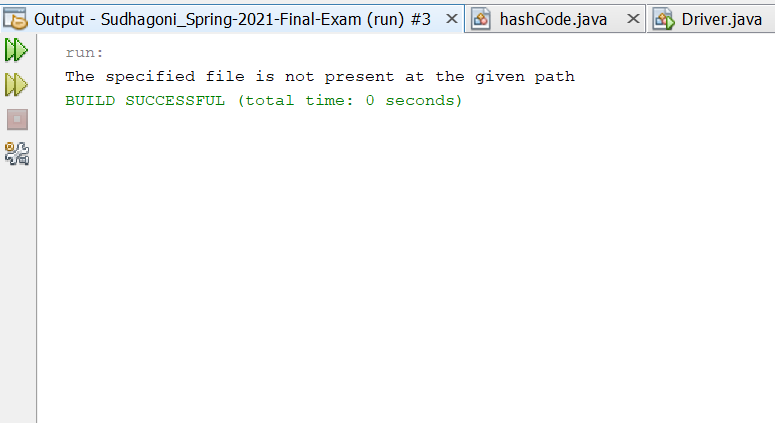
1)Checked exception:  are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

Explanation: Initailly a class is created where two string variables and a double value and long value variables are given,then the constructed and getter and setter methods for each of them are created along with the tostring method.Later a driver class is created named check where a input file having some input is added which is also added into the netbeans project into the package, this file if called correctly the put is given else a exception throw which then enters the try catch statement and prints that the specified output is 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 Question06;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class CheckedClass {  private String string1;  private String string2;  private double value1;  private long value2;  public CheckedClass(String string1, String string2, double value1, long value2) {  this.string1 = string1;  this.string2 = string2;  this.value1 = value1;  this.value2 = value2;  }  public String getString1() {  return string1;  }  public void setString1(String string1) {  this.string1 = string1;  }  public String getString2() {  return string2;  }  public void setString2(String string2) {  this.string2 = string2;  }  public double getValue1() {  return value1;  }  public void setValue1(double value1) {  this.value1 = value1;  }  public long getValue2() {  return value2;  }  public void setValue2(long value2) {  this.value2 = value2;  }  @Override  public String toString() {  return "string1: " + string1 + "\nstring2: " + string2 + "\nvalue1: " + value1 + "\nvalue2: " + value2;  }  }  /\*  \* 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 Question06;  import java.io.\*;  import java.util.Scanner;  /\*\*  \*  \* @author Akanksha sudhagoni  \*/  public class CheckedDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws FileNotFoundException {  // TODO code application logic here  try {  Scanner sc = new Scanner(new File("inputdoc.txt"));  String string1 = sc.next();  String string2 = sc.next();  double value1 = sc.nextDouble();  long value2 = sc.nextLong();  System.out.println("By Akanksha Sudhagoni");  CheckedClass cc = new CheckedClass(string1, string2, value1, value2);  System.out.println(cc.toString());  } catch (FileNotFoundException fnfe) {  System.out.println("The specified file is not "  + "present at the given path");  }  }  } |

Output-





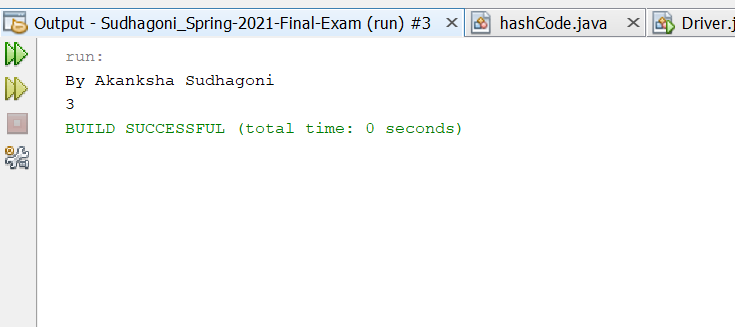
2)Unchecked exception: are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.  
In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

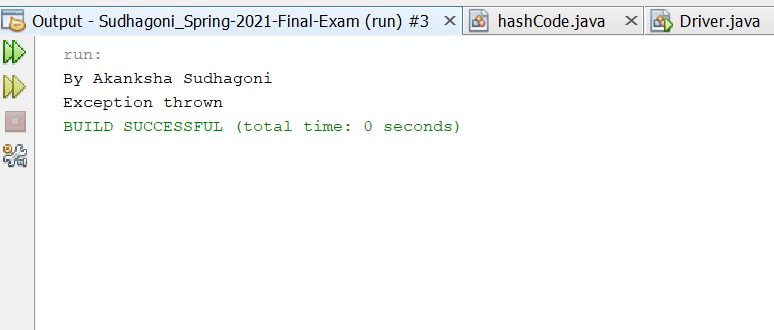
Explanation:

A driver class named UncheckedDriver is created where a list of array is given ,when a number is choose from the array list the output is printed by showing the number else if the number is not in the arraylist then a exception of ArrayIndexOutOfBoundsException ar is thrown and output Exception 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 Question06;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class UncheckedDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  try {  System.out.println("By Akanksha Sudhagoni");  int array[] = {1, 2, 3, 4, 5};  System.out.println(array[2]);  } catch (ArrayIndexOutOfBoundsException ar) {  System.out.println("Exception thrown");  }  }  } |

Output:





7.(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.

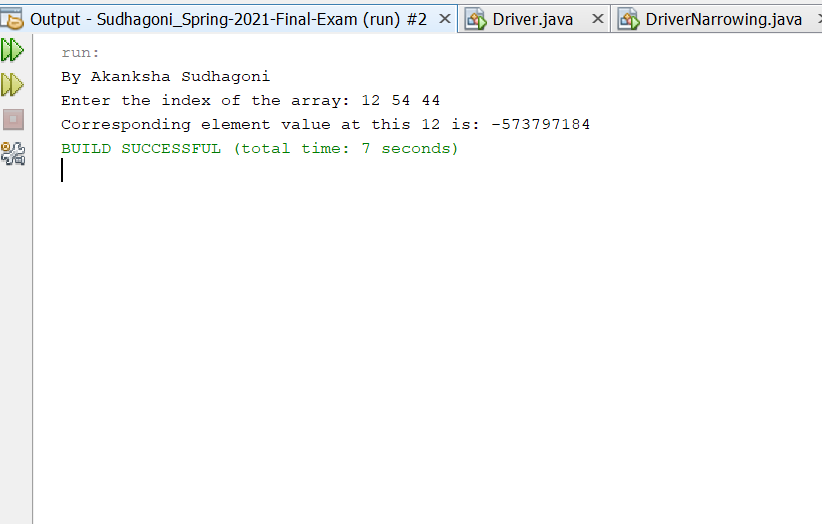
ANSWER

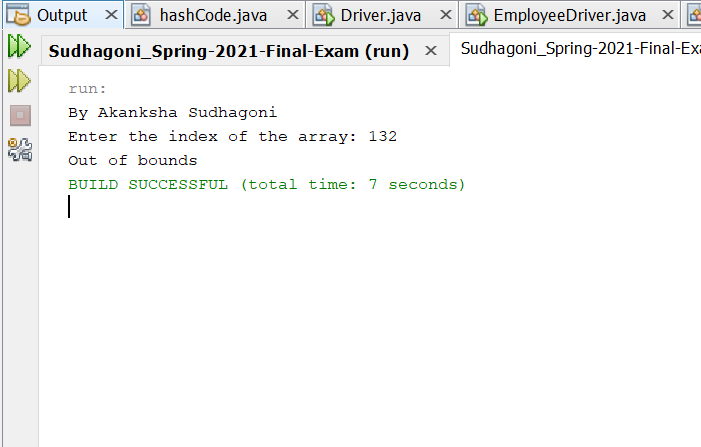
Explanation:

A Driver Class is created where a random array listen of 100 integers is given and then if a random number will be selected within the index of those it will tell if it uis inside the bounds or outstide the bounds.

|  |
| --- |
| package Question07;  import java.util.Random;  import java.util.Scanner;  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Random rand = new Random();  int[] array = new int[100];  for (int i = 0; i < array.length; i++) {  array[i] = rand.nextInt();  }  Scanner nacs = new Scanner(System.in);  System.out.println("By Akanksha Sudhagoni");  System.out.print("Enter the index of the array: ");  int index = nacs.nextInt();  try {  System.out.println("Corresponding element value at this " + index + " is: " + array[index]);  } catch (ArrayIndexOutOfBoundsException ex) {  System.out.println("Out of bounds");  }  }  } |

Output:





8.(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.

ANSWER

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

A user has entered an invalid data.

A file that needs to be opened cannot be found.

A network connection has been lost in the middle of communications or the JVM has run out of memory.

If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature. You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the throw keyword.

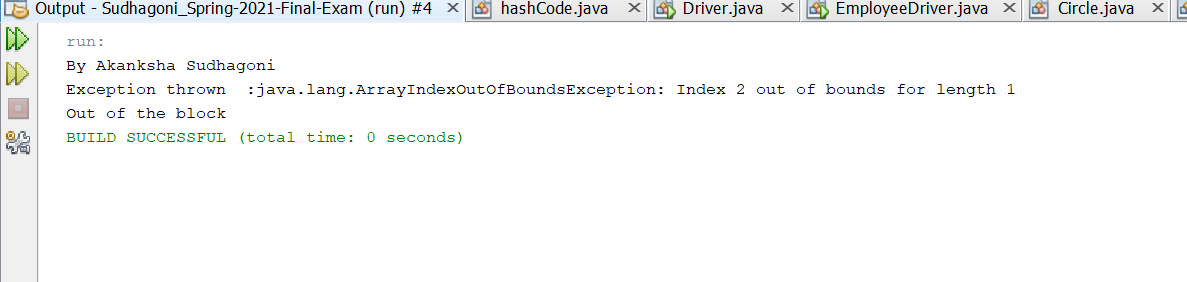
Yes, declare multiple exceptions in a method header are possible. If the method declares multiple exceptions, add list of the exceptions, separated by commas, after throws.

Explanation:

In the driver class, set the array size to 2, and if the user changes it to 3 during execution, the program will catch an exception and throw an Array Index Out Of Bounds Exception. I set the array size to 3 in the program below, which is why the output is outside the block, along with the thrown exception.

|  |
| --- |
| /\*  \* 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;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class ExcepTest {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  try {  int a[] = new int[1];  System.out.println("By Akanksha Sudhagoni");  System.out.println("Access element three :" + a[2]);  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("Exception thrown :" + e);  }  System.out.println("Out of the block");  }  } |

Output:

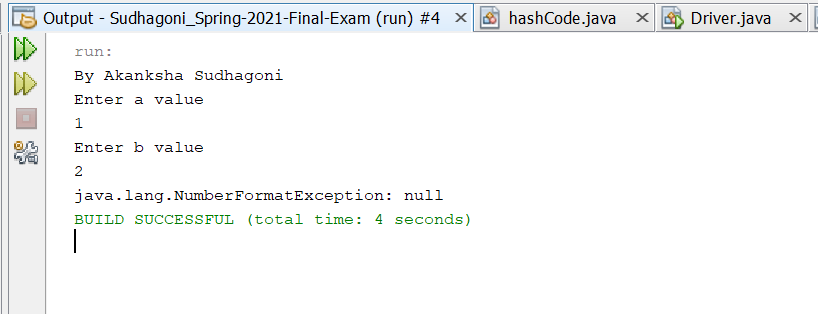


Multiple Exception:

Explanation: A class named multipleException is taken and two different variables are declared and taken from the input using the scanner object ,and the arithmetic exception and number formating exception are given and checked and the output is given accordingly.

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class MultipleException {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  try {  Scanner sc = new Scanner(System.in);  System.out.println("By Akanksha Sudhagoni");  System.out.println("Enter a value");  int a = sc.nextInt();  System.out.println("Enter b value");  int b = sc.nextInt();  method(a, b);  } catch (Exception ex) {  System.out.println(ex);  }  }  private static void method(int integerA, int integerB) throws ArithmeticException, NumberFormatException {  if (integerB == 0) {  throw new ArithmeticException("Arithmetic Exception");  }  integerA = Integer.parseInt(null);  throw new NumberFormatException("NumberFormat Exception");  }  } |

OUTPUT-



9.(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. The throws keyword is used in a method signature and declares which exceptions can be thrown from a method.

The throws keyword specifies the exceptions may be thrown from a method, while the throw keyword throws an exception directly within a method or block of code.

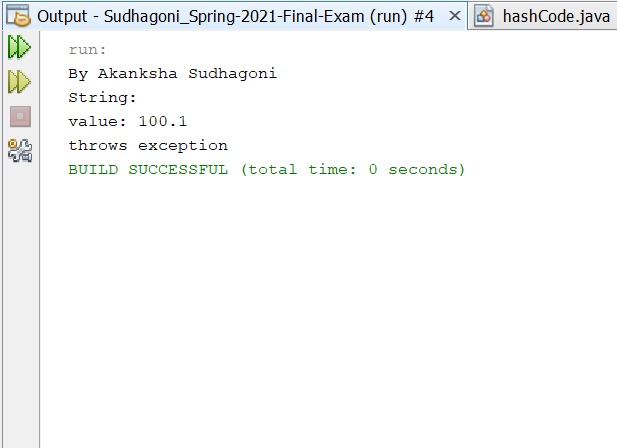
If your code throws more than one exception, you can choose if you want to: use a separate try block for each statement that could throw an exception or. use one try block for multiple statements that might throw multiple exceptions.

Explanation:

A class named ThrowExcep is created where two variables are declared and then the respective constructor and getter,setter methods are given.then a condition is given where if the string is empty Exception thrown is printed else String is printed.In the driver class these methods are called and the respective print statements are given accordingly.

|  |
| --- |
| /\*  \* 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;  /\*\*  \*  \* @author Akankhsa Sudhagoni  \*/  public class ThrowExcep {  private String String;  private double value;  public ThrowExcep(String String, double value) {  this.String = String;  this.value = value;  }  public String getString() {  return String;  }  public void setString(String String) {  this.String = String;  }  public double getValue() {  return value;  }  public void setValue(double value) {  this.value = value;  }  public String throwexcep() {  String st = " ";  if (String.equals(" ")) {  throw new NullPointerException("Exception thrown");  } else {  st = "String is printed";  }  return st;  }  @Override  public String toString() {  return "String: " + String + "\nvalue: " + value;  }  }  /\*  \* 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;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ThrowExcep th = new ThrowExcep(" ", 100.1);  try {  System.out.println(th.toString());  System.out.println(th.throwexcep());  } catch (NullPointerException e) {  System.out.println("throws exception");  }  }  } |

OUTPUT-



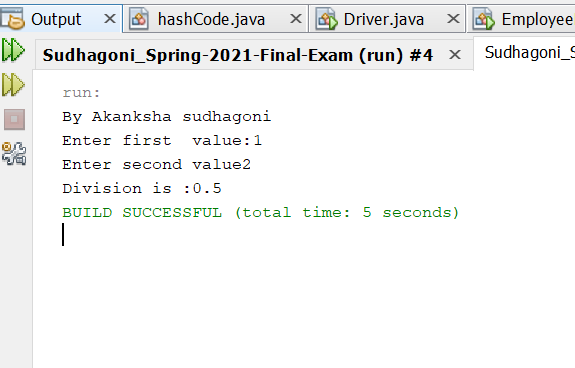
Example2:

Explanation:

Finding the Division is done in main class by dividing a and b, and I scan these inputs using scanner class. If b is zero, it prints "b value cannot be 0" using the throw exception.

To capture the exceptions, try to catch keywords are used.

|  |
| --- |
| /\*  \* 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 Akanksha Sudhagoni  \*/  public class ThrowsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("By Akanksha sudhagoni");  Scanner sc = new Scanner(System.in);  System.out.print("Enter first value:");  double x = sc.nextInt();  System.out.print("Enter second value");  double y = sc.nextInt();  if (y == 0) {  try {  throw new ArithmeticException("second value is not equal to 0");  } catch (ArithmeticException ex) {  System.out.println(ex);  }  } else {  System.out.println("Division is :" + (x / y));  }  }  } |

Output:

10.(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. In most programming environments, a program with an infinite recursion will not really run forever.

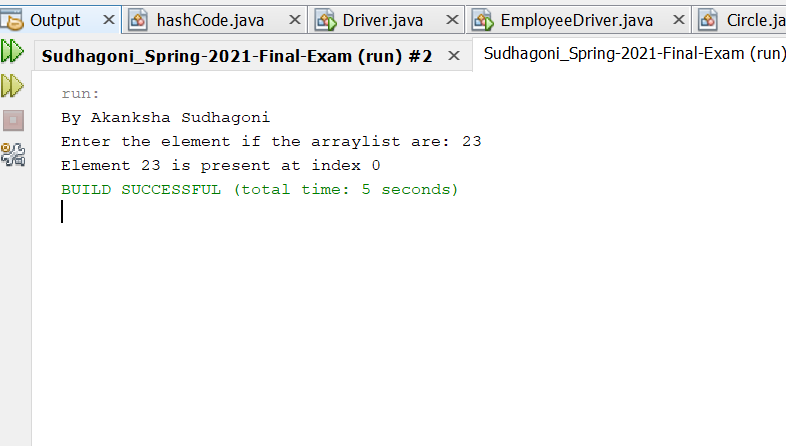
Recursion Example:

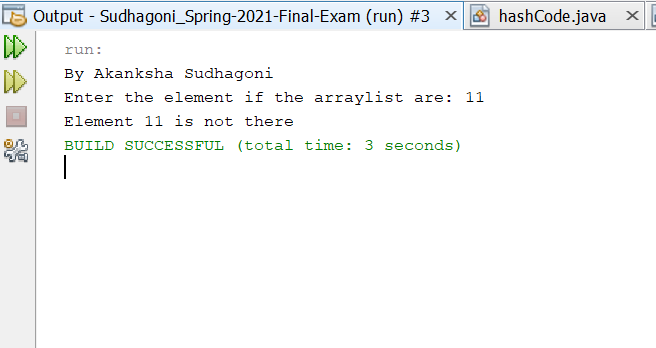
Explanation:

A class RecursiveDriver is created where a array is given and then a method to find the recursive is given, where when a element is given it searches if the element is their in the array or no.If the element is seen then the index of that element is printed else it tells that the elemet is not there.

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \* @author Akanksha Sudhagoni  \*  \*/  public class RecursionDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Scanner nacs = new Scanner(System.in);  int array[] = new int[]{23, 160, 8, 30, 10};  System.out.println("By Akanksha Sudhagoni");  System.out.print("Enter the element if the arraylist are: ");  int x = nacs.nextInt();  int index = recursiveSearch(array, 0, array.length - 1, x);  if (index != -1) {  System.out.println("Element " + x + " is present at index "  + index);  } else {  System.out.println("Element " + x + " is not there");  }  }  private static int recursiveSearch(int[] arr, int i, int j, int k) {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  if (j < i) {  return -1;  }  if (arr[i] == k) {  return i;  }  if (arr[j] == k) {  return j;  }  return recursiveSearch(arr, i + 1, j - 1, k);  }  } |

Output-





InfiniteRecursion:

Explanation:

A main Driver class is created named infiniteRecursion where loop keeps running as shown in the output. To demonstrate infinite recursion, we used a factorial example.

To begin, infiniteRecursion is a function.

The main() function invokes Factorial(num).

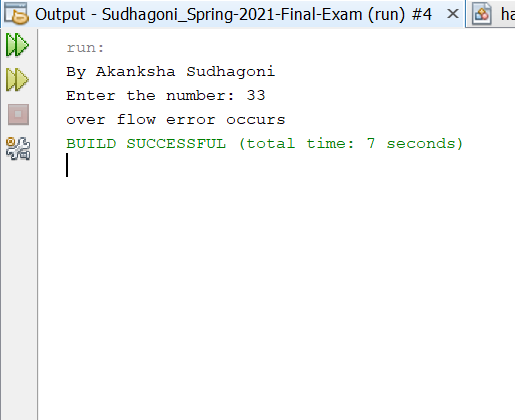
We set the num , and it begins to follow the method, but since there is no condition to stop it, it continues to run. We can see from the above illustration that the process is never-ending. This occurs because the process (infiniteRecursionFactorial(int num)) lacks a base condition.

The condition is not fulfilled at any point in this situation.

Since there is no endpoint, recursive calls can be made indefinitely

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class InfiniteRecursionDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("By Akanksha Sudhagoni");  Scanner sc = new Scanner(System.in);  try {  System.out.print("Enter the number: ");  int num = sc.nextInt();  long factorial = infiniteRecursion(num);  System.out.println(num + "! = " + factorial);  sc.close();  } catch (StackOverflowError e) {  System.out.println("over flow error occurs");  }  }  // example for infinite recursion  /\*\*  \*  \* @param num  \* @return  \*/  public static long infiniteRecursion(int num) {  return num \* infiniteRecursion(num - 1);  }  } |

Output:



11.(10-Points) Write a java program that illustrates how equals() and hashCode() methods work? Explain your code in comments.

ANSWER

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

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

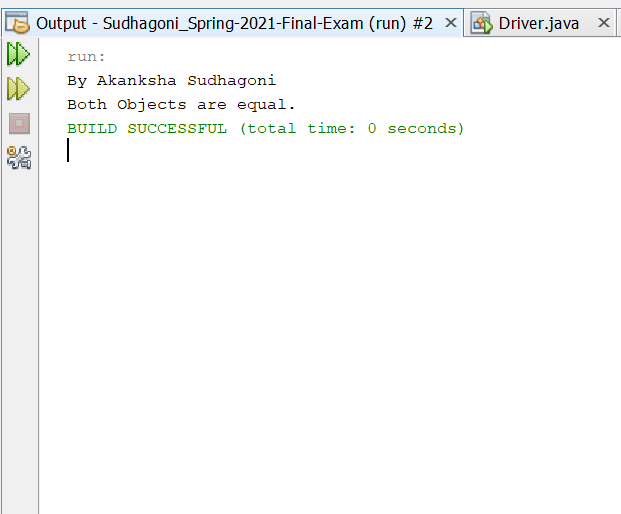
It returns the hashcode value as an Integer. Hashcode value is mostly used in hashing based collections like HashMap, HashSet, HashTable….etc. This method must be overridden in every class which overrides equals() method.

Explanation:

A class called Hashcode is generated with the attributes name, age, and weight, and then the values in the driver class are compared to the values in our regular class and told whether they are equal or not using our equals() and hashcode() methods.

|  |
| --- |
| /\*  \* 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;  /\*\*  \*  \* @author akanksha Sudhagoni  \*/  public class hashCode {  public String name;  public int id;  hashCode(String name, int id) {  this.name = name;  this.id = id;  }  @Override  public boolean equals(Object obj) {  // checking if both the object references are  // referring to the same object.  if (this == obj) {  return true;  }  // it checks if the argument is of the  // type hash by comparing the classes  // of the passed argument and this object.  // if(!(obj instanceof hash)) return false; ---> avoid.  if (obj == null || obj.getClass() != this.getClass()) {  return false;  }  // type casting of the argument.  hashCode hash = (hashCode) obj;  // comparing the state of argument with  // the state of 'this' Object.  return (hash.name == this.name && hash.id == this.id);  }  @Override  public int hashCode() {  // We are returning hash\_id  // as a hashcode value.  // we can also return some  // other calculated value or may  // be memory address of the  // Object on which it is invoked.  // it depends on how you implement  // hashCode() method.  return this.id;  }  }  /\*  \* 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;  /\*\*  \*  \* @author Aaknksha Sudhagoni  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  hashCode g1 = new hashCode("aa", 1);  hashCode g2 = new hashCode("aa", 1);  System.out.println("By Akanksha Sudhagoni");  // comparing above created Objects.  if (g1.hashCode() == g2.hashCode()) {  if (g1.equals(g2)) {  System.out.println("Both Objects are equal. ");  } else {  System.out.println("Both Objects are not equal. ");  }  } else {  System.out.println("Both Objects are not equal. ");  }  }  } |

Output:



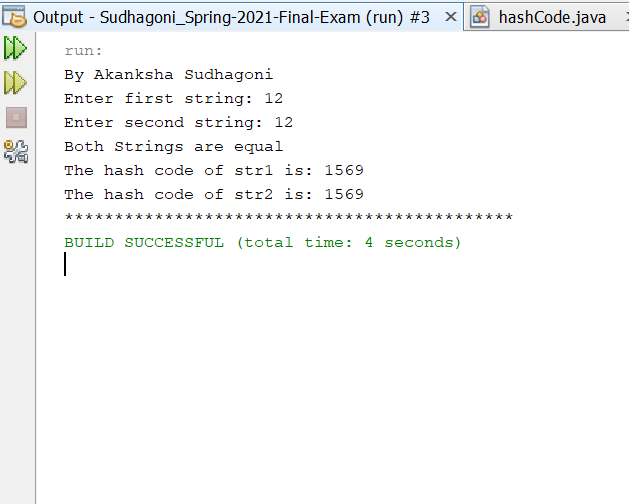
Example2:

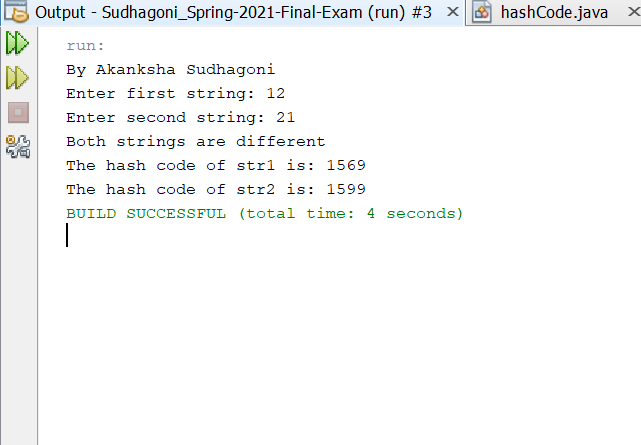
Explanation:

Initially, a driver class named Test Hash is formed, in which I give strings named a and b at the start, and both of the values are equal, and then they are tested using hashcode() and equals(), and the result is returned, indicating whether they are equal or not. The same goes for the other two string values, which are actually given.

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \*  \* @author Akanksha Sudhagoni  \*/  public class HashEqualsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  /\*\*  \* Scanner object is given which takes the values from the keyboard  \*/  Scanner scan = new Scanner(System.in);  System.out.println("By Akanksha Sudhagoni");  /\*\*  \* the first strings are told to take by using the scan.next object and the print statements are given  \*/  System.out.print("Enter first string: ");  String str1 = scan.next();  /\*\*  \* the second strings are told to take by using the scan.next object and the print statements are given  \*/  System.out.print("Enter second string: ");  String str2 = scan.next();  /\*\*  \* the two strings are compared by .equals() and their respective hashcodes are given by using the if else statement  \*/  if (str1.equals(str2)) {  System.out.println("Both Strings are equal");  System.out.println("The hash code of str1 is: " + str1.hashCode());  System.out.println("The hash code of str2 is: " + str2.hashCode());  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  } else {  System.out.println("Both strings are different");  System.out.println("The hash code of str1 is: " + str1.hashCode());  System.out.println("The hash code of str2 is: " + str2.hashCode());  }  }  } |

Output-





12.(15-Points) Design Employee class and Employee driver class as follows:

1. **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.

Explanation:

In The Employee class is responsible for implementing Where the variables empId, employeeName, and empSalary are used. The constructors for all the variables were then written, as well as getter and setter methods.

The toString methods for those variables have been published. The compareTo approach was then overridden for natural empId ordering.

The five employee descriptions were added to an array list called employee List in the driver class, which was then printed in order using the print statement. If employee equals the letter ‘s,' the letter ‘s' is printed.

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

The sala is then printed based on the sorting.

|  |
| --- |
| /\*  \* 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 Akanksha Sudhagoni  \*/  public class Employee implements Comparable<Employee> {  private int empId;  private String empName;  private int empSalary;  public Employee(int empId, String empName, int empSalary) {  this.empId = empId;  this.empName = empName;  this.empSalary = empSalary;  }  public int getEmpId() {  return empId;  }  public void setEmpId(int empId) {  this.empId = empId;  }  public String getEmpName() {  return empName;  }  public void setEmpName(String empName) {  this.empName = empName;  }  public int getEmpSalary() {  return empSalary;  }  public void setEmpSalary(int empSalary) {  this.empSalary = empSalary;  }  @Override  public String toString() {  return empId + " " + empName + " " + empSalary;  }  @Override  public int compareTo(Employee o) {  return Integer.compare(getEmpId(), o.getEmpId());  }  } |

