### **Introduction to Java 2**

**Ans 1.**

/\*Create Java classes having suitable attributes for Library management system.Use OOPs concepts in your design.Also try to use interfaces and abstract classes.

\*/

**Skeleton :**

public class Q1LibraryManagementSystem {

public static void main(String[] args) {

}

interface adminLogin

{

void status();

void issuedBookDetails();

void calculateFine();

void sendMail();

void activateBook();

void deactivateBook();

}

interface memberLogin

{

void issuedBooks();

void finePayment();

void searchBook();

}

interface addBook

{

void add();

}

interface removeBook

{

void remove();

}

class Admin implements adminLogin, addBook, removeBook, searchBook, issueBook, calculateFine

{

int id;

String password;

public void showLibraryDetails()

{

}

@Override

public void status() {

}

@Override

public void issuedBookDetails() {

}

@Override

public void calculateFine() {

}

@Override

public void sendMail() {

}

@Override

public void activateBook() {

}

@Override

public void deactivateBook() {

}

@Override

public void remove(){

}

@Override

public void add() {

}

@Override

public void search() {

// TODO Auto-generated method stub

}

@Override

public void issue() {

// TODO Auto-generated method stub

}

@Override

public void calculate() {

// TODO Auto-generated method stub

}

}

class Book

{

int isbn;

String author;

float price;

int quantity;

String type;

}

class Member implements searchBook, issueBook

{

int id;

String password;

@Override

public void search() {

}

@Override

public void issue() {

}

}

interface searchBook

{

void search();

}

interface issueBook

{

void issue();

}

interface calculateFine

{

void calculate();

}

}

**Ans 2.**

/\*WAP to sort string without using string Methods

\*/

public class Q2StringSort {

public static void main(String[] args) {

String string = "name";

int j=0;

char temp=0;

char[] chars = string.toCharArray();

for (int i = 0; i <chars.length; i++) {

for ( j = 0; j < chars.length; j++) {

if(chars[j]>chars[i]){

temp=chars[i];

chars[i]=chars[j];

chars[j]=temp;

}

}

}

for(int k=0;k<chars.length;k++){

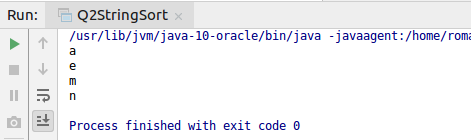
System.out.println(chars[k]);

}

}

}

Output:



**Ans 3.**

/\*WAP to produce NoClassDefFoundError and ClassNotFoundException exception

\*/

Class MyClass

{

Int i;

}

import java.lang.Exception;

public class Main

{

public static void main(String[] args) {

try

{

My my = new My();

System.out.print(my.i);

}

catch(Exception e)

{

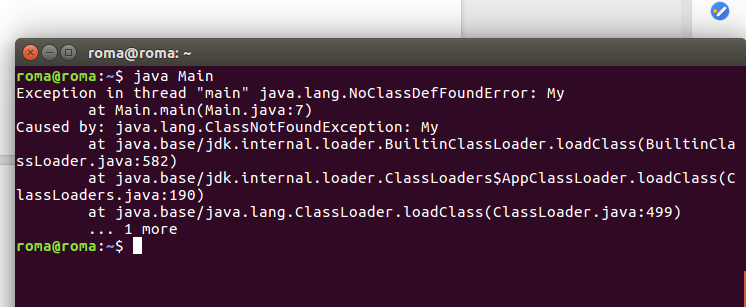
System.out.print(e.getMessage());

}

}

}

Output



ClassNotFoundException

public class MainClass

{

public static void main(String[] args)

{

try

{

Class.forName("System.java.lang");

}catch (ClassNotFoundException e)

{

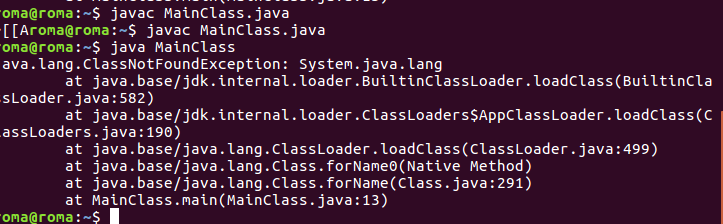
e.printStackTrace();

}

}

}

Output:



**Ans 4.**

/\*WAP to create singleton class.

\*/

public class Q4Demo {

public static void main(String[] args) {

Singleton obj = Singleton.getInstance();

obj.dispaly();

}

}

class Singleton {

private static Singleton singletonObject = new Singleton();

private Singleton()

{

}

public static Singleton getInstance(){

return singletonObject;

}

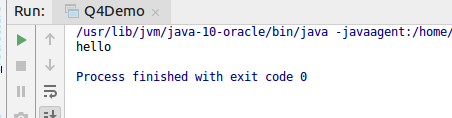
public void dispaly(){

System.out.println("hello");

}

}

Output



**Ans 5.**

/\*WAP to show object cloning in java using cloneable and copy constructor both.

\*/

**public class** Q5Employee **implements** Cloneable {

**private int id**;

**private** String **name**;

**private** Double **salary**;

**private** String **designation**;

Q5Employee(**int** empId, String empName, Double empSalary, String empDesignation)

{

**this**.**id** = empId;

**this**.**name** = empName;

**this**.**salary** = empSalary;

**this**.**designation** = empDesignation;

}

Q5Employee(Q5Employee emp)

{

**this**.**id** = emp.**id**;

**this**.**name** = emp.**name**;

**this**.**salary** = emp.**salary**;

**this**.**designation** = emp.**designation**;

}

**public** Object clone()**throws** CloneNotSupportedException{

**return super**.clone();

}

**public static void** main(String[] args) {

Q5Employee employeeOne = **new** Q5Employee(1, **"Roma"**, 15000.5, **"JVM"**);

Q5Employee employeeTwo = **new** Q5Employee(employeeOne);

System.***out***.println(employeeOne.**id**+**" "**+employeeOne.**name** + **" "**+**" "**+employeeOne.**designation** +**" "**+ employeeOne.**designation**);

System.***out***.println(**"Data from copy constructor"**);

System.***out***.println(employeeTwo.**id**+**" "**+employeeTwo.**name** + **" "**+**" "**+employeeTwo.**designation** +**" "**+ employeeTwo.**designation**);

**try**

{

Q5Employee employeeThree = **new** Q5Employee(2, **"Vaishnavi"**, 12000.56, **"JVM"**);

Q5Employee employeeFour = (Q5Employee)employeeThree.clone();

System.***out***.println(employeeThree.**id**+**" "**+employeeThree.**name** + **" "**+**" "**+employeeThree.**designation** +**" "**+ employeeThree.**designation**);

System.***out***.println(**"Data from clonable interface"**);

System.***out***.println(employeeFour.**id**+**" "**+employeeFour.**name** + **" "**+**" "**+employeeFour.**designation** +**" "**+ employeeFour.**designation**);

}

**catch**(CloneNotSupportedException cns)

{

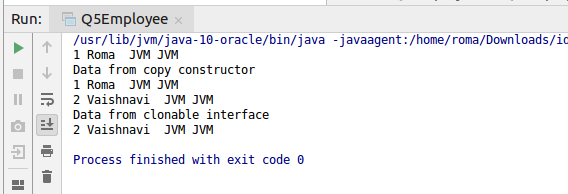
System.***out***.println(**"Exception caught"** +cns);

}

}

}

Output:



**Ans 6.**

/\*WAP showing try, multi-catch and finally blocks.

\*/

public class Q6ExceptionHandling {

public static void main(String[] args) {

String[] string = {"ok", "done","till", "now"};

int[] array ={35,78};

try

{

for(int i=0;i<string.length;i++)

{

int value = Integer.parseInt(string[i]);

}

}

catch(NumberFormatException nfe)

{

System.out.println("NumberFormatException caught");

}

try

{

System.out.println(array[8]);

}

catch(ArrayIndexOutOfBoundsException ae)

{

System.out.println("ArrayIndexOutOfBoundsException caught");

}

finally

{

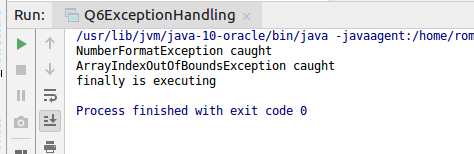
System.out.println("finally is executing");

}

}

}

Output:



**Ans 7.**

/\*WAP to convert seconds into days, hours, minutes and seconds.

\*/

**import** java.util.Scanner;

**public class** Q7Convert {

**public static void** main(String[] args) {

Scanner in = **new** Scanner(System.***in***);

System.***out***.print(**"Input seconds: "**);

**double** seconds = in.nextInt();

**double** mins= seconds/60;

**double** hrs = mins/60;

**int** days = (**int**)hrs/24;

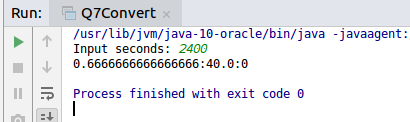
System.***out***.print( hrs + **":"** + mins + **":"** + days);

System.***out***.print(**"\n"**);

}

}

Output



**Ans 8.**

/\*WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use

a)while statement

b)do-while statement

\*/

a.)

**import** java.util.Scanner;

**public class** Q8Words {

**public static void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println(**"Enter a word"**);

String word = scanner.next();

**while** (!word.equals(**"done"**)) {

**if** (word.charAt(0) == word.charAt(word.length() - 1)) {

System.***out***.println(**"First and last character are equals for the word: "** + word);

} **else** {

System.***out***.println(**"First and last character are NOT equals for the word: "** + word);

}

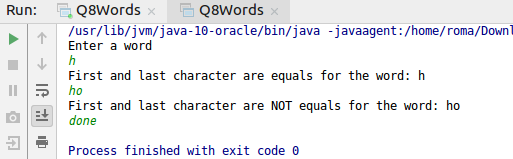
word = scanner.next();

}

}

}

Output:



b)

**import** java.util.Scanner;

**public class** Q8B {

**public static void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println(**"Enter a word"**);

String word = scanner.next();

**do**

{

**if**(word.charAt(0) == word.charAt(word.length() - 1))

{

System.***out***.println(**"First and last character are equals for the word: "** + word);

}

**else**

{

System.***out***.println(**"First and last character are NOT equals for the word: "** + word);

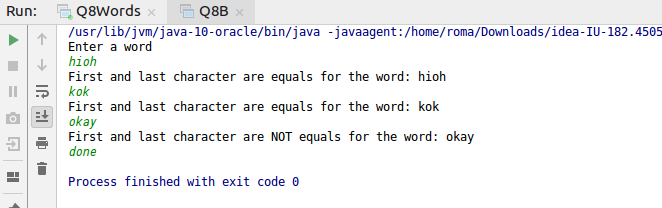
} word = scanner.next();

}**while**(!word.equals(**"done"**));

}

}

Output



**Ans 9.**

/\*Design classes having attributes for furniture where there are wooden chairs and tables, metal chairs and tables. There are stress and fire tests for each products.

\*/

**Skeleton:**

**public class** Q9 {

**public static void** main(String[] args) {

}

}

**abstract class** Wooden {

String **type**;

String **durability**;

**double price**;

String **finishing**;

}

**abstract class** Metal {

String **Type**;

**double price**;

String **finishing**;

}

**interface** stressTest

{

*//some formulae*

}

**interface** loadTest

{

*//some formulae*

}

**class** Chair **extends** Wooden

{

**private double height**;

**private double width**;

**private** String **color**;

**private** String **chairType**;

**private double makingCost**;

}

**class** Table **extends** Wooden

{

**private double height**;

**private double width**;

**private** String **color**;

**private** String **tableType**;

**private double cost**;

}

**class** ChairMetallic **extends** Metal

{

**private double height**;

**private double width**;

**private** String **color**;

**private** String **chairType**;

**private double cost**;

}

**class** TableMetallic **extends** Metal

{

**private double height**;

**private double width**;

**private** String **color**;

**private** String **chairType**;

**private double cost**;

}

**Ans 10.**

/\*Design classes having attributes and method(only skeleton) for a coffee shop.

\*/

**Skeleton:**

**import java.net.Inet4Address;**

**import java.util.Queue;**

**public class Q10Coffee {**

**public static void main(String[] args) {**

**}**

**}**

**class Customer**

**{**

**private String name;**

**private int orderId;**

**private double payment;**

**}**

**class Cashier**

**{**

**private Queue<Integer> orderQueue;**

**private double payemnt;**

**void insertInOrderQueue(int orderId)**

**{**

**orderQueue.add(orderId);**

**}**

**}**

**class Barista**

**{**

**Queue<Integer> completeOrder;**

**int currentOrder = completeOrder.peek();**

**void sendNotification(int currentOrder)**

**{**

**System.*out*.println("Order is completed");**

**}**

**}**

**Ans 11.**

int s = 0;

int t = 1;

int i = 0;

while(i < 10)

{

s = s + i;

int j = i;

while (j > 0)

{

t = t \* (j - i);

j--;

}

s = s\* t;

System.out.println("T is " + t);

i++;

}

System.out.println("S is " + s);

**Ans 12.**

/\*What will be the output on new Child();

\*/

**public class** Q12 {

**public static void** main(String[] args) {

Child child = **new** Child();

}

}

**class** Parent **extends** Grandparent {

{

System.***out***.println(**"instance - parent"**);

}

**public** Parent() {

System.***out***.println(**"constructor - parent"**);

}

**static** {

System.***out***.println(**"static - parent"**);

}

}

**class** Grandparent {

**static** {

System.***out***.println(**"static - grandparent"**);

}

{

System.***out***.println(**"instance - grandparent"**);

}

**public** Grandparent() {

System.***out***.println(**"constructor - grandparent"**);

}

}

**class** Child **extends** Parent {

**public** Child() {

System.***out***.println(**"constructor - child"**);

}

**static** {

System.***out***.println(**"static - child"**);

}

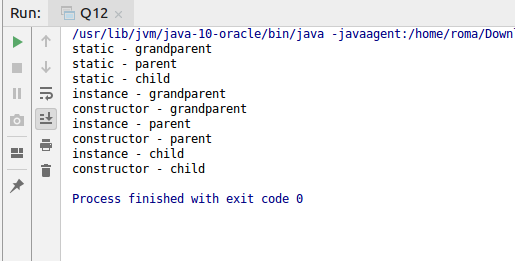
{

System.***out***.println(**"instance - child"**);

}

}

Output



**Ans 13.**

/\*Create a custom exception that do not have any stack trace.

\*/

**class** InvalidAgeException **extends** Exception{

InvalidAgeException(String s){

**super**(s);

}

}

**class** TestCustomException1{

**static void** validate(**int** age)**throws** InvalidAgeException{

**if**(age<18)

**throw new** InvalidAgeException(**"not valid"**);

**else**

System.***out***.println(**"welcome to vote"**);

}

**public void** printStackTrace() {

printStackTrace(System.***err***);

}

**public static void** main(String args[]){

**try**{

*validate*(13);

}**catch**(Exception m){

System.***out***.println(**"hjk"**);

m.printStackTrace();}

}

}