**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 class:** An abstract class is a class that is declared as abstract. It may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. They provide some functionalities across a set of relates classes while also allowing default method implementations. To create an abstract class Abstract keyword is used before the class keyword. The class is now abstract, but it still has three fields, seven methods, and one constructor. The constructor inside the abstract class can only be called during constructor chaining i.e. when we create an instance of sub-classes. This is also one of the reasons abstract class can have a constructor.

**Interfaces:** An interface in the Java programming language is an abstract type that is used to specify a behavior that classes must implement. Interfaces are declared using the interface keyword and may only contain method signature and constant declarations.  All methods of an Interface do not contain implementation (method bodies) as of all versions below Java 8. Interfaces cannot be instantiated but can be implemented . A class that implements an interface must implement all of the non-default methods described in the interface or be an abstract class

**Similarities between abstract classes and interface:**

1. By using both we can go for dynamic polymorphism.
2. Both contain static and final variable.
3. Both are inherited from same domain itself using extends keyword.
4. They provide static method implementation.
5. Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods.

**Difference between abstract classes and interface:**

1. Abstract class can **have abstract and non-abstract** methods, but interfaces have only abstract methods, because it has default and static methods.
2. Abstract classes does not support multiple inheritance, whereas interface supports the multiple inheritance.
3. Abstract classes will have static , non-static, final, non-final variables. Interfaces will have only final and static variables.
4. Abstract class can provide the implementation of interface, interface cannot provide the implementation of abstract class .
5. A Java abstract class can have class members like private, protected. But interfaces are protected by default.

**Interfaces are preferred over abstract classes:**

An abstract class allows us to create functionality that subclasses can implement or override. An interface is better than an abstract class when you want multiple classes to implement that interface and when you don't have to inherit default behavior.

**Example for abstract:**

**Description:**

**Boat class:** Here, the boat class is an abstract **class** and the instance variable of this class is boat capacity and boat cost . And the getter and setter methods for the boat capacity and boat cost are written. . Boat life is an abstract method.

**Deck boat:** the deck boat is a subclass of boat which deck boat class extends the variable and methods of boat class. Variable in the deck boat are speed and type. Here the boat life method

gives the life span of the boat by using the type of boat. And the toString method is uses to display the variables.

**Home boats:** the home boat is subclass of boat like deck boat and the variables in the home boat class are velocity and distance travelled. Here, the boat life method to know the life span of boat as distance travelled is greater than and equal 100 then the life span is 10 and the distance travelled is less than 100 then the life span is 20 and the to string method is uses to display the variables.

**Driver class:** In the driver class we will give the values trough the scanner object and boat1 and boat2 are the objects for deck boat and home boat respectively which is used to call the class and

the print the values.

**Program:**

|  |
| --- |
| /\*  \* 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 question01;  /\*\*  \*  \* @author prasanna arla  \*/  public abstract class Boats {  private int boatCapacity;  private double boatCost;  public Boats(int boatCapacity, double boatCost) {  this.boatCapacity = boatCapacity;  this.boatCost = boatCost;  }  public int getBoatCapacity() {  return boatCapacity;  }  public void setBoatCapacity(int boatCapacity) {  this.boatCapacity = boatCapacity;  }  public double getBoatCost() {  return boatCost;  }  public void setBoatCost(double boatCost) {  this.boatCost = boatCost;  }  public abstract int boatlife();  @Override  public String toString() {  return "boatCapacity=" + boatCapacity + ", boatCost=" + boatCost;  }  } |

|  |
| --- |
| /\* |

|  |
| --- |
| /\*  \* 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 question01;  /\*\*  \*  \* @author prasanna arla  \*/  public class HomeBoats extends Boats {  private double velocity;  private double distanceTraveled;  public HomeBoats(double velocity, double distanceTraveled, int boatCapacity, double boatCost) {  super(boatCapacity, boatCost);  this.velocity = velocity;  this.distanceTraveled = distanceTraveled;  }  public double getVelocity() {  return velocity;  }  public void setVelocity(double velocity) {  this.velocity = velocity;  }  public double getDistanceTraveled() {  return distanceTraveled;  }  public void setDistanceTraveled(double distanceTraveled) {  this.distanceTraveled = distanceTraveled;  }  public int boatlife() {  int year = 0;  if (distanceTraveled >= 100) {  year = 10;  } else if (distanceTraveled < 100) {  year = 20;  }  return year;  }  @Override  public String toString() {  return super.toString() + "\nHomeBoats{" + "velocity=" + velocity + ", distanceTraveled=" + distanceTraveled + ", boatlifetime=" + boatlife() + '}';  }  } |

|  |
| --- |
| /\*  \* 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 question01;  /\*\*  \*  \* @author prasanna arla  \*/  public class DeckBoats extends Boats {  private double speed;  private String type;  public DeckBoats(double speed, String type, int boatCapacity, double boatCost) {  super(boatCapacity, boatCost);  this.speed = speed;  this.type = type;  }  public double getSpeed() {  return speed;  }  public void setSpeed(double speed) {  this.speed = speed;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public int boatlife() {  int year = 0;  if (type.equalsIgnoreCase("travel")) {  year = 20;  } else if (type.equalsIgnoreCase("delivery")) {  year = 10;  }  return year;  }  @Override  public String toString() {  return super.toString() + "\nDeckBoats{" + "speed=" + speed + ", type=" + type + ", boatlifetime=" + boatlife() + '}';  }  } |

|  |
| --- |
| /\*  \* 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 question01;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class BoatsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.println("prasanna arla");  System.out.println("Enter boat capacity :");  int capacity = scan.nextInt();  System.out.print("enter boat cost :$ ");  double cost = scan.nextDouble();  System.out.print("Enter boat speed: m/s ");  double speed = scan.nextDouble();  System.out.print("Enter boat type: ");  String type = scan.next();  System.out.print("Enter boat velocity: m/s ");  double velocity = scan.nextDouble();  System.out.print("enter boat distance travelled: miles ");  double distance = scan.nextDouble();  DeckBoats boat1 = new DeckBoats(speed, type, capacity, cost);  HomeBoats boat2 = new HomeBoats(velocity, distance, capacity, cost);  System.out.println(boat1.toString());  System.out.println(boat2.toString());  }  } |

**OUTPUT:**



**Example for interface:**

**Description:** Tables is a class which has two instance variables . it has a over loaded constructor and setter and getter method to modify the values of variables. As Tables implements two interfaces DinnigTable, CoffeeTable the two methods are over ridded in the tables class.

Both methods give the Area of the tables and the values are shown in the tostring in the main driver class we create a object for the Tables class and print toString.

**Program:**

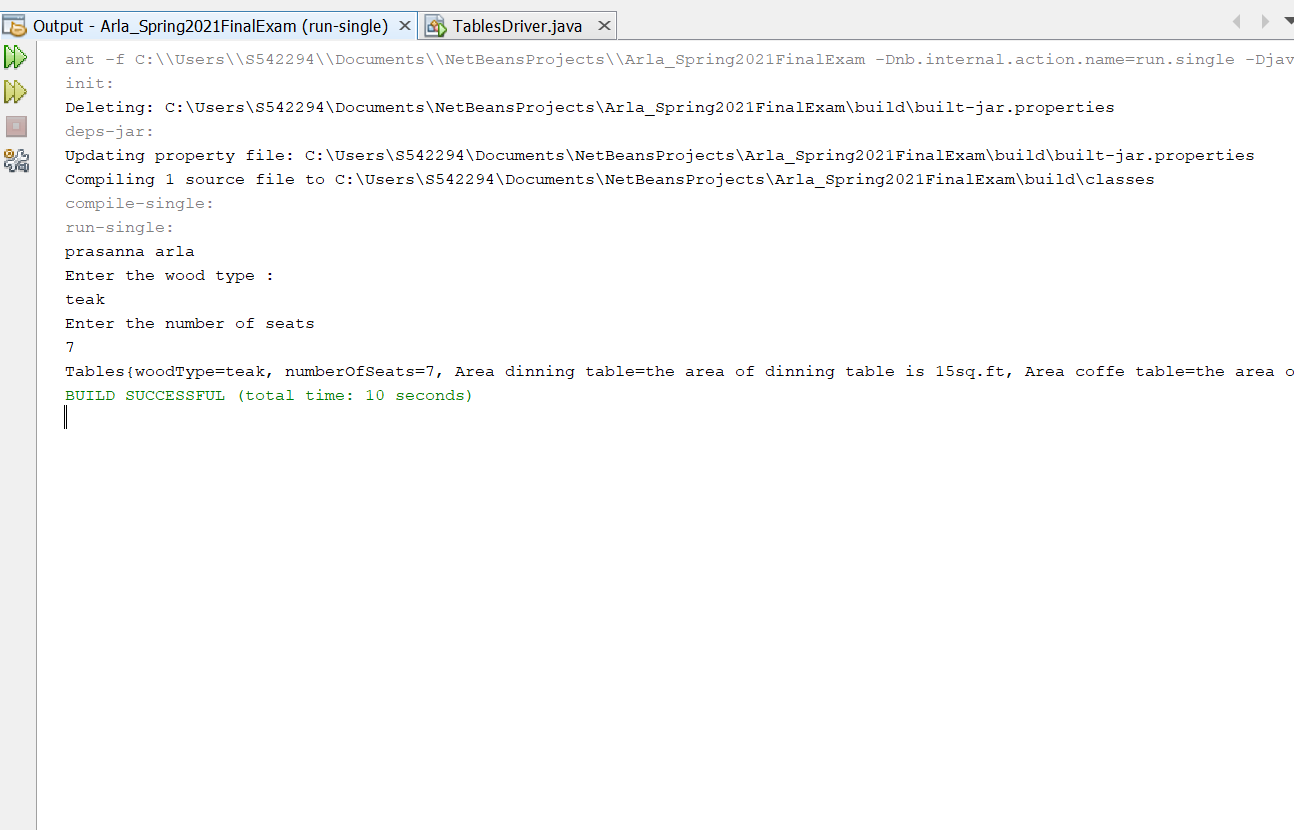
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question01example02;  /\*\*  \*  \* @author prasanna arla  \*/  public class Tables implements DinnigTable, CoffeeTable {  public String woodType;  public int numberOfSeats;  public Tables(String woodType, int numberOfSeats) {  this.woodType = woodType;  this.numberOfSeats = numberOfSeats;  }  public String getWoodType() {  return woodType;  }  public void setWoodType(String woodType) {  this.woodType = woodType;  }  public int getNumberOfSeats() {  return numberOfSeats;  }  public void setNumberOfSeats(int numberOfSeats) {  this.numberOfSeats = numberOfSeats;  }  public String CoffeeLenght() {  String len = "the area of coffee table is 5sq.ft";  return len;  }  public String DinnigLenght() {  String len = "the area of dinning table is 15sq.ft";  return len;  }  @Override  public String toString() {  return "Tables{" + "woodType=" + woodType + ", numberOfSeats=" + numberOfSeats + ", Area dinning table=" + DinnigLenght() + ", Area coffe table=" + CoffeeLenght() + '}';  }  } |

|  |
| --- |
| /\*  \* 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 question01example02;  /\*\*  \*  \* @author prasanna arla  \*/  public interface DinnigTable {  public String DinnigLenght();  } |

|  |
| --- |
| /\*  \* 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 question01example02;  /\*\*  \*  \* @author prasanna arla  \*/  public interface CoffeeTable {  public String CoffeeLenght();  } |

|  |
| --- |
| /\*  \* 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 question01example02;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class TablesDriver {  /\*\*  \* @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("prasanna arla");  System.out.println("Enter the wood type :");  String type = scan.next();  System.out.println("Enter the number of seats");  int seat = scan.nextInt();  Tables table = new Tables(type, seat);  System.out.println(table.toString());  }  } |

**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:**

**Description:**

**GeometricObject class:**

I have taken a class named GeometricObject and given an attribute lengthOfSide of type integer. Then given the constructor for the attribute taken. Added a new method named getLengthOfSide which returns the lengthOfSide.

**Square:** This class extends to GeometricObject implements Colorable. In this I have given a constructor for length of integer type.Then overridden the howToColor method and printed the Color all four sides.

**Colorable:** This class is an interface and given a method howToColor.

**Driver Class:** In this driver class I have an array list for the square and given the five objects named geo1, geo2, geo3, geo4 ,geo5 and added them into the array. By using the for loop and considering the length I have calculated the 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;  /\*\*  \*  \* @author prasanna arla  \*/  public class GeometricObject {  private int lenghtOfSide;  public GeometricObject(int a) {  this.lenghtOfSide = a;  }  public int getLengthOfSide() {  return lenghtOfSide;  }  } |

|  |
| --- |
| /\*  \* 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 prasanna arla  \*/  public class Square extends GeometricObject implements Colorable {  public Square(int length) {  super(length);  }  /\*\*  \*  \*/  @Override  public void howToColor() {  System.out.println("Color all four sides");  }  } |

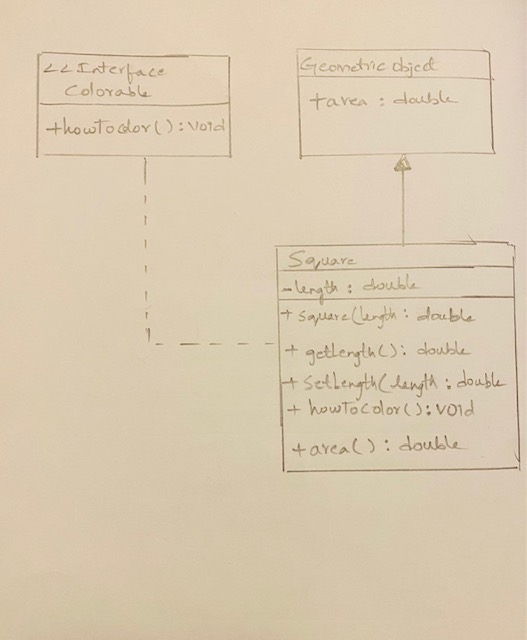
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question02;  /\*\*  \*  \* @author prasanna arla  \*/  public interface Colorable {  public 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 question02;  import java.util.ArrayList;  import java.util.List;  /\*\*  \*  \* @author prasanna arla  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  ArrayList<Square> array = new ArrayList<Square>();  Square u = new Square(3);  Square v = new Square(5);  Square z = new Square(6);  Square y = new Square(9);  Square g = new Square(13);  array.add(u);  array.add(v);  array.add(z);  array.add(y);  array.add(g);  for (int i = 0; i < array.size(); i++) {  int length = array.get(i).getLengthOfSide();  int areaA = length \* length;  System.out.println("prasanna arla");  System.out.println("Area is: " + areaA);  u.howToColor();  }  }  } |

**OUTPUT:**



**UML:**



3.(10-Points) What is casting? What are different types of casting? Explain and demonstrate with examples.

**Answer:**

**Casting:** Assigning a value of one primitive data type to another type is called casting or Casting is a method or process that converts a data type into another data type in both ways manually and automatically. In Java, there are two types of casting:

1. **Widening Casting**
2. **Narrowing 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.

**Narrowing Casting Example:**

**Description:**

**Shapes:** In the shapes class I have taken the variables named numberOfSides of data type integer and length of data type long , then given the constructor for them. After this written the getter , setter and toString methods.

**Polyhedron:** This class extends shapes class . I have taken the variables named edge of data type double and typeOfPolyhedron of datatype string. Then written the constructor for those attributes and given the super class constructors as it extends the superclass . Then I have given the getter , setter method for them and added a new method gerArea to calculate the area , and given the toString method to return the area , edge, and typeOfPolyhedron.

**Driver class:**

In this I have assigned the integer for numberOfSides as 12. And given the datatypes as long and double for length and edge. Then taken the object named cub , in this given the edge, typeOfPolyhedron, numberOfSides and length. Then printed the cub object.

**Program:**

|  |
| --- |
| /\*  \* 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 question03NarrowingCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class ShapesClass {  private int numberOfSides;  private long length;  public ShapesClass(int numberOfSides, long length) {  this.numberOfSides = numberOfSides;  this.length = length;  }  public int getNumberOfSides() {  return numberOfSides;  }  public void setNumberOfSides(int numberOfSides) {  this.numberOfSides = numberOfSides;  }  public long getLength() {  return length;  }  public void setLength(long length) {  this.length = length;  }  @Override  public String toString() {  return "numberOfSides: " + numberOfSides + "\nlength: " + 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 question03NarrowingCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class Polyhedron extends ShapesClass {  private double edge;  private String typeOfPolyhedron;  public Polyhedron(double edge, String typeOfPolyhedron, int numberOfSides, long length) {  super(numberOfSides, length);  this.edge = edge;  this.typeOfPolyhedron = typeOfPolyhedron;  }  public double getEdge() {  return edge;  }  public void setEdge(double edge) {  this.edge = edge;  }  public String getTypeOfPolyhedron() {  return typeOfPolyhedron;  }  public void setTypeOfPolydron(String typeOfPolyhedron) {  this.typeOfPolyhedron = typeOfPolyhedron;  }  public double area() {  double area = 2 \* Math.sqrt(3) \* Math.pow(edge, 2);  return area;  }  @Override  public String toString() {  return super.toString() + "\nedge: " + edge + "\nArea of the Octahedron: " + 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 question03NarrowingCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class ShapesDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  double edge = 12.7;  long length = (long) edge;  int numberOfSides = (int) length;  Polyhedron cub = new Polyhedron(edge, "Octahedron", numberOfSides, length);  System.out.println("prasanna arla");  System.out.println(cub.toString());  }  } |

**OUTPUT:**



**Widening casting Example:**

**Description:**

**Shapes :** In this I have taken the variables named numberOfSides and length and then written the constructor for them. After this I have given the getter , setter methods and toString .

**Cuboid:** This cuboid class extends to shapes , I have taken variables named width of datatype long and height of datatype float. Then given the constructor for them. Written the getter , setter methods for them and writer one more method for area named getArea of type double. Then written the toString method which returns the width, height, area.

**Driver:** In this I have given the value 17 for the nuberOfSides and assigned height and width as numberOfSides of datatypes float and long. Then width of data type double as numberOfSides. Then written an object named cub , given values for that and printed the toString.

**Program:**

|  |
| --- |
| /\*  \* 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 question03WideningCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class Shapes {  private int numberOfSides;  private double length;  public Shapes(int numberOfSides, double length) {  this.numberOfSides = numberOfSides;  this.length = length;  }  public int getNumberOfSides() {  return numberOfSides;  }  public void setNumberOfSides(int numberOfSides) {  this.numberOfSides = numberOfSides;  }  public double getLength() {  return length;  }  public void setLength(double length) {  this.length = length;  }  @Override  public String toString() {  return "numberOfSides: " + numberOfSides + "\nlength: " + 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 question03WideningCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class Cuboid extends Shapes {  private long width;  private float height;  public Cuboid(long width, float height, int numberOfSides, double length) {  super(numberOfSides, length);  this.width = width;  this.height = height;  }  public long getWidth() {  return width;  }  public void setWidth(long width) {  this.width = width;  }  public float getHeight() {  return height;  }  public void setHeight(float height) {  this.height = height;  }  public double area() {  double area = 2 \* height \* (super.getLength() + width);  return area;  }  @Override  public String toString() {  return super.toString() + "\nwidth: " + width + "\nheight: " + height  + "\nArea: " + 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 question03WideningCasting;  /\*\*  \*  \* @author prasanna arla  \*/  public class ShapesDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int numberOfSides = 17;  float height = numberOfSides;  long width = numberOfSides;  double length = width;  Shapes cub = new Cuboid(width, height, numberOfSides, length);  System.out.println("prasanna arla");  System.out.println(cub.toString());  }  } |

**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.

**Answer:**

**Description:**

**Circle:** In this class I have taken an variable named radius, given the constructor for that.Then written the getter and setter method for that . Added a new method named getArea of type double, then written the toString method for radius.

**ComparableCircle:** This class extends to Circle implements Comparable<ComparableCircle>. Then written the constructor for the radius. I have given the toString for superclass area and for radius of superclass.

**Driver :** In this I have 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.

**Program:**

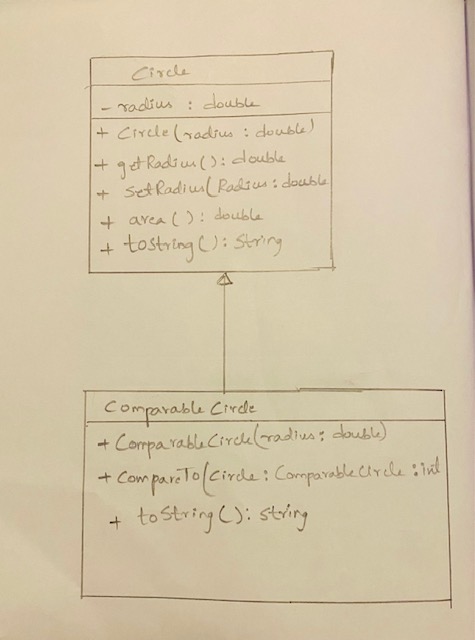
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question05;  /\*\*  \*  \* @author prasanna arla  \*/  public class Circle {  private double 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 prasanna arla  \*/  public class ComparableCircle extends Circle implements Comparable<ComparableCircle> {  public ComparableCircle(double radius) {  super(radius);  }  @Override  public int compareTo(ComparableCircle circle) {  return Double.compare(super.area(), circle.area());  }  @Override  public String toString() {  return "radius:- " + super.getRadius();  }  } |

|  |
| --- |
| /\*  \* 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;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ComparableCircle circle1 = new ComparableCircle(15.8);  ComparableCircle circle2 = new ComparableCircle(16.4);  System.out.println("prasanna arla");  System.out.println("Larger instance: " + findLargest(circle1, circle2));  }  private static String findLargest(ComparableCircle c1, ComparableCircle c2) {  //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  String p = " ";  if (c1.compareTo(c2) > 0) {  p = "The first circle area is large.";  } else if (c1.compareTo(c2) < 0) {  p = "The second circle area is large.";  } else {  p = "Both the areas are equal";  }  return p;  }  } |

**OUTPUT:** 

**UML:**



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

**Answer:**

**Exception:** An event which occurs during the execution of the program is called an exception. For examples arithmetic exception, Null pointer exception, Divide by zero exception and many. An exception is something that is left out or not done on purpose. An exception to a rule does not follow that rule.

**Advantages of Exception:**

* 1. Separating Error Handling Code from "Regular" Code.
  2. Propagating Errors Up the Call Stack.
  3. Grouping Error Types and Error Differentiation.

**Disadvantages of Exception:**

1.They can only trap runtime errors.

2. Exceptions are hard to ignore, unlike error codes.

3. We experience unnecessary overheads.

**Checked Exception:**

The exceptions that are checked at the compile time are called checked exceptions. If a checked exception is thrown in the method then it should handle the exception using try catch block or it should declare the exception using throws keyword, otherwise compilation error occurs. Examples for checked exceptions are IOexception, SQLexception, ParseException. But most of the exceptions are bad because they are forced to deal with the API that overuse them.

**Unchecked Exception:**

The exceptions that are occurred at the execution time are called unchecked exceptions. These are also called as runtime exceptions. They include bugs such as logic errors or improper use of an API. These are ignored at the time of compilation. All unchecked exceptions are direct subclasses of runtime exception class.

**Examples of unchecked exception:**

* NullPointerException
* ArithmeticException
* NumberFormatException
* IllegalArgumentException
* ArrayIndexOutOfBondsException

**Advantages of Unchecked exception:**

* Separating error handling code from “Regular “ code.
* Grouping and Differentiating error types.
* Propagating errors up the call stack.

**Disadvantages of Unchecked exception:**

* When unchecked exceptions are not declared by methods they may throw it becomes more difficult to handle them .
* If exception is raised and if it is not handled the program will stopped.

**Program for checked exception:**

**Description:**

**Store:**In class train, declared four variables number of sellerId and customerName, noOfItems and cost. Declared constructor, setter and getter methods. Declaring a method named MoneyReceivedBySeller and written the toString method.

**General:** This class extends to the Store and I have taken two variables named costPrice and sellingPrice. Written the constructor, setter , getter methods for them and taken a method named MoneyReceivedBySeller which returns the totalRecieved.

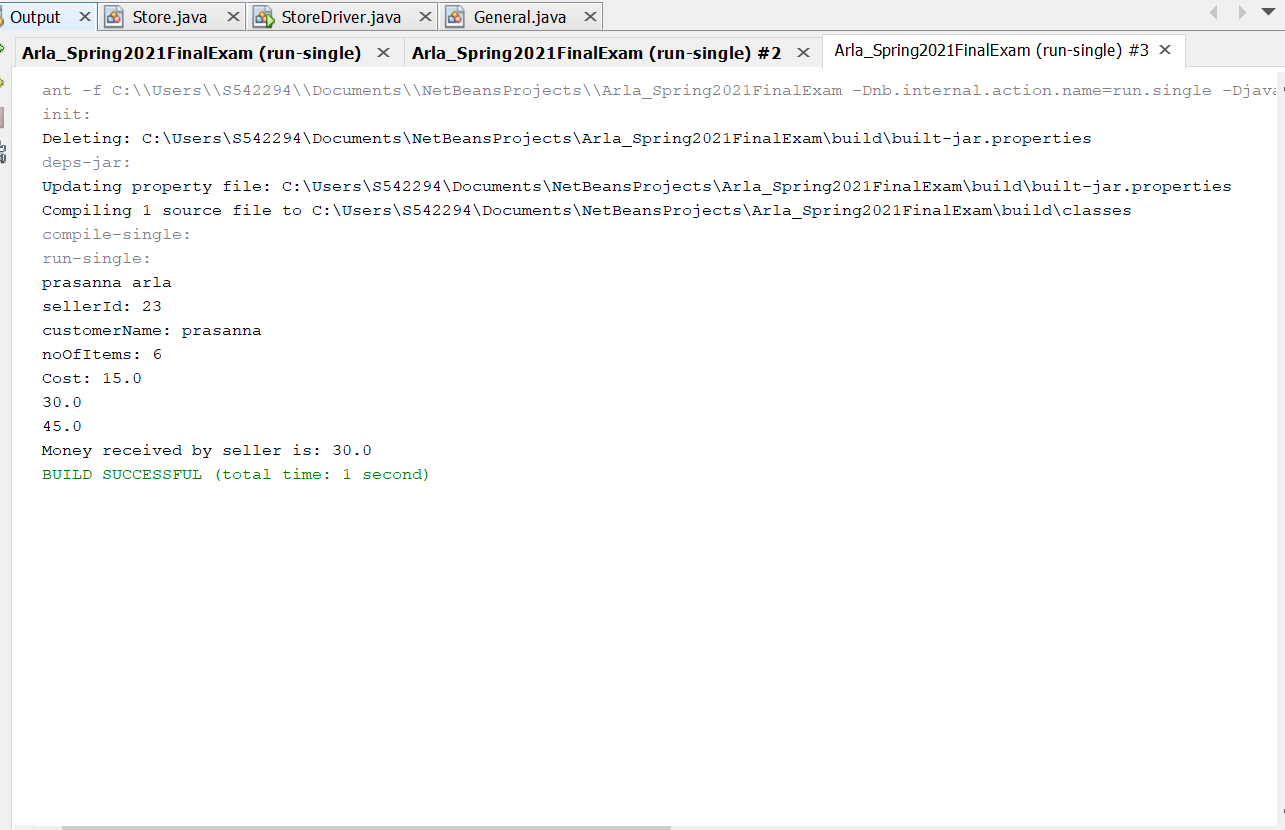
**Driver**: In this I have taken a scanner object and given input file. Then used the while loop and given an object named store and it gives the selling price and cost price and money received by seller. If the file is not found then the file not found exception is thrown.

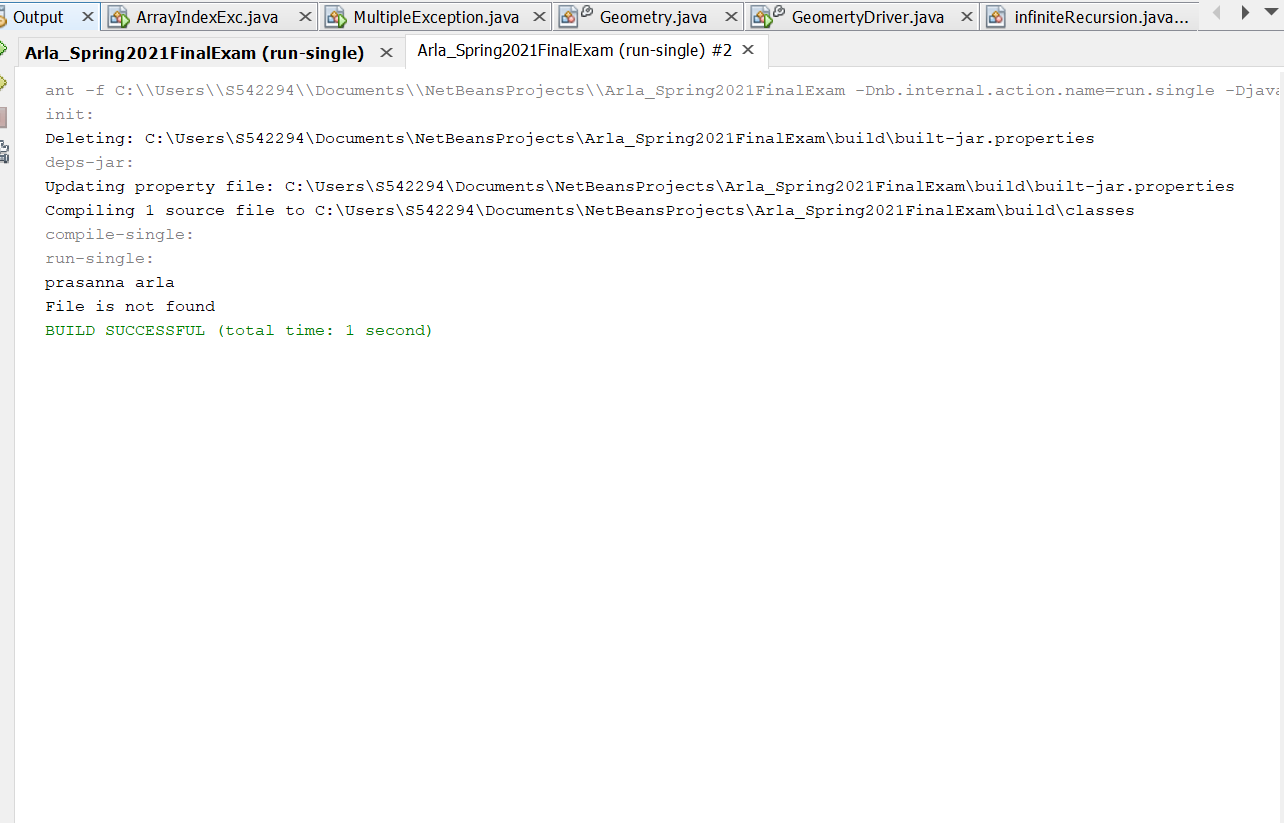
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question06checkedexception;  /\*\*  \*  \* @author prasanna arla  \*/  public class Store {  private int sellerId;  private String customerName;  private int noOfItems;  private double Cost;  public Store(int sellerId, String customerName, int noOfItems, double Cost) {  this.sellerId = sellerId;  this.customerName = customerName;  this.noOfItems = noOfItems;  this.Cost = Cost;  }  public int getSellerId() {  return sellerId;  }  public void setSellerId(int sellerId) {  this.sellerId = sellerId;  }  public String getCustomerName() {  return customerName;  }  public void setCustomerName(String customerName) {  this.customerName = customerName;  }  public int getNoOfItems() {  return noOfItems;  }  public void setNoOfItems(int noOfItems) {  this.noOfItems = noOfItems;  }  public double getCost() {  return Cost;  }  public void setCost(double Cost) {  this.Cost = Cost;  }  public double MoneyReceivedBySeller() {  double moneyReceived = noOfItems \* Cost;  return moneyReceived;  }  @Override  public String toString() {  return "sellerId: " + sellerId + "\ncustomerName: " + customerName + "\nnoOfItems: " + noOfItems + "\nCost: " + 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 question06checkedexception;  /\*\*  \*  \* @author prasanna arla  \*/  public class General extends Store {  private double costPrice;  private double sellingPrice;  public General(double costPrice, double sellingPrice, int sellerId, String customerName, int noOfItems, double Cost) {  super(sellerId, customerName, noOfItems, Cost);  this.costPrice = costPrice;  this.sellingPrice = sellingPrice;  }  public double getCostPrice() {  return costPrice;  }  public void setCostPrice(double costPrice) {  this.costPrice = costPrice;  }  public double getSellingPrice() {  return sellingPrice;  }  public void setSellingPrice(double sellingPrice) {  this.sellingPrice = sellingPrice;  }  @Override  public double MoneyReceivedBySeller() {  double profit = sellingPrice - costPrice;  double totalReceived = super.getCost() + profit;  return totalReceived;  }  } |

|  |
| --- |
| /\*  \* 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 question06checkedexception;  /\*\*  \*  \* @author prasanna arla  \*/  import java.io.File;  import java.io.FileNotFoundException;  import java.util.Scanner;  public class StoreDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws FileNotFoundException {  // TODO code application logic here  try {  Scanner scan = new Scanner(new File("fileinput.txt"));  while (scan.hasNext()) {  int sellerId = scan.nextInt();  String customerName = scan.next();  int noOfItems = scan.nextInt();  double Cost = scan.nextDouble();  double costPrice = scan.nextDouble();  double sellingPrice = scan.nextDouble();  Store store = new Store(sellerId, customerName, noOfItems, Cost);  General gen = new General(costPrice, sellingPrice, sellerId, customerName, noOfItems, Cost);  System.out.println("prasanna arla");  System.out.println(store.toString());  System.out.println(gen.getCostPrice());  System.out.println(gen.getSellingPrice());  System.out.print("Money received by seller is: ");  System.out.println(gen.MoneyReceivedBySeller());  }  } catch (FileNotFoundException e) {  System.out.println("File is not found");  }  }  } |

**Output:**





**Program for unchecked exception:**

**Description:**

**Fruits:**

In this class I have taken three attributes named typeOfFruit , weight, cost.And written the constrictor, getter , setter for them and also the toString method for them which return all the variables.

**FruitDriver:**

In this I have taken an scanner object which takes the system input. By using the print statement we give the weight and the cost of the fruit. Next by using the null exception we gave the fruit type. By using the equals , if the typeOfFruit is equal to mango then it prints fruit name is mango otherwise exception is 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 question06uncheckedexception;  /\*\*  \*  \* @author prasanna arla  \*/  public class Fruits {  private String typeOfFruit;  private long weight;  private double cost;  public Fruits(String typeOfFruit, long weight, double cost) {  this.typeOfFruit = typeOfFruit;  this.weight = weight;  this.cost = cost;  }  public String getTypeOfFruit() {  return typeOfFruit;  }  public void setTypeOfFruit(String typeOfFruit) {  this.typeOfFruit = typeOfFruit;  }  public long getWeight() {  return weight;  }  public void setWeight(long weight) {  this.weight = weight;  }  public double getCost() {  return cost;  }  public void setCost(double cost) {  this.cost = cost;  }  @Override  public String toString() {  return "typeOfFruit: " + typeOfFruit + "\nweight: " + weight + "\ncost: " + 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 question06uncheckedexception;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class FruitDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws NullPointerException {  // TODO code application logic here  Scanner scan = new Scanner(System.in);  System.out.println("prasanna arla");  System.out.print("Enter the weight in lbs: ");  long weight = scan.nextLong();  System.out.print("Enter the cost in $: ");  double cost = scan.nextDouble();  String typeOfFruit = null;  System.out.println("Enter the fruit name:");  typeOfFruit = scan.next();  Fruits fr = new Fruits(typeOfFruit, weight, cost);  System.out.println(fr.toString());  try {  if (typeOfFruit.equals("mango")) {  System.out.println("Fruit name is mango");  }  } catch (NullPointerException np) {  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.

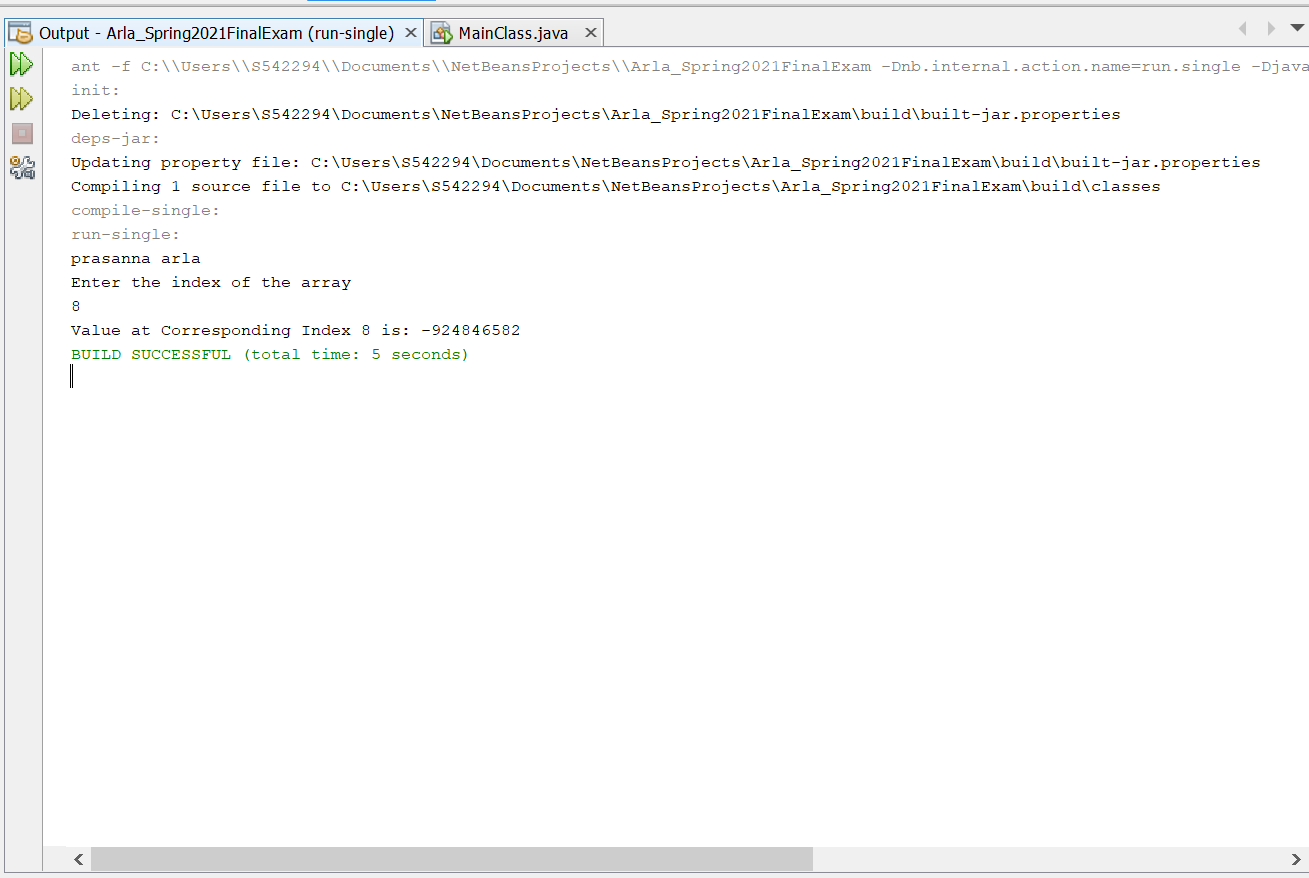
**Answer:**

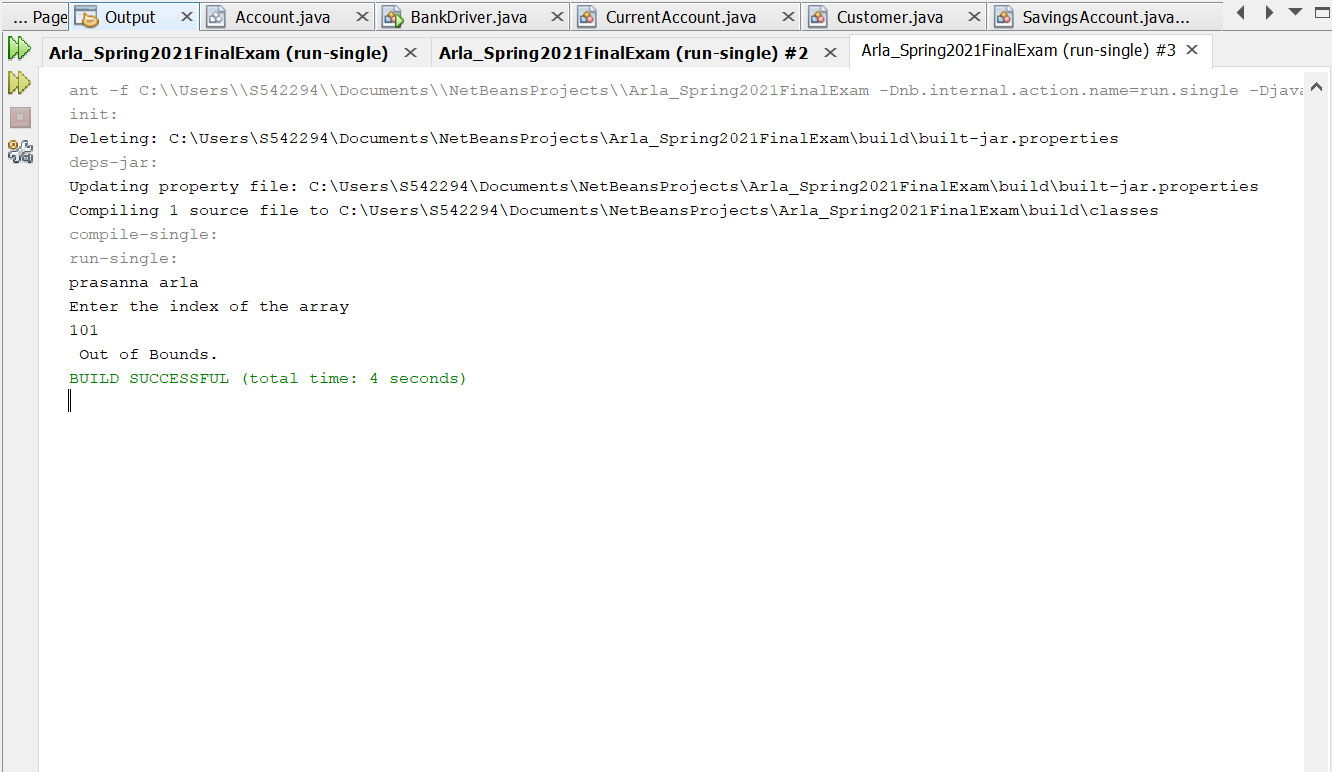
**Description:** In this I have taken a object named rand and taken an array of length 100 .Then using the scanner and scanA object all the input values are taken. Using the print statement we enter the index values as of our wish if the index is less than 100 it prints the value of the index number otherwise it prints Out of Bounds.

**Program:**

|  |
| --- |
| /\*  \* 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.\*;  /\*\*  \*  \* @author prasanna arla  \*/  public class MainClass {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Random rand = new Random();  int[] arr = new int[100];  for (int i = 0; i < arr.length; i++) {  arr[i] = rand.nextInt();  }  Scanner scanA = new Scanner(System.in);  System.out.println("prasanna arla");  System.out.println("Enter the index of the array ");  int input = scanA.nextInt();  if (input >= 0 && input < 100) {  System.out.println("Value at Corresponding Index " + input + " is: " + arr[input]);  } else {  System.out.println(" Out of Bounds.");  }  scanA.close();  }  } |

**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:**

The purpose of declaring exceptions is to inform the java runtime system what can go wrong. In java the main concepts is code reusability. We declare an exception using the keyword throw in method declaration. When we create an exception, we can have checked exception by extending the exception class. Similarly, if we want to have an unchecked exception, we can extend Runtime Exception class. We always throw a single exception in a single statement in java when catching the exception after try block un try catch.

We can declare multiple exceptions by using the commas (,). Every method must declare the types of checked exceptions it might throw. If a method implementation contains a throw statement, it is possible that an exception will be thrown from within the methods. A method can catch an exception by providing an exception handler for that type of exception. If a method chooses not to catch an exception, then the method must declare that it can throw it. In the method header if the method throws an exception.

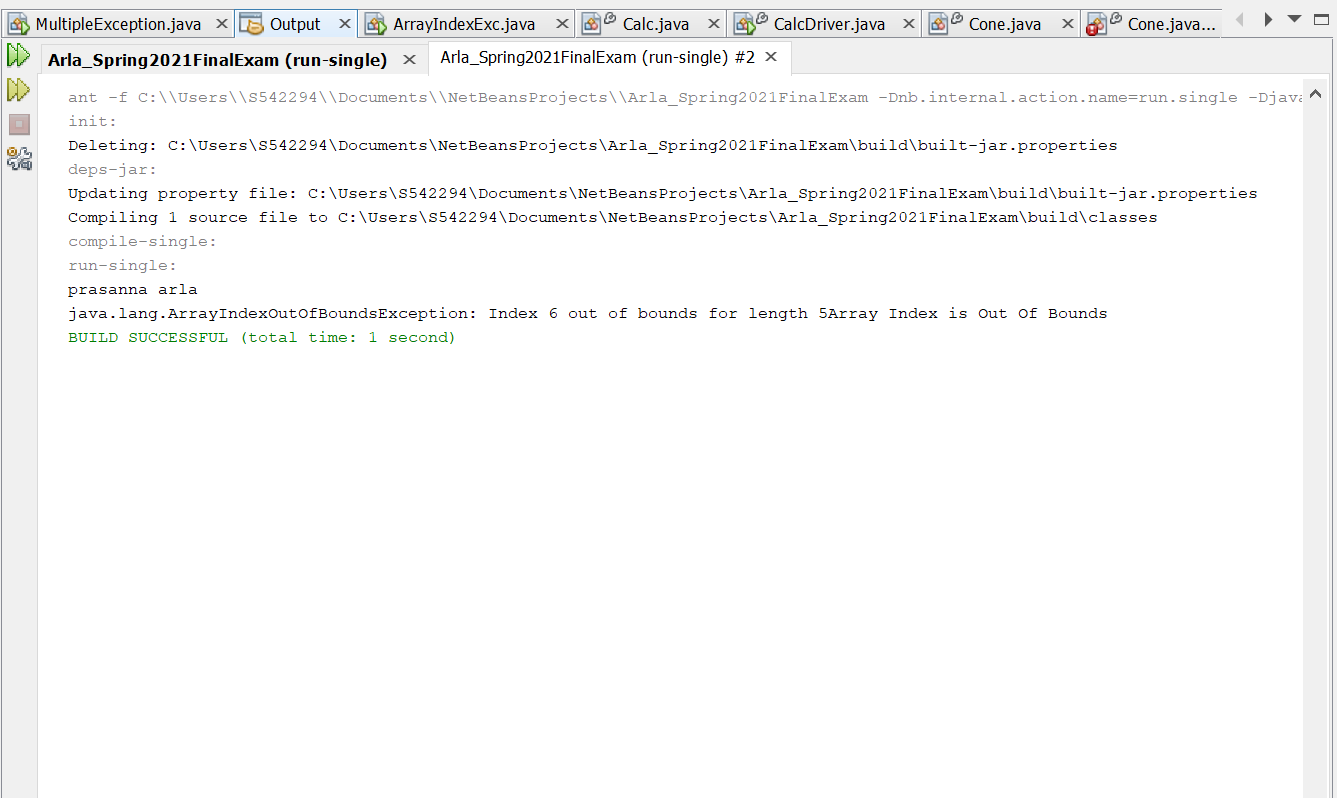
Within the method for custom exceptions or user thrown exceptions you’d use the throw keyword. Yes, we can declare the multiple exceptions in a method header, they are separated by commas after throws.

**Example 01:**

**Program Description:** 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 question08;  /\*\*  \*  \* @author prasanna arla  \*/  public class ArrayIndexExc {  /\*\*  \* @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(e + "Array Index is Out Of Bounds");  }  }  } |

**Output:**

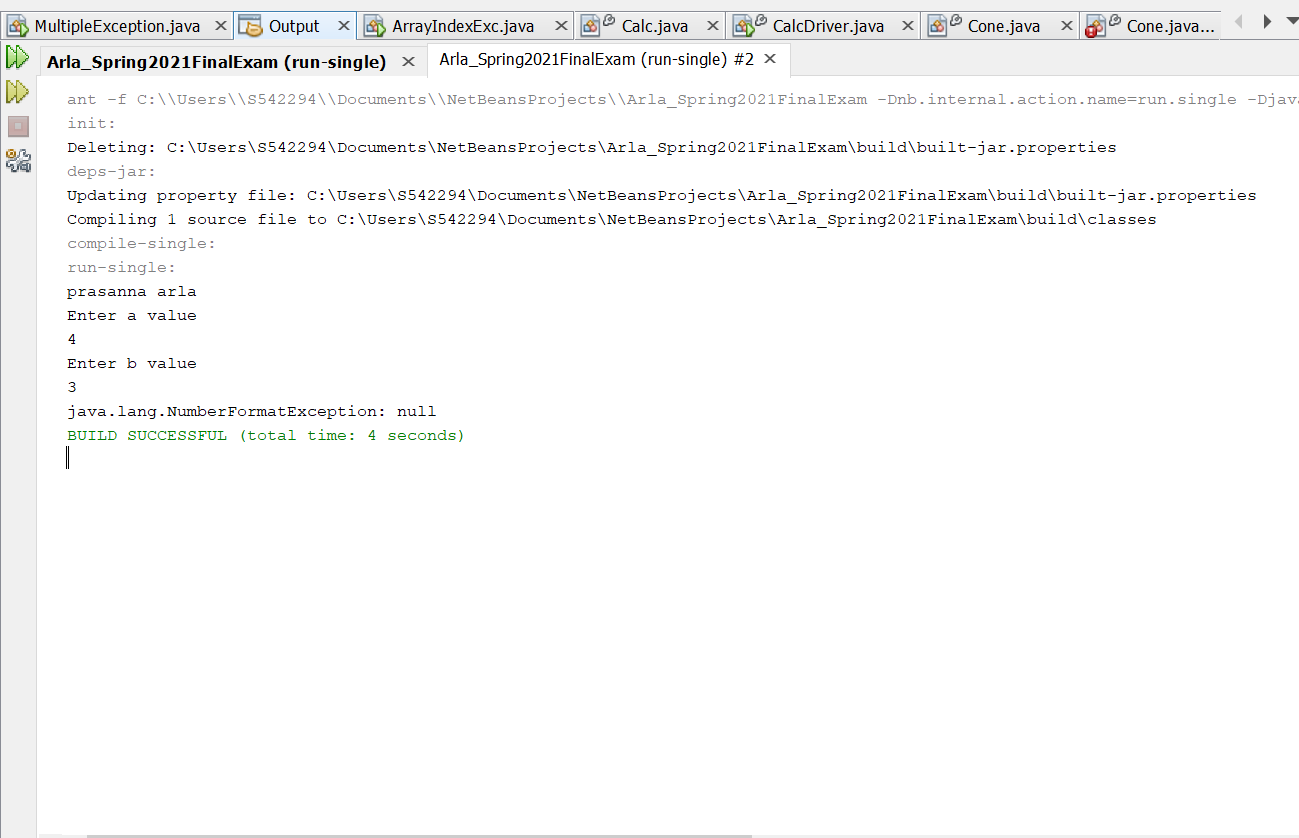


**Example 02:**

**Description:** In this I have taken scanner object to take the system input. And by using the print statement I have entered the value a and value b by using the sc.nxtInt(). Then used the method for and implemented the catch for exception and prints the exception statement. However during the method declaration you’d still need to mention all the exceptions that can be thrown from within that method . If integerB == 0 then Arithmetic Exception" is thrown. If integerA = Integer.parseInt(null) then NumberFormat Exception is printed.

|  |
| --- |
| package question08;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class MultipleException {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  try {  Scanner sc = new Scanner(System.in);  System.out.println("prasanna arla");  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 keyword Throw is used to throw an exception. When throwing the exception we always throw a single exceptions in a single java statement. In java the exceptions are inbuilt or custom. Whether it may be anything one throw the exception to let the user know that the current flow of the program cannot proceed further. When an exception is thrown the flow of program execution transfers from the try block to catch block. We use throw keyword within a method The syntax is: Here the throwable object is an instance of class Throwable or subclass of the Throwable class.

Throw throwableObject;

Throws keyword is used in the Method Header whenever a single or multiple exception is thrown. It is used to declare an exception. The programmer gets the information that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained. Here these exceptions could be user thrown exceptions or because of the codelike like in case of creating a file, by default the method using the file needs to throw the FileNotFoundException. If there occurs any unchecked exception such as NullPointerException, it is programmers’ fault that he is not performing checkup before the code being used.

return\_ type method\_ name() throws exception\_class\_name{

//block code

If one or more exception is thrown, we can choose if you we can use try block for each statement that could thrown an exception or use one try block for multiple statements that might throw multiple exceptions.

**Example 01:**

**Description:**

**Class:**

In this class I have taken two variables named value 1 and value2 and created the constructor, getter and setter for them. Then added a method called sum which returns the sum of two values. Then by using the exception ,if sum is less than zero them exception is thrown or else sum is printed . Next the toString the value1 and value2 is printed.

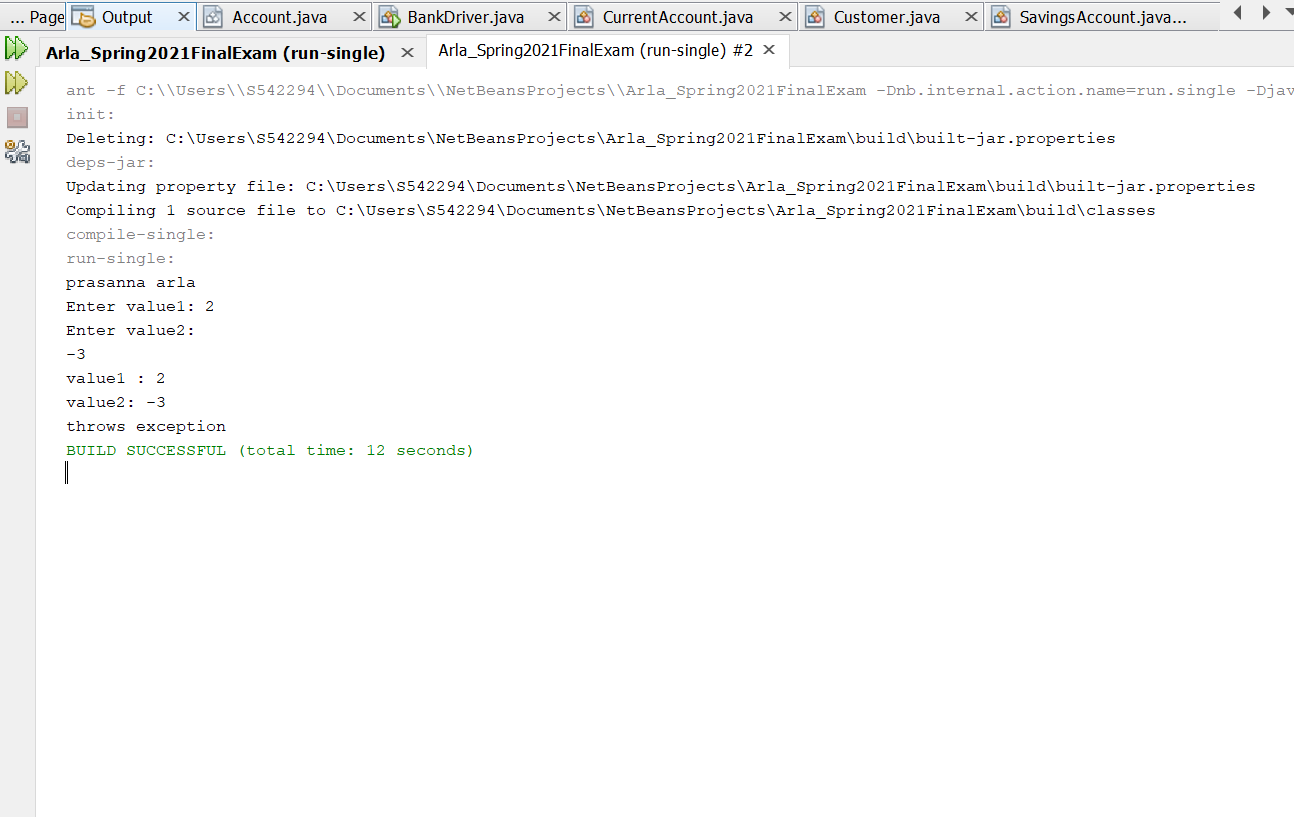
**Driver class:** In this I have taken an scanner object , then through print statement I have entered the value1 and value2 numbers. Then using the try catch block printed the toString method and exception method. After this exceptiontoString is given which returns value1 and value2.

|  |
| --- |
| /\*  \* 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 question09example01;  /\*\*  \*  \* @author prasanna arla  \*/  public class Class {  private int value1;  private int value2;  public Class(int value1, int value2) {  this.value1 = value1;  this.value2 = value2;  }  public int getValue1() {  return value1;  }  public void setValue1(int value1) {  this.value1 = value1;  }  public int getValue2() {  return value2;  }  public void setValue2(int value2) {  this.value2 = value2;  }  public int sum() {  return value1 + value2;  }  public String Exception() {  String st = " ";  if ((value1 + value2) < 0) {  throw new ArithmeticException("Throws exception");  } else {  st = ("The sum is: " + sum());  }  return st;  }  @Override  public String toString() {  return "value1 : " + 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 question09example01;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class DriverClass {  /\*\*  \* @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("prasanna arla");  System.out.print("Enter value1: ");  int value1 = scan.nextInt();  System.out.println("Enter value2: ");  int value2 = scan.nextInt();  try {  Class c = new Class(value1, value2);  System.out.println(c.toString());  System.out.println(c.Exception());  } catch (Exception e) {  System.out.println("throws exception");  }  }  } |

**Output:**





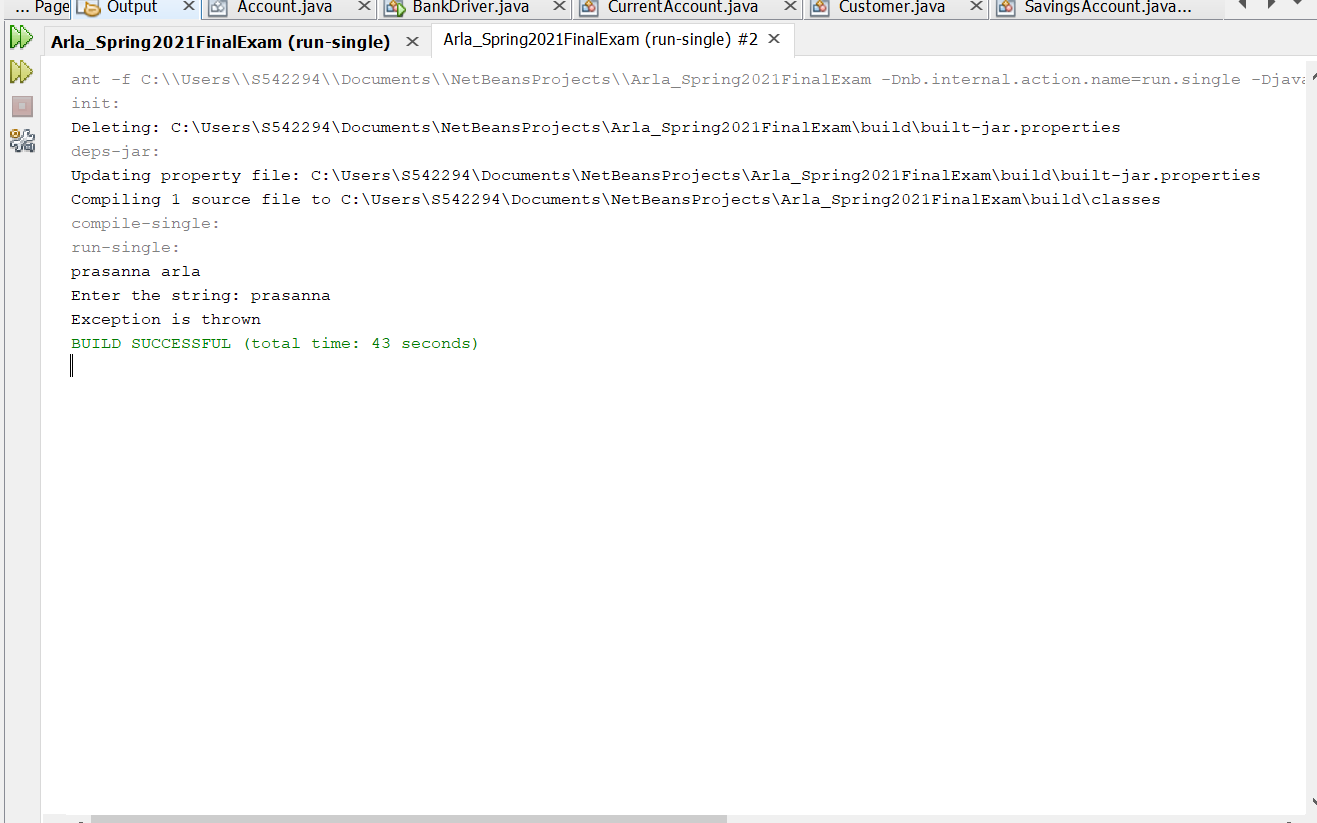
**Example02:**

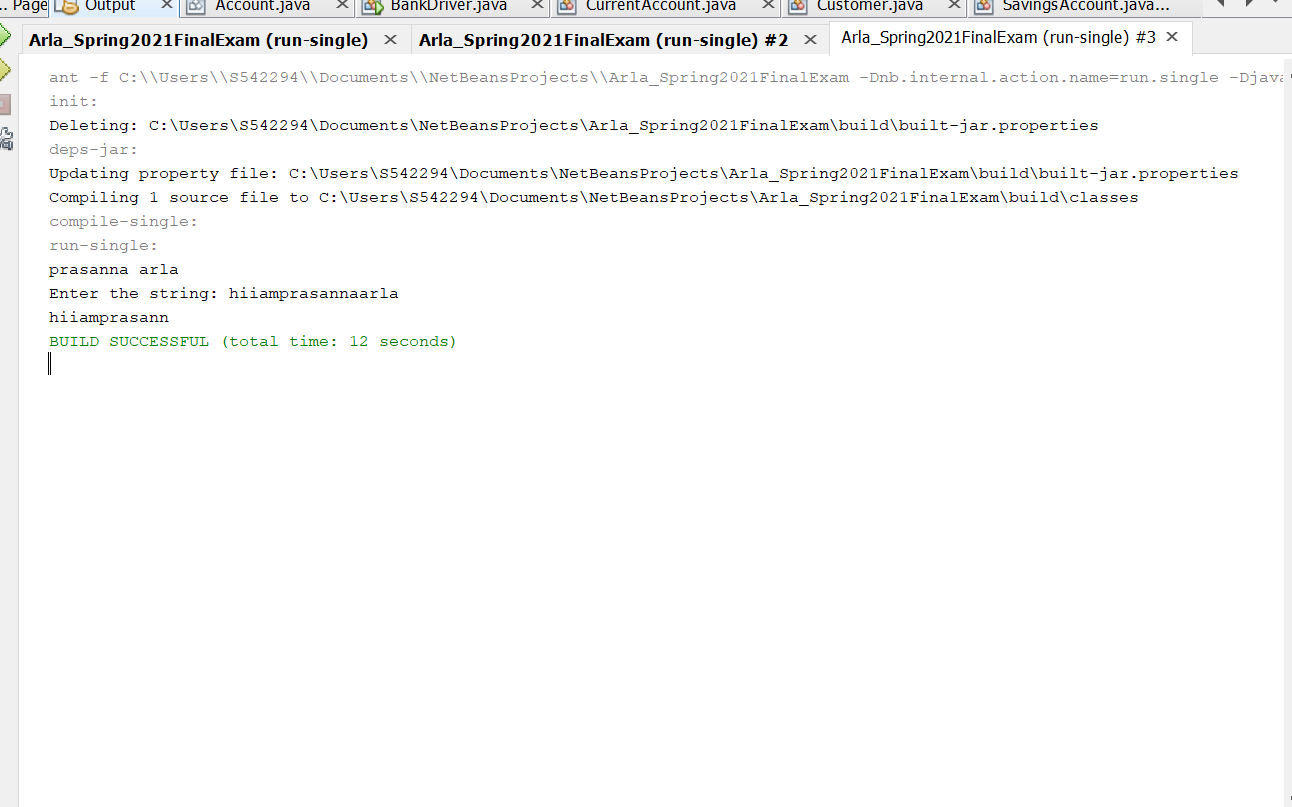
**Description: This throws the** IndexOutOfBoundsException , in this driver class I have taken a scanner object which takes the system input . After that using the try catch block if the substring is between the index(0,12) then the string is printed or else the exception is thrown.

**Throws:**

|  |
| --- |
| /\*  \* 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 question09examples02;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class ThrowsDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws IndexOutOfBoundsException {  Scanner scan = new Scanner(System.in);  System.out.println("prasanna arla");  System.out.print("Enter the string: ");  try {  String str = scan.next();  System.out.println(str.substring(0, 12));  } catch (Exception e) {  System.out.println("Exception is thrown");  }  }  } |

**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:**

**Recursive method:**

The technique of making a function call itself is called recursive method. the technique of making a function call itself. Recursive functions use something called “the call stack”. the technique of making a function call itself. Recursion is made for solving problems that can be broken down into smaller, repetitive problems. It is especially good for working on things that have many possible branches and are too complex for an iterative approach. It helps to break complicated problems so that we have a simple solution.

**Advantages:**

* 1. Recursion can reduce time complexity
  2. Recursion is better at tree traversal.
  3. Adds more clarity and reduces the time for debugging the code.

**Infinite recursion:** Infinite recursion is the non-terminating execution of a block of code. Infinite recursion is usually caused by a bug in code. In this type of recursion there is no base condition. An infinite loop occurs with iteration if the loop-continuation test never becomes false, infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on the base case. It keeps on calling itself as there is no condition which is getting satisfied. To explain it in an simple way, it occurs if the recursion step does not reduce the problem.

**Example 01 Description:**

**InfiniteRecursion:** In this class I have taken two attributes named value1 and value2. And inserted the constructor for the attributes and inserted the getter and setter methods for them. After this I have implemented a new method named series The method is calculating the power of value1 to value2 and we are adding the result to the output of series function by calling recursively since we are calling and this method recursively this loop will never break. Then added the toString method which returns the value1 and value2.

**DriverClass:** In driver class I have take a scanner class the which scans the two values. Value1 and value2 from the CMD and initialize the numbers by creating an object to infiniteRecursion class. Then by using the try block taken an new object inr which initializes the value1 and value2. Then through the print statement the inr object toSting is printed. After this the try block again goes to the method named series , this method is having the math function and then series (n). As this runs continuously the loop will never break.

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package question10;  /\*\*  \*  \* @author prasanna arla  \*/  public class infiniteRecursion {  private double value1;  private double value2;  public infiniteRecursion(double value1, double value2) {  this.value1 = value1;  this.value2 = value2;  }  public double getValue1() {  return value1;  }  public void setValue1(double value1) {  this.value1 = value1;  }  public double getValue2() {  return value2;  }  public void setValue2(double value2) {  this.value2 = value2;  }  public double power(double x) {  if (value1 == 0 || value2 == 0) {  return 0.0;  } else {  return power(Math.pow(value1, value2));  }  }  @Override  public String toString() {  return "number1: " + value1 + "\nnumber2: " + 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 question10;  import java.util.Scanner;  /\*\*  \*  \* @author prasanna arla  \*/  public class DriverClass {  /\*\*  \* @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("prasanna arla");  System.out.print("Enter the first value: ");  double value1 = scan.nextDouble();  System.out.print("Enter the second value: ");  double value2 = scan.nextDouble();  try {  infiniteRecursion in = new infiniteRecursion(value1, value2);  System.out.println(in.toString());  System.out.println(in.power(12.5));  } catch (StackOverflowError o) {  System.out.println("infinite recursion occured");  }  }  } |

**Output:**

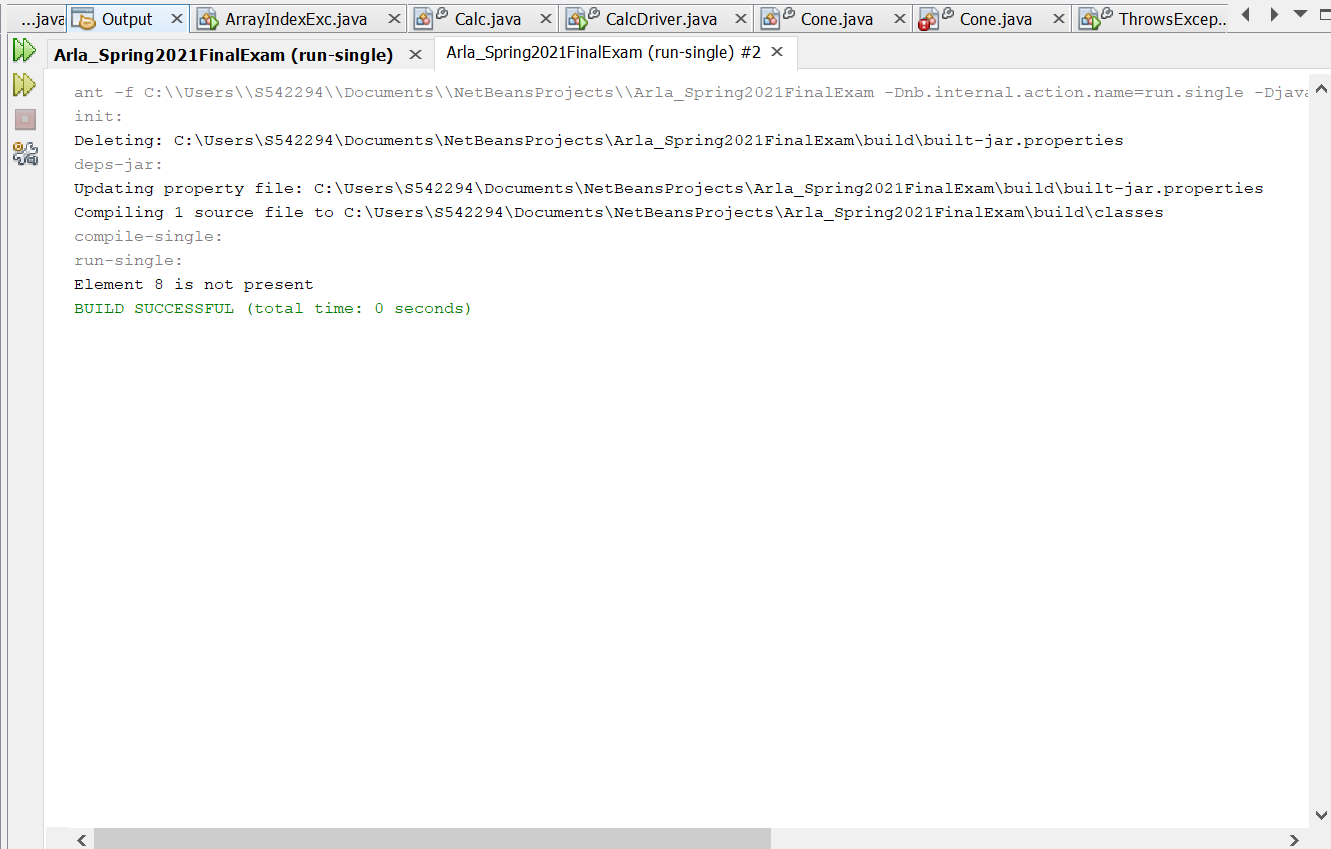


**Example 02:**

**Description:** In this driver class I have taken a variable named x of type integer and then declared an array. Given index as 8 also declared a array list. By using recursive search in array list and loop, if x is not equal to one, then print along with the index else print that the index 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 question10example02;  /\*\*  \*  \* @author prasanna arla  \*/  public class Driver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int i = 8;  int arr[] = new int[]{12, 30, 15, 7, 11};  int x = recursiveSearch(arr, 0, arr.length - 1, i);  if (x != -1) {  System.out.println("prasanna arla");  System.out.println("Element " + i + " is located at index "  + x);  } else {  System.out.println("Element " + i + " is not present");  }  }  private static int recursiveSearch(int[] array, int x, int y, int z) {  if (y < x) {  return -1;  }  if (array[x] == z) {  return x;  }  if (array[y] == z) {  return y;  }  return recursiveSearch(array, x + 1, y - 1, z);  }  } |

**Output:**



11.(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:

**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:**

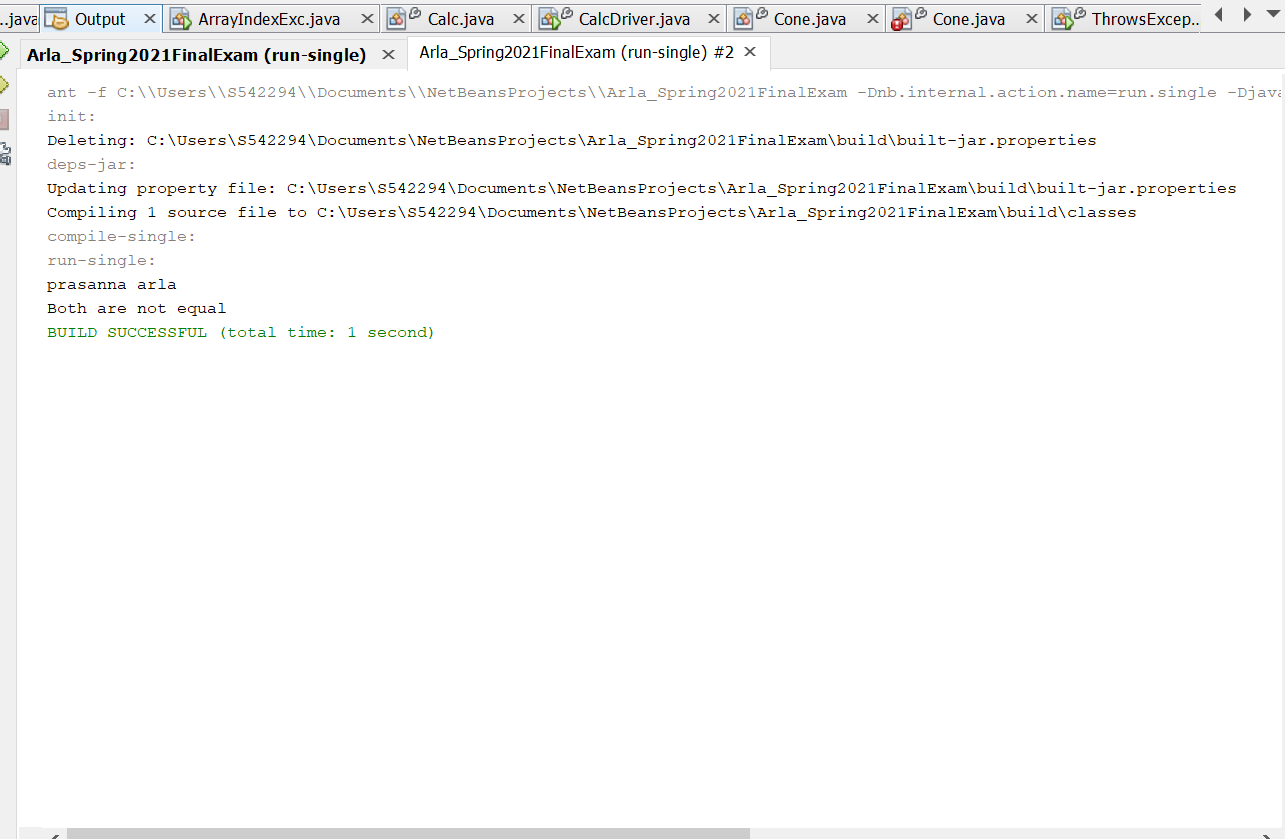
It returns the hashcode value as an Integer. Hashcode value is mostly used in hashing based collections like HashMap, HashSet, HashTable….etc. This method must be overridden in every class which overrides equals() method.

**Example 01:**

|  |
| --- |
| /\*  \* 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.Objects;  /\*\*  \*  \* @author prasanna arla  \*/  public class Geometry {  private String shapeName;  private double side1;  private double side2;  /\*\*  \*  \* @param shapeName name of the shop is taken  \* @param side1 is taken  \* @param side2 is taken  \*/  public Geometry(String shapeName, double side1, double side2) {  this.shapeName = shapeName;  this.side1 = side1;  this.side2 = side2;  }  /\*\*  \*  \* @return gets the shapename  \*/  public String getShapeName() {  return shapeName;  }  /\*\*  \*  \* @param shapeName is set here  \*/  public void setShapeName(String shapeName) {  this.shapeName = shapeName;  }  /\*\*  \* getter method for side1  \*  \* @return will get side1  \*/  public double getSide1() {  return side1;  }  /\*\*  \* '  \* Setter method for side1  \*  \* @param side1 is set  \*/  public void setSide1(double side1) {  this.side1 = side1;  }  /\*\*  \* getter method for side2  \*  \* @return will get side2  \*/  public double getSide2() {  return side2;  }  /\*\*  \* setter method for side2  \*  \* @param side2 is set  \*/  public void setSide2(double side2) {  this.side2 = side2;  }  /\*\*  \*  \* @return to check the hashcode is checked  \*/  @Override  public int hashCode() {  int hash = 7;  hash = 59 \* hash + Objects.hashCode(this.shapeName);  hash = 59 \* hash + (int) (Double.doubleToLongBits(this.side1) ^ (Double.doubleToLongBits(this.side1) >>> 32));  hash = 59 \* hash + (int) (Double.doubleToLongBits(this.side2) ^ (Double.doubleToLongBits(this.side2) >>> 32));  return hash;  }  /\*\*  \* equals method  \*  \* @param obj  \* @return -returns boolean value  \*/  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Geometry other = (Geometry) obj;  if (Double.doubleToLongBits(this.side1) != Double.doubleToLongBits(other.side1)) {  return false;  }  if (Double.doubleToLongBits(this.side2) != Double.doubleToLongBits(other.side2)) {  return false;  }  if (!Objects.equals(this.shapeName, other.shapeName)) {  return false;  }  return true;  }  /\*\*  \* toString method  \*  \* @return -returns shapeName, side1, side2  \*/  @Override  public String toString() {  return "shapeName: " + shapeName + "\nside1: " + side1 + "\nside2: " + side2;  }  } |

|  |
| --- |
| /\*  \* 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 prasanna arla  \*/  public class GeomertyDriver {  /\*\*  \* @param args the main driver class  \*/  public static void main(String[] args) {  // TODO code application logic here  /\*\*  \* two objects are created geo1 and geo2  \*/  Geometry geo1 = new Geometry("rectangle", 21, 12);  Geometry geo2 = new Geometry("pentagon", 15, 16);  System.out.println("prasanna arla");  /\*\*  \* if else if statement is given to check by hashcode() are equal or no  \*/  if (geo1.hashCode() == geo2.hashCode()) {  if (geo1.equals(geo2)) {  System.out.println("Both are equal");  } else {  System.out.println("Not equal");  }  } else {  System.out.println("Both are not equal");  }  }  } |

**Output:**

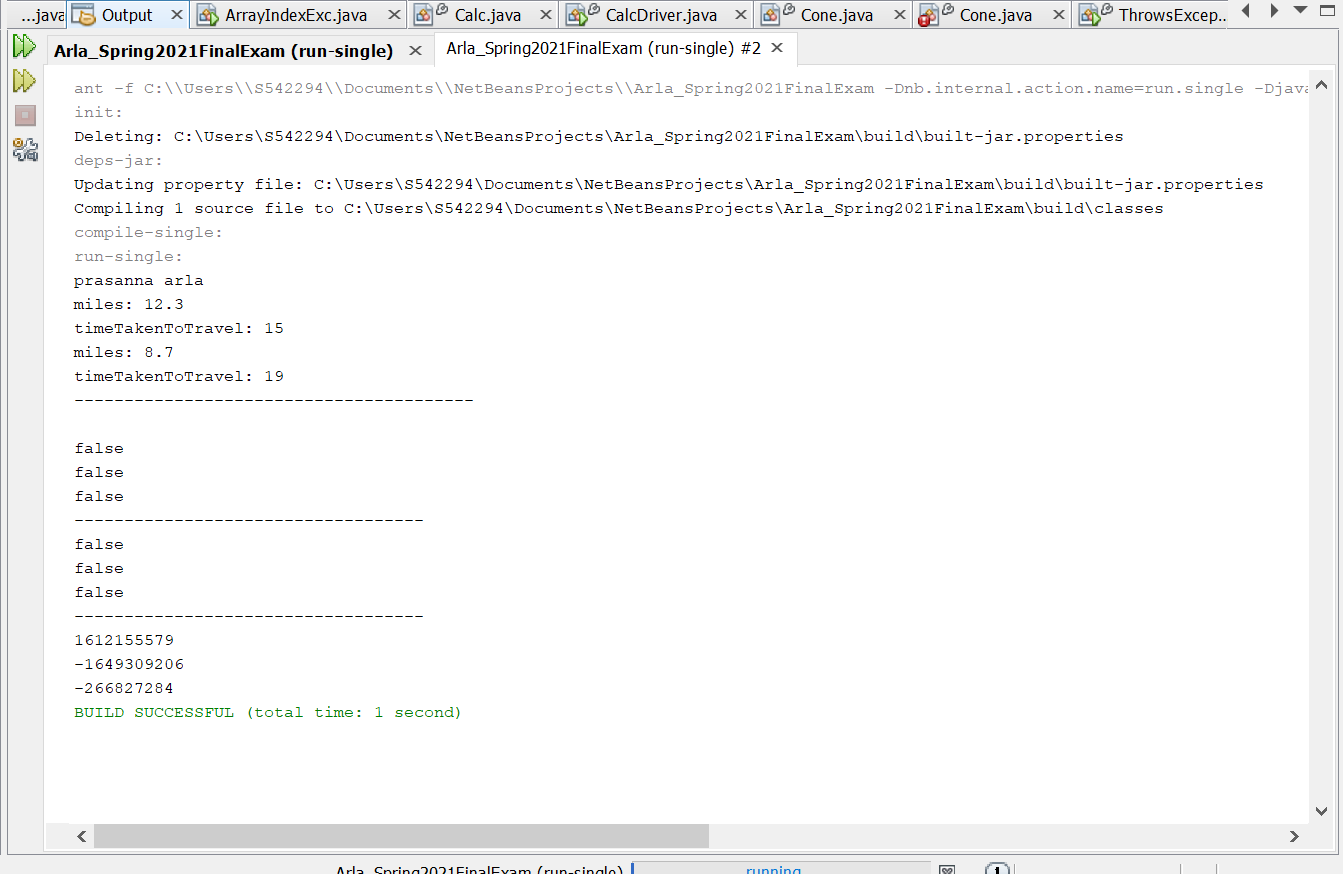


**Example 02:**

|  |
| --- |
| /\*  \* 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 question11example02;  /\*\*  \*  \* @author prasanna arla  \*/  public class Automobiles {  private final double miles;  private final int timeTakenToTravel;  /\*\*  \* two argument constructor  \*  \* @param miles- datatype of double  \* @param timeTakenToTravel -datatype of int  \*/  public Automobiles(double miles, int timeTakenToTravel) {  this.miles = miles;  this.timeTakenToTravel = timeTakenToTravel;  }  /\*\*  \* getter method for miles  \*  \* @return -miles  \*/  public double getMiles() {  return miles;  }  /\*\*  \* getter method for timeTakenToTravel  \*  \* @return -timeTakenToTravel  \*/  public int getTimeTakenToTravel() {  return timeTakenToTravel;  }  /\*\*  \* getter method for speed  \*  \* @return -speed  \*/  public double getSpeed() {  double speed = 0.0;  speed = miles / timeTakenToTravel;  return speed;  }  /\*\*  \* hashcode  \*  \* @return -hash of datatype int  \*/  @Override  public int hashCode() {  int hash = 7;  hash = 71 \* hash + (int) (Double.doubleToLongBits(this.miles) ^ (Double.doubleToLongBits(this.miles) >>> 32));  hash = 71 \* hash + this.timeTakenToTravel;  return hash;  }  /\*\*  \* equals() methods it compare equality of two Objects  \*  \* @param obj  \* @return-booelan  \*/  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Automobiles other = (Automobiles) obj;  if (Double.doubleToLongBits(this.miles) != Double.doubleToLongBits(other.miles)) {  return false;  }  if (this.timeTakenToTravel != other.timeTakenToTravel) {  return false;  }  return true;  }  /\*\*  \* toString method  \*  \* @return -miles, timeTakenToTravel  \*/  @Override  public String toString() {  return "miles: " + miles + "\ntimeTakenToTravel: " + timeTakenToTravel;  }  } |

|  |
| --- |
| /\*  \* 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 question11example02;  /\*\*  \*  \* @author prasanna arla  \*/  public class AutomobilesDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  /\*\*  \* three objects a1,a2,a3 are created and values are given for them  \*/  Automobiles a1 = new Automobiles(12.3, 15);  Automobiles a2 = new Automobiles(8.7, 19);  Automobiles a3 = new Automobiles(6.8, 21);  System.out.println("prasanna arla");  /\*\*  \* prints the toString for the object a1  \*/  System.out.println(a1.toString());  /\*\*  \* prints the toString for the object a2  \*/  System.out.println(a2.toString());  System.out.println("----------------------------------------");  System.out.println(" ");  /\*\*  \* prints the value after comparision of a1 and a2  \*/  System.out.println(a1.equals(a2));  /\*\*  \* prints the value after comparision of a2 and a3  \*/  System.out.println(a2.equals(a3));  /\*\*  \* prints the value after comparision of a3 and a1  \*/  System.out.println(a3.equals(a1));  System.out.println("-----------------------------------");  System.out.println(a1 == a2);  System.out.println(a2 == a3);  System.out.println(a3 == a1);  System.out.println("-----------------------------------");  /\*\*  \* prints the booelan value which is true or false  \*/  System.out.println(a1.hashCode());  System.out.println(a2.hashCode());  System.out.println(a3.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.
* 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:**

**Description:**

**Employee:** This class implements Comparable<Employee>. I have taken the variables named empId, employeeName, empSalary. 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.

**Driver:** In this I have taken an array list named employeeList 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 ‘e’ , the ‘e’ is printed .Then the sorting method takes place here .If emp1 salary is less than the emp2 it returns the-1 .If it is equal it returns the 0 , else +1. Then according o the sorting, it prints the salary**.** Then comparison between the employees takes for the names and by sorting the names are printed. Then employee ‘e’ of employee details is printed.

**Program:**

|  |
| --- |
| /\*  \* 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.Comparator;  /\*\*  \*  \* @author prasanna arla  \*/  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 e) {  int c = ((Integer) this.empId).compareTo(e.empId);  return c;  }  } |

|  |
| --- |
| /\*  \* 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 prasanna arla  \*/  public class EmployeeDriver {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  ArrayList<Employee> employeeList = new ArrayList<Employee>();  employeeList.add(new Employee(03, "prasanna", 11111.67));  employeeList.add(new Employee(01, "pranathi", 10098.89));  employeeList.add(new Employee(100, "kranthi", 87690.76));  employeeList.add(new Employee(76, "vinay", 78965.50));  employeeList.add(new Employee(111, "ammulu", 153762.45));  System.out.println("Prasanna Arla ");  System.out.println("Employees in original order:");  for (Employee e : employeeList) {  System.out.println(e);  }  System.out.println();  Collections.sort(employeeList);  System.out.println("Sorting by emp id: ");  for (Employee e : employeeList) {  System.out.println(e);  }  System.out.println();  Collections.sort(employeeList, new Comparator<Employee>() {  public int compare(Employee emp1, Employee emp2) {  if (emp1.getEmpSalary() < emp2.getEmpSalary()) {  return -1;  } else if (emp1.getEmpSalary() == emp2.getEmpSalary()) {  return 0;  } else {  return +1;  }  }  });  System.out.println("Sorting by the salary:");  for (Employee e : employeeList) {  System.out.println(e);  }  System.out.println();  Collections.sort(employeeList, new Comparator<Employee>() {  public int compare(Employee emp1, Employee emp2) {  return (emp1.getEmployeeName()).compareTo(emp2.getEmployeeName());  }  });  System.out.println("Sorting by name: ");  for (Employee e : employeeList) {  System.out.println(e);  }  System.out.println();  }  } |

**OUTPUT:**

