**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: Abstract class is a class that may or may not have to abstract methods where methods are declared but not defined.

Interface: Interface is similar to abstract class but in interface all the methods are just declared but not implemented.

Similarities for abstract class and interfaces:

|  |  |
| --- | --- |
| Abstract class | Interfaces |
| 1. We cannot create objects in abstract class. | 1.we cannot create object in interfaces. |
| 1. Abstract class cannot be instantiate. | 1. Interfaces cannot be instantiate. |

Difference between abstract class and interfaces:

|  |  |
| --- | --- |
| Abstract class | Interfaces |
| 1. We can use abstract keyword to declare abstract class. | 1.We can use interface keyword to declare interface. |
| 1. Abstract class can hold abstract methods and non-abstract methods. | 2.Interface can hold only abstract methods. |
| 1. Abstract class can be used to extend another class using extends keyword. | 3. Interface can be used to implement using implements keyword. |
| 1. Abstract class can hold constructor, main method and static methods. | 4. Interface cannot hold constructor, main method and static methods. |
| 1. Abstract class can extends only one class. | 5. Interface can implements more than one class. |
| 1. Abstract class cannot achieve multiple inheritance | 6. Interface can achieve multiple inheritance. |

Interfaces are preferred over abstract class:

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

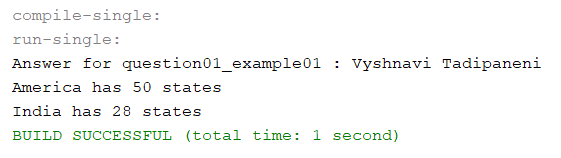
Explanation:

* Creating a abstract class country and creating a abstract method to get no of states.
* Creating a class India and extending the country class to get no of states and it returns the no of states.
* Creating a class America and extending the country class to get no of states and it returns the no of states.
* Creating a Main Method(Driver class) using classes creating an object by taking reference of another class and printing the statement.

Java 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 Question1\_AbstractClass;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public abstract class Country {  public abstract int getNoOfStates();  }//end of country  /\*  \* 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\_AbstractClass;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class America extends Country {  public America() {  }  @Override  public int getNoOfStates() {  return 50;  }  }//end of America  /\*  \* 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\_AbstractClass;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class India extends Country {  public India() {  }  @Override  public int getNoOfStates() {  return 28;  }  }  /\*  \* 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\_AbstractClass;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question1\_Example01 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question01\_example01 : Vyshnavi Tadipaneni");  Country country;  country = new America();  System.out.println("America has " + country.getNoOfStates() + " states");  country = new India();  System.out.println("India has " + country.getNoOfStates() + " states");  }  }//end of main method. |

Output:



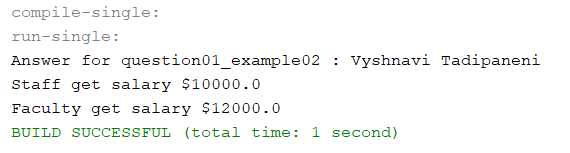
Explanation:

* Creating a abstract class Employee and creating a abstract method to get Salary.
* Creating a class staff and extending the employee class to get no salary and it returns the salary.
* Creating a class Faculty and extending the employee class to get no of states and it returns the salary.
* Creating a Main Method(Driver class) using classes creating an object by taking reference of another class and printing the statement.

Java Program : Example02

|  |
| --- |
| /\*  \* 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\_AbstractClass2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public abstract class Employee {  abstract double getSalary();  }//end of employee  /\*  \* 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\_AbstractClass2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Faculty extends Employee {  @Override  double getSalary() {  return 12000.00;  }  }//end of faculty  /\*  \* 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\_AbstractClass2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Staff extends Employee {  @Override  double getSalary() {  return 10000.00;  }  }//end of staff  /\*  \* 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\_AbstractClass2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question1\_Example02 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question01\_example02 : Vyshnavi Tadipaneni");  Employee employee;  employee = new Staff();  System.out.println("Staff get salary $" + employee.getSalary());  employee = new Faculty();  System.out.println("Faculty get salary $" + employee.getSalary());  }  }//end of main method. |

Output:



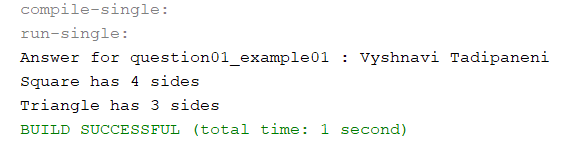
Explanation:

* Creating an interface with Solid and method to get no of sides.
* Creating a square class that implements Solid, by using the get method it returns the no of sides of square.
* Creating a triangle class the implements Solid, by using the get method it returns the no of sides of triangle.
* Creating a Main Method(Driver class) , creating a objects for interface and classes with reference of another class and printing the statement.

Java Program: Example1

|  |
| --- |
| /\*  \* 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\_Interface;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public interface Solid {  public int getNoOfSides();  }//end of solid  /\*  \* 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\_Interface;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Square implements Solid {  public int getNoOfSides() {  return 4;  }  }//end of square  /\*  \* 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\_Interface;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Triangle implements Solid {  public int getNoOfSides() {  return 3;  }  }//end of triangle  /\*  \* 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\_Interface;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question1\_Example01 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question01\_example01 : Vyshnavi Tadipaneni");  Solid x;  x = new Square();  System.out.println("Square has " + x.getNoOfSides() + " sides");  x = new Triangle();  System.out.println("Triangle has " + x.getNoOfSides() + " sides");  }  }//end of main method |

Output:



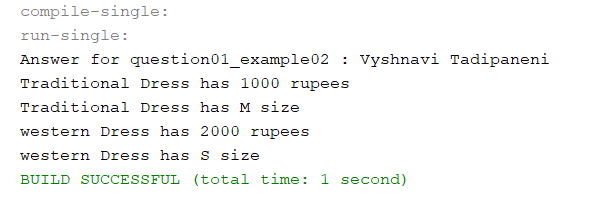
Explanation:

* Creating an interface with Dresses and method to get cost and size.
* Creating a traditionaldresses class that implements dresses, by using the get method it returns the cost and size of a dress.
* Creating a westerndresses class the implements dresses, by using the get method it returns the cost and size of a dress.
* Creating a Main Method(Driver class) , creating a objects for interface and classes with reference of another class and printing the statement.

Java Program : Example02

|  |
| --- |
| /\*  \* 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\_Interface2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public interface Dresses {  public int getCost();  public String getSize();  }//end of dresses  /\*  \* 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\_Interface2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class TraditionalDresses implements Dresses {  public int getCost() {  return 1000;  }  @Override  public String getSize() {  return "M";  }  }//end of traditionaldresses  /\*  \* 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\_Interface2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class WesternDresses implements Dresses {  public int getCost() {  return 2000;  }  @Override  public String getSize() {  return "S";  }  }//end of westerndresses  /\*  \* 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\_Interface2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question1\_Example2 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question01\_example02 : Vyshnavi Tadipaneni");  Dresses x;  x = new TraditionalDresses();  System.out.println("Traditional Dress has " + x.getCost() + " rupees");  System.out.println("Traditional Dress has " + x.getSize() + " size");  x = new WesternDresses();  System.out.println("western Dress has " + x.getCost() + " rupees");  System.out.println("western Dress has " + x.getSize() + " size");  }  }//end of main method |

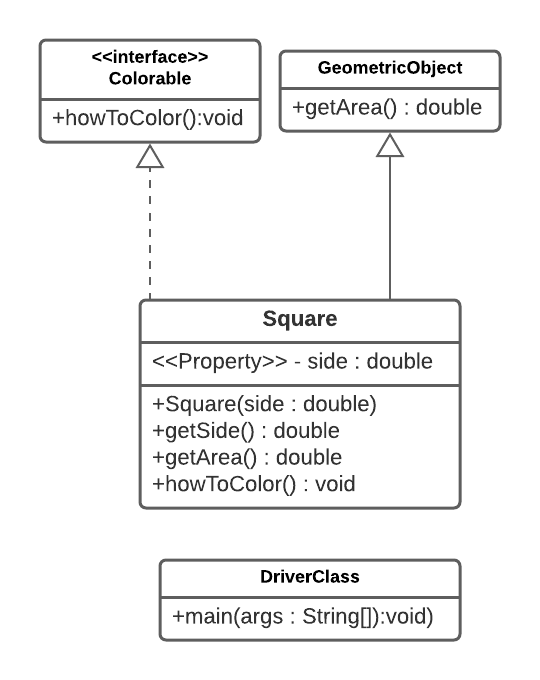
Output:



2.(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:

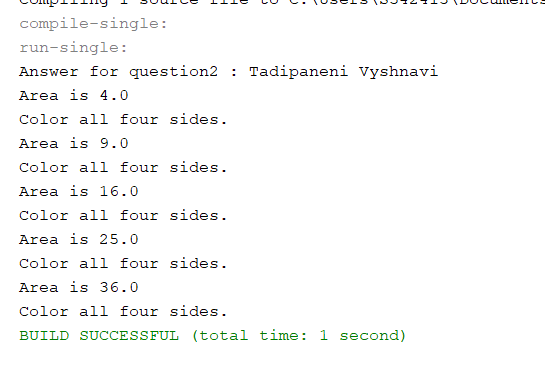
UML Diagram:



Java 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 Question2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public interface Colorable {  public void howToColor();  }//end of colorable interface  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public abstract class GeometricObject {  public abstract double getArea();  }//end of geometric object  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Square extends GeometricObject implements Colorable {    public double side;  public Square(double side) {  this.side = side;  }  public double getSide() {  return side;  }  public void setSide(double side) {  this.side = side;  }    @Override  public double getArea() {  return side \* side;  }  @Override  public void howToColor() {  System.out.println("Color all four sides.");  }  }//end of square  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question2;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class DriverClass {  public static void main(String[] args) {  /\*\*  \* @param args the command line arguments  \*/  System.out.println("Answer for question2 : Tadipaneni Vyshnavi");  GeometricObject[] objects = {new Square(2), new Square(3), new Square(4), new Square(5),  new Square(6)};  for (int i = 0; i < objects.length; i++) {  System.out.println("Area is " + objects[i].getArea());  if (objects[i] instanceof Colorable) {  ((Colorable) objects[i]).howToColor();  }  }  }  }//end of main method |

Output:



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

Answer:

Casting: It is a process of converting one data type to another data type is known as casting. Type casting is when you assign a value of one primitive data type to another type.

They are two types of casting in java. They are:

1. Widening type casting
2. Narrowing type casting

Widening type casting: casting: It is a process of converting lower data type is converted into higher data type since there is no loss of data. This is also known as implicit type casting.

Narrowing type casting: It is a process of converting higher data type is converted into lower data type since there is no loss of data. This is also known as explicit type casting.

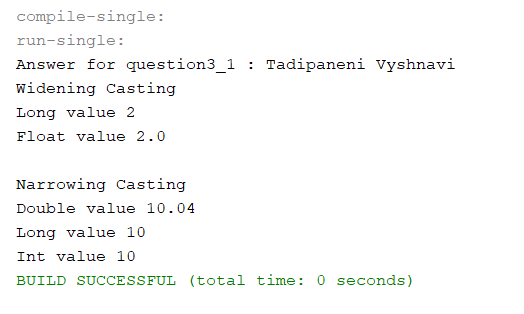
Explanation:

* Creating a main method in widening type casting to print the statement in order.
* Here, Iam creating an objects using primitive datatype to convert lower data type to higher data type.
* In this program I have used int, float and long and printing the statements.
* Creating narrowing type casting to print the statement in order.
* Creating an object using primitive data type to convert higher data type to lower data type.
* In this program I have used double, long and int and printing the statements.

Java 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 Question3;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question3\_1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  // Widening Casting  System.out.println("Answer for question3\_1 : Tadipaneni Vyshnavi");  System.out.println("Widening Casting");  int number = 2;  long longvalue = number; // Widening casting from int to long  float floatValue = longvalue; // Widening casting from long to float  System.out.println("Long value " + longvalue);  System.out.println("Float value " + floatValue);  // Narrowing Casting  System.out.println("\nNarrowing Casting");  double doubleNumber = 10.04;  longvalue = (long) doubleNumber; // Narrowing casting from double to long  number = (int) longvalue; // Narrowing casting from double to int  System.out.println("Double value " + doubleNumber);  System.out.println("Long value " + longvalue);  System.out.println("Int value " + number);  }  } |

Output:



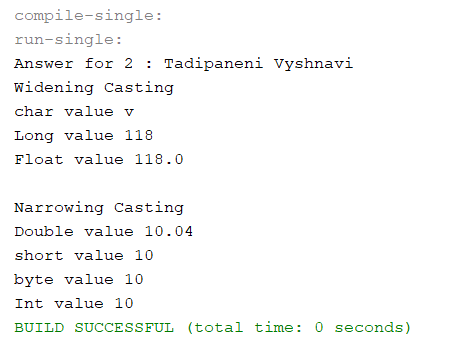
Explanation:

* Creating a main method in widening type casting to print the statement in order.
* Here, Iam creating an objects using primitive datatype to convert lower data type to higher data type.
* In this program I have used char, long and float and printing the statements.
* Creating narrowing type casting to print the statement in order.
* Creating an object using primitive data type to convert higher data type to lower data type.
* In this program I have used short, byte and int and printing the statements.

Java Program: Example02

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question3;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question3\_2 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for 2 : Tadipaneni Vyshnavi");  // Widening Casting  System.out.println("Widening Casting");  char a = 'v';  long longvalue = a; // Widening casting from int to long  float floatValue = longvalue; // Widening casting from long to float  System.out.println("char value " + a);  System.out.println("Long value " + longvalue);  System.out.println("Float value " + floatValue);  // Narrowing Casting  System.out.println("\nNarrowing Casting");  double doubleNumber = 10.04;  short shortvalue = (short) doubleNumber;//Narrowing casting from double to short  byte bytevalue = (byte) doubleNumber; // Narrowing casting from double to long  int number = (int) doubleNumber; // Narrowing casting from double to int  System.out.println("Double value " + doubleNumber);  System.out.println("short value " + shortvalue);  System.out.println("byte value " + bytevalue);  System.out.println("Int value " + number);  }  } |

Output:



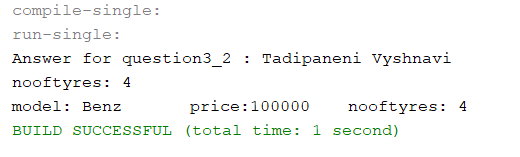
Explanation:

* Creating a vehicle class using variables, constructor and methods.
* Creating a car class using variables, constructor and methods.
* Creating a main method in widening type casting to print the statement in same order by using classes.
* Creating an object inside the main method and printing the statements by lower data type to higher data type.

Java 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 Question3\_Example\_widening;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Vehicle {  private int nooftyres;  public Vehicle(int nooftyres) {  this.nooftyres = nooftyres;  }  public int getNooftyres() {  return nooftyres;  }  public void setNooftyres(int nooftyres) {  this.nooftyres = nooftyres;  }  @Override  public String toString() {  return "nooftyres: " + nooftyres;  }  }//end of vehicle  /\*  \* 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\_Example\_widening;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Car extends Vehicle {  private String model;  private int price;  public Car(String model, int price, int nooftyres) {  super(nooftyres);  this.model = model;  this.price = price;  }  public String getModel() {  return model;  }  public void setModel(String model) {  this.model = model;  }  public int getPrice() {  return price;  }  public void setPrice(int price) {  this.price = price;  }  @Override  public String toString() {  return "model: " + model + "\t price:" + price + "\t" + super.toString();  }  }//end of car  /\*  \* 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\_Example\_widening;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question3\_02 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Vehicle vehicle = new Vehicle(4);  System.out.println("Answer for question3\_2 : Tadipaneni Vyshnavi");  System.out.println(vehicle.toString());  Vehicle car = new Car("Benz", 100000, 4);  System.out.println(car.toString());  }  }//end of main method. |

Output:



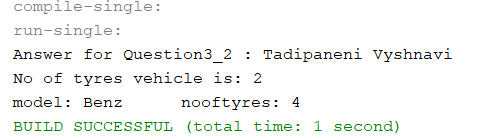
Explanation:

* Creating a vehicle class using variables, constructor and methods.
* Creating a car class using variables, constructor and methods.
* Creating a main method in narrowing type casting to print the statement in same order by using classes.
* Creating an object inside the main method and printing the statements by lower data type to higher data type.

Java Program :Narrowing type casting

|  |
| --- |
| /\*  \* 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\_Example\_Narrowing;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Vehicle {  private int nooftyres;  public Vehicle(int nooftyres) {  this.nooftyres = nooftyres;  }  public int getNooftyres() {  return nooftyres;  }  public void setNooftyres(int nooftyres) {  this.nooftyres = nooftyres;  }  @Override  public String toString() {  return "\tnooftyres: " + nooftyres;  }  }//end of vehicle  /\*  \* 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\_Example\_Narrowing;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Car extends Vehicle {  private String model;  public Car(String model, int nooftyres) {  super(nooftyres);  this.model = model;  }  public String getModel() {  return model;  }  public void setModel(String model) {  this.model = model;  }  @Override  public String toString() {  return "model: " + model + super.toString();  }  }//end of car  /\*  \* 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\_Example\_Narrowing;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question3\_2 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Vehicle vehicle = new Vehicle(2);  System.out.println("Answer for Question3\_2 : Tadipaneni Vyshnavi");  System.out.println("No of tyres vehicle is: " + vehicle.getNooftyres());  Car car = new Car("Benz", 4);  System.out.println(car.toString());  vehicle = car;//Polymorphic substitution  car = (Car) vehicle;//narrow type Casting  }  }//end of main method. |

Output:



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

Fruit

Orange

Apple

GoldenDelicious

McIntosh

Assume that the following code is given:

Fruit fruit = new GoldenDelicious();

Orange orange = new Orange();

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

1. Is fruit instanceof Fruit?

|  |
| --- |
| Answer: legal  Explanation: we can observer that Fruit is parent class for of all the classes and any sub class object assigned to parent object will be instance of parent class i.e, Fruit. |

1. Is fruit instanceof Orange?

|  |
| --- |
| Answer: illegal  Explanation: There is no direct  relation exists between GoldenDelicious and Orange. The parent class of GoldenDelicious is Apple class for which  the parent class of is Fruit |

1. Is fruit instanceof Apple?

|  |
| --- |
| Answer: legal  Explanation: Yes, fruit is an object of GoldenDelicious class and  parent class of GoldenDelicious is Apple. |

d.  Is fruit instanceof GoldenDelicious?

|  |
| --- |
| Answer: legal  Explanation: fruit is an object of GoldenDelicious class which is an object to GoldenDelicious class itself. |

e.  Is fruit instanceof McIntosh?

|  |
| --- |
| Answer: illegal  Explanation: fruit is an object of GoldenDelicious and it is  direct subclass of Fruit class.  McIntosh is the subclass of Apple as there is no relation between McIntosh and Goldendelicious classes. |

f.  Is orange instanceof Orange?

|  |
| --- |
| Answer :legal  Explanation: As we can see clearly orange is an instance for  Orange class. |

g.  Is orange instanceof Fruit?

|  |
| --- |
| Answer :illegal  Explanation: since orange object is an instance of Orange class which is subclass of parent class Fruit. |

h.  Is orange instanceof Apple?

|  |
| --- |
| Answer:illegal  Explanation: Apple  and Orange classes are subclasses of Fruit class and there is no direct relation between Orange and Apple classes |

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

|  |
| --- |
| Answer: legal  Explanation: The method makeAppleCider is specified in the Apple class and cannot be called using a fruit object. The method from the Apple class cannot be invoked by an orange object because the Orange and Apple classes are subclasses of the Fruit class and are unrelated. Apple is the parent class of GoldenDelicious class. However, the generated object is of the Fruit type, which is a parent. |

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

|  |
| --- |
| Answer: legal  Explanation: Since the makeOrangeJuice method is in the Orange class and orange is an object for Orange class, orange can invoke the makeOrangeJuice method, whereas fruit is an instance of the GoldenDelicious class and there is no relationship between the Orange and GoldenDelicious class, so fruit cannot call the makeOrangeJuice method. |

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

|  |
| --- |
| Answer: illegal  Explanation: It is not legal because Orange class and Apple class are  different sub classes under the parent class Fruit. |

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

|  |
| --- |
| Answer: Illegal  Explanation: It's not legal since we can't  create an instance of parent class with child class reference.  The parent class is Apple, while the sub class is MacIntosh. |

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

|  |
| --- |
| Answer: legal  Explanation: Because we can create an instance of child class using the reference of parent class. The parent class is Apple, while the sub class is MacIntosh. |

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

Answer:

UML Diagram:

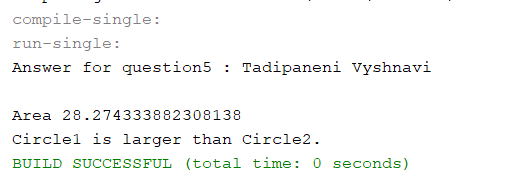
Diagram

Description automatically generated

Java 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 Question5;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Circle {  private double radius;    public Circle(double radius) {  this.radius = radius;  }  public double getRadius() {  return radius;  }    public double getArea() {  return radius \* radius \* Math.PI;  }    }//end of circle  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class ComparableCircle extends Circle implements Comparable<ComparableCircle> {  public ComparableCircle(double radius) {  super(radius);  }  @Override  public int compareTo(ComparableCircle circle) {  if (getArea() > circle.getArea()) {  return 1;  } else if (getArea() < circle.getArea()) {  return -1;  } else {  return 0;  }  }  @Override  public String toString() {  return "\nArea " + super.getArea();  }  }end of comparable circle  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question5;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class DriverClass {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question5 : Tadipaneni Vyshnavi");  ComparableCircle circle1 = new ComparableCircle(3);  ComparableCircle circle2 = new ComparableCircle(2);  System.out.println(circle1.toString());  if (circle1.compareTo(circle2) > 0) {  System.out.println("Circle1 is larger than Circle2.");  } else if (circle1.compareTo(circle2) < 0) {  System.out.println("Circle2 is larger than Circle1.");  } else {  System.out.println("Both circles are the same");  }  }  }//end of main method. |

Output:



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

Answer:

Exception: An exception is a problem that arises during the exception of a program. When exception is occurs the normal flow of program gets disturbed and program terminated.

They are two types of exceptions. They are:

1. Checked exceptions.
2. Unchecked exceptions.

Checked Exception: It occurs at the compiles time. This is also known as compile-time exceptions. All the checked exceptions are not simply ignored, the programmer should handle these exceptions.

1. Class Not Found Exception.
2. File Not Found Exception.
3. SQL Exception.
4. IO Exception.

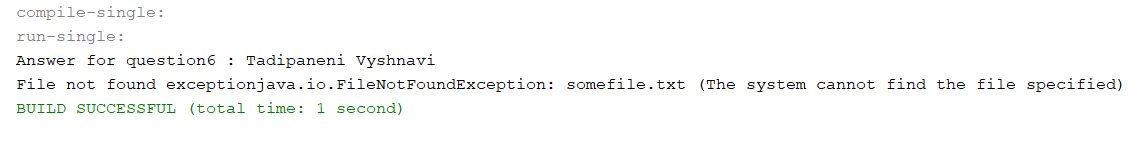
Explanation :

* Creating a main method to print the exception error message.
* Creating try block inside the try block creating a fileReader object and placing the file location path.
* And printing the error message by using catch block.

Java 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 Question6;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question6\_1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws FileNotFoundException {  // TODO code application logic here  System.out.println("Answer for question6 : Tadipaneni Vyshnavi");  try {  FileReader file1 = new FileReader("somefile.txt");  FileReader file2;  } catch (FileNotFoundException ex) {  System.out.println("File not found exception" + ex);  }  }  } |

Output:



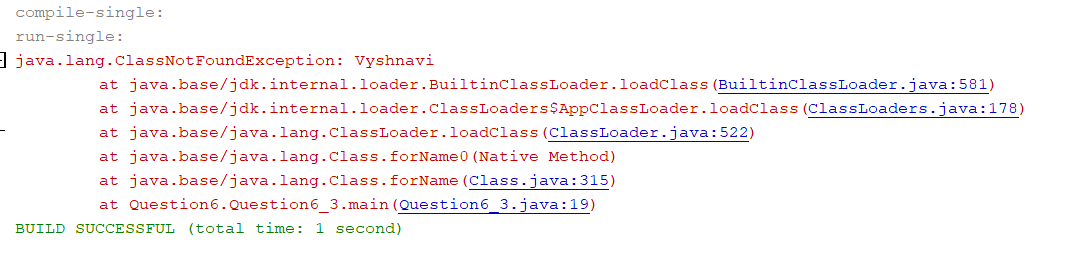
Explanation:

* Creating a main method to print the exception error message.
* Creating try block inside the try block creating a class object and placing the file location path.
* And printing the error message by using catch block.

Java 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 Question6;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question6\_3 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String args[]) {  try {  Class.forName("Vyshnavi");  } catch (ClassNotFoundException ex) {  ex.printStackTrace();  }  }  } |

Output:



Unchecked Exception: It occurs at the run time. This is also known as run-time exceptions. These includes programming bugs, such as logic errors or improper use of API. Runtime exceptions are ignored at the time of execution.

1. Arithmetic Exception.
2. Null Pointer Exception.
3. Class Cast Exception.
4. Illegal Argument Exception.

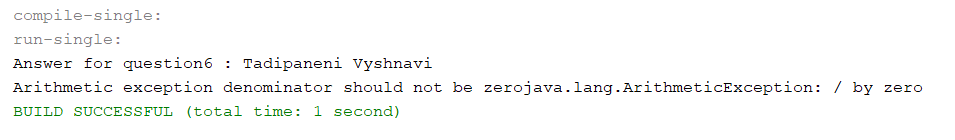
Explanation :

* Creating a main method to print the exception error message.
* Initailizing the values by using the datatype.
* Creating try block inside the try block writing the excutable code to run the program and printing the statement.
* And printing the error message by using catch block.

Java 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 Question6;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question6\_2 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int num1 = 10;  int num2 = 0;  System.out.println("Answer for question6 : Tadipaneni Vyshnavi");  try {  int res = num1 / num2;  System.out.println(res);  } catch (ArithmeticException e) {  System.out.println("Arithmetic exception denominator should not be zero" + e);  }  }  } |

Output:



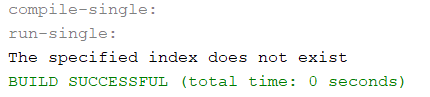
Explanation:

* Creating a main method to print the exception error message.
* Initailizing the values by using the datatype.
* Creating try block inside the try block writing the excutable code to run the program and printing the statement.
* And printing the error message by using catch block.

Java 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 Question6;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question6\_4 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  try {  int arr[] = {11, 21, 51, 71, 90, 61};  System.out.println(arr[7]);  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("The specified index does not exist");  }  }  } |

Output:



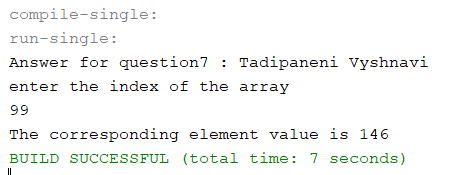
1. (10-Points) Write a program that meets the following requirements:

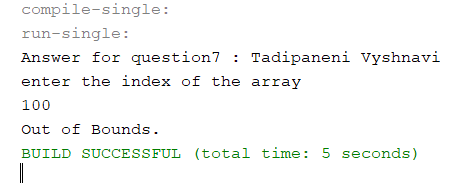
* Creates an array with 100 randomly chosen integers.
* Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message Out of Bounds.

Answer:

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package Question7;  import java.util.Scanner;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question\_7 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question7 : Tadipaneni Vyshnavi");  int[] arr = new int[100];  for (int i = 0; i < arr.length; i++) {  arr[i] = (int) (Math.random() \* 1000) + 1;  }  Scanner input = new Scanner(System.in);  System.out.println("enter the index of the array ");  try {  System.out.println("The corresponding element value is " + arr[input.nextInt()]);  } catch (ArrayIndexOutOfBoundsException ex) {  System.out.println("Out of Bounds.");  }  }  } |

Output:





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

Answer:

The purpose of declaring exceptions is informed by the java interpreter. The main method starts executing a java program. Every method must declare the types of checked exceptions it might throw and java does not need to declare unchecked exceptions in this method.

Throwing an exception is simple as the throw statement. Each and every exception includes a message which is human readable error description. It often be related to problems with user input, server, backend etc.

Yes, we can declare multiple exceptions in a method header. If the method declare multiple exceptions, add to list of exceptions, separated by comma after throws.

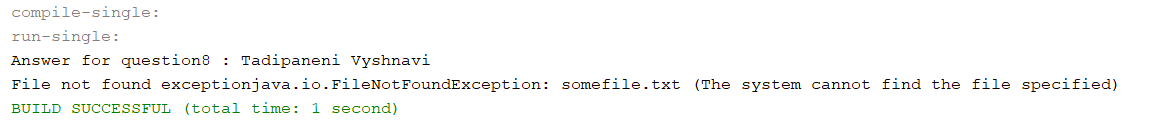
Explanation:

* Creating a main method to print the statements and throwing the exceptions.
* Initializing the filereader.
* Creating a try block inside the try block creating an object for file to place the specified path location.
* Creating a multiple catch block to print the error message if it is error.

Java 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 Question8;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question8\_1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws FileNotFoundException, ArithmeticException, IOException {  // TODO code application logic here  System.out.println("Answer for question8 : Tadipaneni Vyshnavi");  FileReader file2;  try {  file2 = new FileReader("somefile.txt");  file2.read();  } catch (FileNotFoundException e) {  System.out.println("File not found exception" + e);  } catch (ArithmeticException | IOException e) {  System.out.println("Arithmetic exception" + e);  }  }  } |

Output:



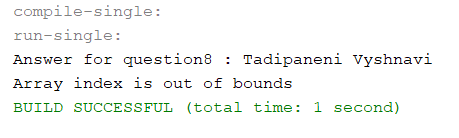
Explanation:

* Creating a class inside the class creating an arraylist using static method.
* Creating a main method inside the main method writing try block inside the try block initializing some array and applying some value to it.
* And writing the error message to move into catch block if the condition fails.

Java 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 Question8;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question8\_2 {  public static int[] array1 = new int[10];  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question8 : Tadipaneni Vyshnavi");  try {  array1[11] = 9;  System.out.println("Array index is valid");  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("Array index is out of bounds");  }  }  } |

Output:



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

Answer:

The throwkeyword is usedto explicitly throw an exception within a method or block of code. It gives an information to the programmer 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.

The throwskeyword is usedto declare which exceptions can be thrown from a method. The throwskeyword is usedin a method signature and declares which exceptions can be thrown from a method.

Yes, If your code throws more than oneexception, youcan choose ifyou want to use a separate try block for each statement that couldthrow an exception or use one try block for multiplestatements that might throwmultipleexceptions.

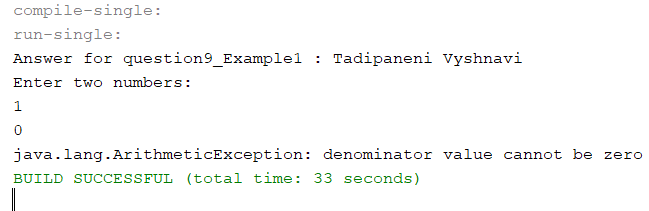
Explanation:

* Creating a main method and printing the statements and by throwing the exceptions.
* Enter the two numbers, and initialing the two numbers using scanner class.
* Creating a try block, inside the try block writing if condition, if condition is true it will throw the exception.
* Or else it will move to the catch block and print the error message.

Java 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 Question9;  import java.util.Scanner;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question9\_1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws ArithmeticException {  // TODO code application logic here  Scanner scan = new Scanner(System.in);  System.out.println("Answer for question9\_Example1 : Tadipaneni Vyshnavi");  System.out.println("Enter two numbers:");  int a = scan.nextInt();  int b = scan.nextInt();  try {  if (b == 0) {  throw new ArithmeticException("denominator value cannot be zero");  }  } catch (ArithmeticException e) {  System.out.println(e);  }  }  } |

Output:



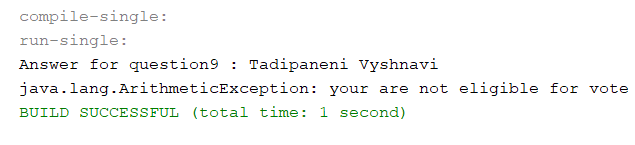
Explanation:

* Creating a class using a method getagerestrictions and initializing the age.
* Inside the method writing if condition, if condition is true if condition will excute or else else block.
* Creating a main method and printing the statements and by throwing the exceptions.
* Enter the two numbers, and initialing the two numbers using scanner class.
* Creating a try block, inside the try block writing if condition, if condition is true it will throw the exception.
* Or else it will move to the catch block and print the error message.

Java 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 Question9;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question9\_2 {  public void getageRestriction(int age) throws ArithmeticException {  if (age < 21) {  throw new ArithmeticException("your are not eligible for vote");  } else {  System.out.println("Thanks for voting");  }  }  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question9 : Tadipaneni Vyshnavi");  try {  Question9\_2 throwsexample = new Question9\_2();  throwsexample.getageRestriction(20);  } catch (ArithmeticException e) {  System.out.println(e);  }  }  } |

Output:



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

Answer:

Recursive method: A method that calls itself is known as a recursive method. And, this process is Known as recursion.

Infinite recursion: Infinite recursion is the non-terminating execution of a block of code. Infinite recursion is usually caused by a bug in code.

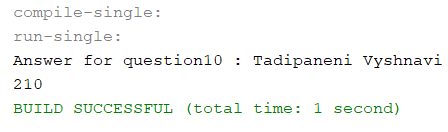
Explanation:

* Creating a main method to print the sum of 20 integer values.
* Initializing the value to a, and printing the a value.
* Creating a method using static and initializing a variable and writing the if condition inside the method.
* If condition is true it will execute or else else block will execute and returns 0.

Java 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 Question10;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question\_10\_1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for question10 : Tadipaneni Vyshnavi");  int a = sum(20);  System.out.println(a);  }  public static int sum(int i) {  if (i > 0) {  return i + sum(i - 1);  } else {  return 0;  }  }  } |

Output:



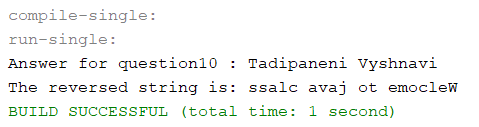
Explanation:

* Creating a main method to print reverse of a string.
* Intializing the string and value to it and printing the statement.
* And writing the reverse of a string method to reverse the string if string is empty returns null, if the string has any value it will reverse the string and prints the value.

Java 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 Question10;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question\_10\_2 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  String str = "Welcome to java class";  String reversed = reverseString(str);  System.out.println("Answer for question10 : Tadipaneni Vyshnavi");  System.out.println("The reversed string is: " + reversed);  }  public static String reverseString(String str) {  if (str.isEmpty()) {  return str;  }  return reverseString(str.substring(1)) + str.charAt(0);  }  } |

Output:



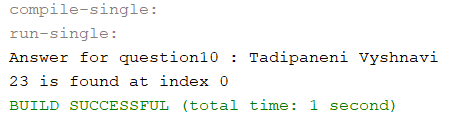
Explanation:

* Creating a main method to print the search element in the array list.
* Creating a class in linear search and writing a method inside it, if the condition is true it returns the search values.
* If the condition fails it will else if condition and returns the search value
* If the condition fails again it will go to the else condition and returns the search value.
* Initializing the list of array list of elements and finding the index values and printing the value in the array list with index value.

Java 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 Question10;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question\_10 {  static int linearSearch(int a[], int start, int end, int number) {  if (start > end) {  return -1;  } else if (a[start] == number) {  return start;  } else if (a[end] == number) {  return end;  }  return linearSearch(a, start + 1, end - 1, number);  }  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  int[] a = {23, 2, 4, 6, 12, 7, 1, 99};  int index = linearSearch(a, 0, a.length - 1, 23);  System.out.println("Answer for question10 : Tadipaneni Vyshnavi");  if (index != -1) {  System.out.println(23 + " is found at index " + index);  } else {  System.out.println(23 + " is not present");  }  }  } |

Output:



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

Answer :

Hash code(): HashCode() returns an integer value, generated by a hashing algorithm. Objects that are equal (according to their equals()) must return the same hashcode.

Equals(): Equals() method compares the two given strings based on the content of the string. If any character is not matched, it returns false. If all characters are matched, it returns true. The String equals() method overrides the equals() method of Object class.

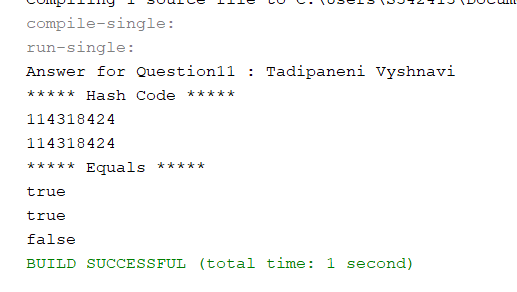
Explanation:

* Creating a class bottle using variables, constructor and methods
* Creating a getter and setter methods and hashcode, equals methods.
* Creating a main method to print the hashcode values and equalizing the two objects.
* Creating a two objects and initializing the values inside the objects.
* And printing the hashcode values using the objects and printing the hashcode values.
* Comparing the two objects and printing the value using Boolean datatype.

JavaProgram:

|  |
| --- |
| /\*  \* 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;  /\*\*  \* creating a class using variables,methods and constructor.  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Bottle {  //Initializing variable using datatype and access modifier.  private String color;  //Initializing variable using datatype and access modifier.  private String type;  /\*\*  \* Creating constructor with arguments.  \*  \* @param color  \* @param type  \*/  public Bottle(String color, String type) {  this.color = color;  this.type = type;  }  /\*\*  \* creating a method to get color.  \*  \* @return  \*/  public String getColor() {  return color;  }  /\*\*  \* creating a method to get type of a bottle.  \*  \* @return  \*/  public String getType() {  return type;  }  /\*\*  \* creating a method to get hashCode values.  \*  \* @return  \*/  @Override  public int hashCode() {  int hash = 3;  hash = 67 \* hash + Objects.hashCode(this.color);  hash = 67 \* hash + Objects.hashCode(this.type);  return hash;  }  /\*\*  \* creating a method to get equals method.  \*  \* @param obj  \* @return  \*/  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Bottle other = (Bottle) obj;  if (!Objects.equals(this.color, other.color)) {  return false;  }  if (!Objects.equals(this.type, other.type)) {  return false;  }  return true;  }  }//end of bottle  /\*  \* 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;  /\*\*  \* creating a main method to print the values.  \*  \* @author Tadipaneni Vyshnavi  \*/  public class Question11 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  System.out.println("Answer for Question11 : Tadipaneni Vyshnavi");  //Creating an object using constructor variables.  Bottle bottle1 = new Bottle("red", "plain");  //creating an object using constructor variables.  Bottle bottle2 = new Bottle("red", "plain");  System.out.println("\*\*\*\*\* Hash Code \*\*\*\*\* ");  //Printing hashcode value for bottle object1.  System.out.println(bottle1.hashCode());  //Printing hashcode value for bottle object2.  System.out.println(bottle2.hashCode());  System.out.println("\*\*\*\*\* Equals \*\*\*\*\* ");  //Printing an statement that compares two objects if it is same print true or else false.  System.out.println(bottle1.equals(bottle2));  //Printing an statement that compares two objects if it is same print true or else false.  System.out.println(bottle2.equals(bottle1));  //Printing an statement that compares bottle object1 with bottle object2 by calling the method name.  System.out.println(bottle1.equals(bottle2.getColor()));  }  }//end of main method. |

Output:



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

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

1. **EmployeeDriver Class**

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

Answer:

|  |
| --- |
| /\*  \* 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 Tadipaneni Vyshnavi  \*/  public class Employee implements Comparable<Employee> {  private int empId;  private String empName;  private double empSalary;  public Employee(int empId, String empName, double empSalary) {  this.empId = empId;  this.empName = empName;  this.empSalary = empSalary;  }  public int getEmpId() {  return empId;  }  @Override  public String toString() {  return empId + " " + empName + " " + empSalary;  }  public String getEmpName() {  return empName;  }  public double getEmpSalary() {  return empSalary;  }  @Override  public int compareTo(Employee o) {  return this.empId - o.empId;  }  }//end of employee 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 Question12;  import java.util.Comparator;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class NameComparator implements Comparator<Employee> {  @Override  public int compare(Employee o1, Employee o2) {  return o1.getEmpName().compareTo(o2.getEmpName());  }  }//end of Name comparator.  /\*  \* 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 Tadipaneni Vyshnavi  \*/  public class SalaryComparator implements Comparator<Employee> {  @Override  public int compare(Employee o1, Employee o2) {  return (int) (o1.getEmpSalary() - o2.getEmpSalary());  }  }//end of salary comparator.  /\*  \* 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;  /\*\*  \*  \* @author Tadipaneni Vyshnavi  \*/  public class DriverClass {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  Employee employee1 = new Employee(1, "EmployeeX", 1000);  Employee employee2 = new Employee(2, "EmployeeA", 1200);  Employee employee3 = new Employee(3, "EmployeeR", 800);  Employee employee4 = new Employee(4, "EmployeeS", 920);  Employee employee5 = new Employee(5, "EmployeeM", 1500);  ArrayList<Employee> employees = new ArrayList<>();  employees.add(employee2);  employees.add(employee1);  employees.add(employee5);  employees.add(employee4);  employees.add(employee3);  System.out.println("Answer for question12 : Tadipaneni Vyshnavi");  System.out.println("Default Employee");  for (int i = 0; i < employees.size(); i++) {  System.out.println(employees.get(i));  }  Collections.sort(employees);  System.out.println("");  System.out.println("Sort the list by its natural order:");  for (int i = 0; i < employees.size(); i++) {  System.out.println(employees.get(i));  }  Collections.sort(employees, new SalaryComparator());  System.out.println("");  System.out.println("Sort the list by supplying a new Comparator<Employee> that sorts by salary:");  for (int i = 0; i < employees.size(); i++) {  System.out.println(employees.get(i));  }  Collections.sort(employees, new NameComparator());  System.out.println("");  System.out.println("Sort the list by supplying a new Comparator<Employee> that sorts by name:");  for (int i = 0; i < employees.size(); i++) {  System.out.println(employees.get(i));  }  }  }//end of main method. |

Output:

