**General :**

1. **Use javadoc comment in a program to demonstrate it. Also explain javah and javap with example.**

**Code:**

Javadoc is a tool which comes with JDK and it is used for generating Java code documentation in HTML format from Java source code, which requires documentation in a predefined format.

Following is a simple example where the lines inside /\*….\*/ are Java multi-line comments. Similarly, the line which preceeds // is Java single-line comment.

Ex.

Code:

import java.io.\*;

/\*\*

\* <h2>Add Two Numbers!</h2>

\* The AddNum program implements an application that

\* simply adds two given integer numbers and Prints

\* the output on the screen.

\* <p>

\* <b>Note:</b> Giving proper comments in your program makes it more

\* user friendly and it is assumed as a high quality code.

\*

\* @author Jay Bhavsar

\* @version 1.0

\* @since 2020-01-20

\*/

public class AddNum {

/\*\*

\* This method is used to add two integers. This is

\* a the simplest form of a class method, just to

\* show the usage of various javadoc Tags.

\*

\* @param numA This is the first paramter to addNum method

\* @param numB This is the second parameter to addNum method

\* @return int This returns sum of numA and numB.

\*/

public int addNum(int numA, int numB) {

return numA + numB;

}

/\*\*

\* This is the main method which makes use of addNum method.

\*

\* @param args Unused.

\* @return Nothing.

\* @exception IOException On input error.

\* @see IOException

\*/

public static void main(String args[]) throws IOException {

AddNum obj = new AddNum();

int sum = obj.addNum(10, 20);

System.out.println("Sum of 10 and 20 is :" + sum);

}

}

**Output:**

Text

Description automatically generated

1. **Write a java program to explain dynamic method dispatch.**

Code: // A Java program to illustrate Dynamic Method

// Dispatch using hierarchical inheritance

class A

{

void m1()

{

System.out.println("Inside A's m1 method");

}

}

class B extends A

{

// overriding m1()

void m1()

{

System.out.println("Inside B's m1 method");

}

}

class C extends A

{

// overriding m1()

void m1()

{

System.out.println("Inside C's m1 method");

}

}

// Driver class

class Dispatch

{

public static void main(String args[])

{

// object of type A

A a = new A();

// object of type B

B b = new B();

// object of type C

C c = new C();

// obtain a reference of type A

A ref;

// ref refers to an A object

ref = a;

// calling A's version of m1()

ref.m1();

// now ref refers to a B object

ref = b;

// calling B's version of m1()

ref.m1();

// now ref refers to a C object

ref = c;

// calling C's version of m1()

ref.m1();

}

}

Output:

Text

Description automatically generated

**3. Explain access specifiers and modifiers in java with example.**

**4. Explain this() for constructor overloading and super() for superclass’s constructor calling with example.**

Code:Super()

class Parent {

Parent()

{

System.out.println("Parent Class Constructor Called...!!!");

}

}

class Child extends Parent {

Child()

{

super();

System.out.println("Flow comes back from " + "Parent class constructor to Child Constructor..!!");

}

public static void main(String[] args)

{

new Child();

System.out.println("Inside Main");

}

}

Output:

Text

Description automatically generated

This()

Code:

import java.util.\*;

public class Student{

int rollno;

String name,degree;

float fees;

Student(){ //non-param Constructor

System.out.println("Hello!!");

}

Student(int rollno,String name,String degree,float fees) //param constructor

{

this();

this.rollno=rollno;

this.name=name;

this.degree=degree;

this.fees=fees;

System.out.println(rollno+" "+name+" "+degree+" "+fees);

}

public static void main(String args[])

{

Student std=new Student(41,"Jay Bhavsar","MCA",40000);

}

}

Output:

Text

Description automatically generated

**5. Write program in java to explain deep and shallow cloning.**

**Shallow Copy**

Code:

import java.io.\*;

public class GFG {

public static void main(String[] args)

{

GFG t1 = new GFG();

GFG t2 = t1;

if(t1 == t2){

System.out.println("Shallow Copy");

}

else{

System.out.println("Deep Copy");

}

}

}

Output:

Text

Description automatically generated with medium confidence

**---------------------**

**Deep Copy:**

**---------------------**

public class GFGD implements Cloneable {

public Object clone() throws CloneNotSupportedException

{

return (GFGD)super.clone();

}

public static void main(String[] args) throws CloneNotSupportedException

{

GFGD t1 = new GFGD();

GFGD t2 = (GFGD)t1.clone();

if (t1==t2){

System.out.println("Shallow Clone");

}else{

System.out.println("Deep Clone");

}

}

}

**Output:**

**Text

Description automatically generated**

**6. Write programs to explain all types of nested and inner classes.**

**Member Inner Class:**

Code:

class OuterClass {

int x = 10;

class InnerClass {

int y = 5;

}

}

public class MemberInnerClass{

public static void main(String[] args) {

OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.y + myOuter.x);

}

}

Output:

Text

Description automatically generated

**Annonnomus class**

Code:

abstract class Person{

abstract void eat();

}

public class TestAnonymousInner{

public static void main(String args[]){

Person p=new Person(){

void eat(){System.out.println("nice fruits");

}

};

p.eat();

}

}

Output:

Text

Description automatically generated

**local inner class**

Code:

public class localInner1{

private int data=30;//instance variable

void display(){

class Local{

void msg(){System.out.println(data);}

}

Local l=new Local();

l.msg();

}

public static void main(String args[]){

localInner1 obj=new localInner1();

obj.display();

}

}

Output:

Text

Description automatically generated

**static nested class**

Code:

class OuterClass

{

static int outer\_x = 10;

int outer\_y = 20;

private static int outer\_private = 30;

static class StaticNestedClass

{

void display()

{

System.out.println("outer\_x = " + outer\_x);

System.out.println("outer\_private = " + outer\_private);

}

}

}

public class StaticNestedClassDemo

{

public static void main(String[] args)

{

OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

nestedObject.display();

}

}

**Output:**

Text

Description automatically generated

**7. Create class called sports\_accessories with attributes Accessory\_id, description, quantity, rate, used\_in\_game. Accept details of 10 accessories from user (5 records), store it in array of objects. Display details of all accessories used in game cricket.**

Code:

import java.util.Scanner;

import java.io.\*;

public class Sports\_accessories {

int Accessory\_id, quantity;

String description;

float rate;

String used\_in\_game;

public Sports\_accessories(int accessory\_id, int quantity, String description, float rate, String used\_in\_game) {

this.Accessory\_id = accessory\_id;

this.quantity = quantity;

this.description = description;

this.rate = rate;

this.used\_in\_game = used\_in\_game;

}

public static void main(String[] args) throws IOException {

Sports\_accessories [] sa\_array ;

sa\_array = new Sports\_accessories[3];

System.out.println("You can enter maximum 5 records");

Scanner sc = new Scanner(System.in);

BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

int j =0;

while (j<=2) {

System.out.println("Enter Id: \n");

int A\_id = Integer.parseInt(br.readLine());

System.out.println("Enter Quantity: \n");

int quant = Integer.parseInt(br.readLine());

System.out.println("Enter Accessories \n");

String desc =br.readLine();

System.out.println("Enter Rate :\n");

float rate = Float.parseFloat(br.readLine());

System.out.println("used in game:\n");

String used\_in\_game = br.readLine();

sa\_array[j]= new Sports\_accessories(A\_id, quant, desc, rate, used\_in\_game);

j++;

}

System.out.println("Showing Accesories Used In Sport Cricket");

for (int i=0; i < sa\_array.length; i++) {

if( sa\_array[i].used\_in\_game.equals("Cricket")){

System.out.println("Accessory\_id : " +sa\_array[i].Accessory\_id + "| quantity : " +sa\_array[i].quantity + "| accessories: " + sa\_array[i].description +"| rate : " + sa\_array[i].rate +"| used\_in\_game :" + sa\_array[i].used\_in\_game);

}

}

}

}

Output:

Text

Description automatically generated

Text

Description automatically generated

**8. Write a program to demonstrate Arrays class’s any 10 methods with example. (hint: java.util.Arrays)**

import java.util.Arrays;

// Main class

class arrayMethod {

// Main driver method

public static void main(String[] args)

{

// Get the Array

int intArr[] = { 10, 20, 15, 22, 45,1};

int intArr1[] = { 10, 20, 15, 22,45};

int intArr2[][] = { { 10, 20, 15, 22, 35 } };

int intArr3[][] = { { 10, 15, 22 } };

int intKey = 22;

String[] stringArray = { "a", "b", "c", "d", "e" };

Arrays.sort(intArr);

Arrays.sort(intArr1);

// true

// To convert the elements as List

System.out.println("Integer Array as List: "+ Arrays.asList(intArr));

System.out.println(intKey + " found at index = " + Arrays.binarySearch(intArr, intKey));

System.out.println(intKey+ " found at index = "+ Arrays.binarySearch(intArr, 1, 3, intKey));

System.out.println("Integer Array: "+ Arrays.toString(Arrays.copyOf(intArr, 10)));

System.out.println("Integer Array: "+ Arrays.toString(Arrays.copyOfRange(intArr, 1, 3)));

System.out.println("Integer Arrays on comparison: "+ Arrays.deepEquals(intArr2, intArr3));

System.out.println("Integer Array: "+ Arrays.deepHashCode(intArr2));

System.out.println("Integer Array: "+ Arrays.deepToString(intArr3));

System.out.println("The element mismatched at index: "

+ Arrays.mismatch(intArr, intArr1));

Arrays.parallelSort(intArr);

System.out.println("Integer Array: "

+ Arrays.toString(intArr));

}

}

Output:

Text

Description automatically generated

**Arrays**

**9). Define a class Student (name, roll\_no, class and marks of 6 subjects). Create an array of 5 Student objects. Calculate the percentage of each student using a method per(). Define a static method “sortStudent” which sorts the array based on percentage. Display the student details in sorted order.**

**Code:**

class Student{

String name;

int roll\_no;

String sclass;

int[] marks;

float percentage;

Student(String n, int r,String s, int[] m){

name =n;

roll\_no=r;

sclass=s;

marks=m;

}

void per(){

percentage = (marks[0]+marks[1]+marks[2]+marks[3]+marks[4]+marks[5])/6;

}

static void display(Student[] s){

System.out.println("Student Data Sorted In Ascending order");

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

System.out.println(s[i].name + " "+s[i].percentage);

}

}

static void sortStudent(Student[] s){

Student temp;

int n = s.length;

for(int i=0;i< n-1;i++){

for(int j=0 ; j < n-i-1;j++){

if(s[j].percentage > s[j+1].percentage){

temp = s[j];

s[j]=s[j+1];

s[j+1]=temp;

}

}

}

}

public static void main(String[] args){

Student[] s = new Student[6];

s[0] = new Student("Dino" ,3 , "MCA" , new int[]{40 ,41 ,42,43,44,45});

s[0].per();

s[1] = new Student("Emiway" ,4 , "MCA" , new int[]{30 ,31 ,32,33,34,35});

s[1].per();

s[2] = new Student("Divine" ,1 , "MCA" , new int[]{50 ,51 ,52,53,54,55});

s[2].per();

s[3] = new Student("Raftaar" ,2 , "MCA" , new int[]{10 ,11 ,12,13,14,15});

s[3].per();

s[4] = new Student("KamBhari" ,5 , "MCA" , new int[]{20 ,21 ,22,23,24,25});

s[4].per();

s[5] = new Student("JB" ,6 , "MCA" , new int[]{20 ,21 ,22,23,24,25});

s[5].per();

Student.sortStudent(s);

Student.display(s);

}

}

**Output:**

**Text

Description automatically generated**

**9. Define a class Student (name, roll\_no, class and marks of 6 subjects). Create an array of 5 Student objects. Calculate the percentage of each student using a method per(). Define a static method “sortStudent” which sorts the array based on percentage. Display the student details in sorted order.**

Code:

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

class Student

{

int rollno;

String name;

float per;

static int count;

Student(){}

Student(String n,float p)

{

count++;

rollno=count;

name=n;

per=p;

}

void display2()

{

System.out.println("Rollno \t\t Name \t\t Percntag");

}

void display()

{

System.out.println(rollno+"\t\t"+name+"\t\t"+per);

}

float getper()throws IOException

{/\*

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int a[]=new int[6];

for(int i=0;i<6;i++)

{

a[i]=Integer.parseInt(br.readLine());

}

int t=0;

for(int i=0;i<6;i++)

{

t=t+a[i];

}

System.out.print("\n percentage:"+t);

float per;

per=(t/(float)6);\*/

return per;

}

static void counter()

{

System.out.println("ln Object is created "+count);

}

public static void sortStudent(Student s[],int n)throws IOException

{

for(int i=n-1;i>=0;i--)

{

for(int j=0;j<i;j++)

{

if(s[j].getper()>s[j+1].getper())

{

Student t=s[j];

s[j]=s[j+1];

s[j+1]=t;

}

}

}

for(int i=0;i<n;i++)

s[i].display();

}

}

class Studentclass

{

public static void main(String args[]) throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter no. of Student:");

int n=Integer.parseInt(br.readLine());

Student p[]=new Student[n];

for(int i=0;i<n;i++)

{

System.out.print("\n Enter Your Name:");

String name=br.readLine();

System.out.print("\n Enter percentage:");

float per=Float.parseFloat(br.readLine());

p[i]=new Student(name,per);

p[i].counter();

}

Student.sortStudent(p,Student.count);

}

}

Output:

Text

Description automatically generated

**10.Define a class Staff with members’ id, name, DOB, joining\_date and salary. Define class TeachingStaff with subjects[], experience and extends Staff. Also define class NonTeachingStaff with department, shift and extends Staff. Now define array DeptStaff with members from TeachingStaff and NonTeachingStaff as per requirement. Display details of all objects.**

Code:

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

class Staff

{

int id;

double salary;

String name,joining\_date,DOB;

}

class TeachingStaff extends Staff

{

String Subjects;

int experience;

//DeptStaff array=new DeptStaff[] ;

public void accept()throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the ID :");

id=Integer.parseInt(br.readLine());

System.out.println("Enter the Name :");

name=br.readLine();

System.out.println("Enter the Date Of Birth : ");

DOB=br.readLine();

System.out.println("Enter the Date Of Joining : ");

joining\_date=br.readLine();

System.out.println("Enter the Salary: ");

salary=Double.parseDouble(br.readLine());

System.out.println("Enter the Subjcts: ");

//Subjects[] ss=new Subjects[3];

Subjects=br.readLine();

System.out.println("Enter the Experience: ");

experience=Integer.parseInt(br.readLine());

}

public void display()

{

System.out.println("\n ---------- Teaching Staff ------------");

System.out.println("ID: "+id);

System.out.println("Name: "+name);

System.out.println("Date Of Birth: "+DOB);

System.out.println("Date Of Joining: "+ joining\_date);

System.out.println("Salary: "+salary);

System.out.println("Subjcts: "+Subjects);

System.out.println("Experience: "+experience);

}

}

class NonTeachingStaff extends Staff

{

String department,shift;

public void accept() throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the Department : ");

department=br.readLine();

System.out.println("Enter the Shift : ");

shift=br.readLine();

}

public void display()

{

System.out.println("\n ----------Non Teaching Staff------------");

System.out.println("Department: "+department);

System.out.println("Shift: "+shift);

}

}

public class Q10

{

public static void main(String [] args)throws IOException

{

/\*TeachingStaff s=new TeachingStaff();

s.accept();

s.display();

NonTeachingStaff n=new NonTeachingStaff();

n.accept();

n.display();\*/

int i;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Select Any One: ");

System.out.println("1.TeachingStaff");

System.out.println("2.NonTeachingStaff");

int ch=Integer.parseInt(br.readLine());

switch(ch){

case 1:

System.out.println("Enter the number of TeachingStaff: ");

int n=Integer.parseInt(br.readLine());

TeachingStaff[] l=new TeachingStaff[n];

for(i=0;i<n;i++){

l[i]=new TeachingStaff();

l[i].accept();

}

for(i=0;i<n;i++){

l[i].display();

}

break;

case 2:

System.out.println("Enter the number of NonTeachingStaff: ");

int m=Integer.parseInt(br.readLine());

NonTeachingStaff[] h=new NonTeachingStaff[m];

for(i=0;i<m;i++){

h[i]=new NonTeachingStaff();

h[i].accept();

}

for(i=0;i<m;i++){

h[i].display();

}

break;

}

}

}

Output:

A computer screen capture

Description automatically generated with medium confidence

Text

Description automatically generated

**11.Define Employee class (name, designation, salary). Define a default and parameterized constructor. Override the toString method. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.**

Code:

Code:

class Employee{

String name;

String designation;

float salary;

static int count;

Employee(){

name = "jay";

designation = "Manager";

salary = 18000.500f;

count++;

}

Employee(String nm , String des , float sal){

name = nm;

designation = des;

salary =sal;

count++;

}

Employee(String nm , String des){

name = nm;

designation = des;

salary =10000.0f;

count++;

}

public String toString(){

return (name + " " + designation + " " + salary);

}

static void count\_object\_method(){

System.out.println("Object Created : "+count);

}

public static void main(String args[]){

Employee obj = new Employee();

System.out.println(obj);

Employee.count\_object\_method();

Employee obj2 = new Employee("Jack","Business Anaylst",67.500f);

System.out.println(obj2);

Employee.count\_object\_method();

Employee obj3 = new Employee("Shreya","DBA");

System.out.println(obj3);

Employee.count\_object\_method();

}

}

Output:

Text

Description automatically generated

**Use of Abstract Interface**

**12.Create classes Food and Drink that extend the Product class, add subclass specific attributes and methods, override parent class methods and change the Product class from concrete to abstract. Create class ProductManager and add factory methods to it to create instances of Food and Drink. The Shop class is then modified to use these factory methods instead of using constructors of Food and Drink directly.**

**Code:**

abstract class Product

{

int productCode;

float productRate;

String productName;

String productType;

void setProductCode(int productCode)

{

this.productCode=productCode;

}

int getProductCode()

{

return productCode;

}

void setProductType(String productType)

{

this.productType=productType;

}

String getProductType()

{

return productType;

}

void setProductName(String productName)

{

this.productName=productName;

}

String getProductName()

{

return productName;

}

}

class Food extends Product

{

String foodBrand;

String foodCategory;

void prepareFood()

{

System.out.println("Types of Foods");

}

}

class Drink extends Product

{

String drinkType;

void prepareDrink()

{

System.out.println("Types of Drinks");

}

}

class ProductManage

{

Food getObjFood()

{

return new Food();

}

Drink getObjDrink()

{

return new Drink();

}

}

interface Rateable

{

void findRating(int r, Product p);

}

class Review implements Rateable

{

String review;

public void findRating(int r, Product p)

{

System.out.println("customer has rated " + p.getProductName() + " with "+ r + " star");

}

void printReview(String review, Product p)

{

System.out.println("customer gives Feedback " + p.getProductName() +" "+ review);

}

}

public class foodFactory

{

public static void main(String[] args)

{

ProductManage p=new ProductManage();

Food food=p.getObjFood();

Drink drink=p.getObjDrink();

food.setProductCode(1);

food.setProductType("Gravy");

food.setProductName("Chicken Tikka");

food.prepareFood();

drink.setProductCode(2);

drink.setProductType("Drinks");

drink.setProductName("Milk-Shake");

drink.prepareDrink();

Review r=new Review();

r.findRating(4,food);

r.printReview("is tasty", food);

r.findRating(10,drink);

r.printReview("Quality is so good", drink);

}

}

**Output:**

**Text

Description automatically generated**

**14.Create a package named com Define subpackages; a. transact: with class Transaction with static methods credit() and debit() b. loan: with class LoanAccount with method doTransaction() which calls Transaction class mehods. Create one LoanAccount object in main to perform operations on it by accepting command line arguments.**

**Code:**

**com**

package com.loan;

public class LoanAccount

{

public void doTransaction(float cAmt, float wAmt)

{

com.transact.Transaction.credit(cAmt);

com.transact.Transaction.debit(wAmt);

}

}

----------

package com.transact;

public class Transaction

{

public static void credit(float cAmt)

{

System.out.println("you have credited " + cAmt + " amount");

}

public static void debit(float wAmt)

{

System.out.println("you have debited "+ wAmt + " amount");

}

}

--------------

import com.loan.LoanAccount;

class GetLoan

{

public static void main(String a[])

{

float cAmount=Float.parseFloat(a[0]);

float wAmount=Float.parseFloat(a[1]);

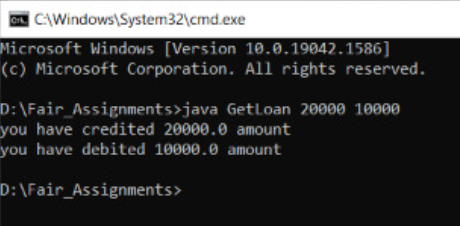
LoanAccount la=new LoanAccount();

la.doTransaction(cAmount,wAmount);

}

}

**Output:**

****

**Exception handling:**

**16.Write a program to accept senior citizens name and age from command prompt. If age is below 60, throw “InvalidAgeException” exception.**

**Code:**

import java.io.\*;

import java.io.IOException;

public class ExceptionHandling\_age extends Exception {

public static void main(String[] args) throws IOException {

try{

String name = args[0];

int age = Integer.parseInt(args[1]);

System.out.println("Name: "+name+" Age: "+age);

if(age < 60)

{

throw new InvalidAgeException();

}

}catch (InvalidAgeException e) {

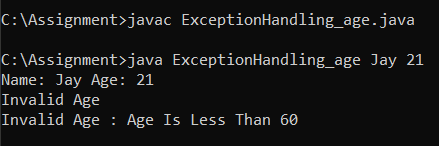
System.out.println("Invalid Age : Age Is Less Than 60");

}

}

}

**Output:**



**17.Write an application to define a user defined exception “InsufficientFundException”. Read the amount from console and if amount is available in your account, then draw the amount. If amount is not available, throw “InsufficientFundException” and display amount available for withdrawal.**

**Code:**

import java.io.\*;

import java.io.IOException;

public class AmountException extends Exception

{

public static void main(String args[])throws IOException

{

try

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Account Balance is 40000 ");

int a=40000;

System.out.println("Enter the Amount :");

int amto = Integer.parseInt(br.readLine());

if(amto>=a)

{

throw new InsufficientFundException();

}

System.out.println("Amount Sucfficient");

System.out.println("Withdrawn Amount is "+amto);

System.out.println("Balance : "+(a-amto));

}

catch(InsufficientFundException n)

{

System.out.println("Invalid Amount Deposite Amount Grater Than Balance");

}

}

}

**Output:**

**Text

Description automatically generated**

**Packages: Multithreading**

**18.Create thread to display prime numbers between 1 to 500 after every 3 second.**

**Code:**

import java.util.Scanner;

// thread to print prime numbers

class prime\_thread extends Thread {

public synchronized void run()

{

int i = 0;

int num = 0;

String primeNumbers = "";

for (i = 1; i <= 500; i++) {

int counter = 0;

for (num = i; num >= 1; num--) {

// condition to check if the number is prime

if (i % num == 0) {

// increment counter

counter = counter + 1;

}

}

if (counter == 2) {

primeNumbers = primeNumbers + i + " ";

}

}

System.out.println("\nPrime numbers from 0 to 100 : \n"

+ primeNumbers);

System.out.println();

}

}

public class PrimeNoMultithreading {

public static void main(String args[])

{

prime\_thread t1 = new prime\_thread();

Thread m1 = new Thread(t1);

Scanner sc = new Scanner(System.in);

// start() method starts the execution of thread.

m1.start();

try {

// join() method waits for the thread to die

m1.join();

Thread.sleep(3);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**Output:**

**Graphical user interface

Description automatically generated with low confidence**

**19.Write a thread program to display Perfect numbers between one to 1000 after every 5 seconds. e. g. 6 = 3 + 2 + 1 28 = 14+7+4+2+1**

**Code:**

import java.util.Scanner;

// thread to print prime numbers

class prime\_thread extends Thread {

public synchronized void run()

{

int i = 0;

int num = 0;

String primeNumbers = "";

for (i = 1; i <= 500; i++) {

int counter = 0;

for (num = i; num >= 1; num--) {

// condition to check if the number is prime

if(i % num == 0) {

// increment counter

counter = counter + 1;

}

}

if (counter == 2) {

primeNumbers = primeNumbers + i + " ";

}

}

System.out.println("\nPrime numbers from 0 to 100 : \n"

+ primeNumbers);

System.out.println();

}

}

public class PrimeNoMultithreading {

public static void main(String args[])

{

prime\_thread t1 = new prime\_thread();

Thread m1 = new Thread(t1);

Scanner sc = new Scanner(System.in);

// start() method starts the execution of thread.

m1.start();

try {

// join() method waits for the thread to die

m1.join();

Thread.sleep(3);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**Output:**

**Text

Description automatically generated**

**20.Execution of two or more threads occurs in a random order. The keyword 'synchronized' in Java is used to control the execution of thread in a strict sequence. In the following, the program is expected to print some numbers. Do the necessary use of 'synchronized' keyword, so that, the program prints the output in the following order: -----------------OUTPUT------------------- 5 10 15 20 25 100 200 300 400 500**

**Code:**

class Execute{

synchronized void print(int n){

for(int i=1;i<=5;i++){

System.out.println(n\*i);

try{

Thread.sleep(400);

}catch(Exception e){

System.out.println(e);

}

}

}

}// Ending Execute class

class Thread1 extends Thread{

Execute t;

Thread1(Execute t){

this.t=t;

}

public void run(){

t.print(5);

}

}

class Thread2 extends Thread{

Execute t;

Thread2(Execute t){

this.t=t;

}

public void run(){

t.print(100);

}

}

public class Question30{

public static void main(String args[]){

Execute obj = new Execute();//only one object

Thread1 t1=new Thread1(obj);

Thread2 t2=new Thread2(obj);

t1.start();

t2.start();

}

}

**Output:**

**Text

Description automatically generated**

**Useful Classes**

**21.Write a java Program using Regular Expression to check password which should have at least one Capital letter, 1 digit, 1 special character and length should be more than 6 and less than or equal to 8.**

**Code:**

// Java program to validate

// the password using ReGex

import java.util.regex.\*;

class RegExp{

// Function to validate the password.

public static boolean

isValidPassword(String password)

{

// Regex to check valid password.

String regex = "^(?=.\*[0-7])"

+ "(?=.\*[a-z])(?=.\*[A-Z])"

+ "(?=.\*[@#$%^&+=])"

+ "(?=\\S+$).{8,20}$";

// Compile the ReGex

Pattern p = Pattern.compile(regex);

// If the password is empty

// return false

if (password == null) {

return false;

}

// Pattern class contains matcher() method

// to find matching between given password

// and regular expression.

Matcher m = p.matcher(password);

// Return if the password

// matched the ReGex

return m.matches();

}

// Driver Code.

public static void main(String args[])

{

// Test Case 1:

String str1 = "jay@55";

System.out.println(str1+":"+isValidPassword(str1));

// Test Case 2:

String str2 = "jay55";

System.out.println(str2+":"+isValidPassword(str2));

// Test Case 3:

String str3 = "jay@ jan25";

System.out.println(str3+":"+isValidPassword(str3));

// Test Case 4:

String str4 = "252001";

System.out.println(str4+":"+isValidPassword(str3));

// Test Case 5:

String str5 = "JB@25";

System.out.println(str5+":"+isValidPassword(str5));

// Test Case 6:

String str6 = "JB@jan25";

System.out.println(str6+":"+isValidPassword(str6));

}

}

**Output:**

**Text

Description automatically generated**

**22.Write a program using Observable and Observer Class.**

**Code:**

import java.util.\*;

class Organizer implements Observer

{

public void update(Observable a, Object obj)

{

System.out.println("Attendees are now :" + obj.toString());

}

}

class Attendee extends Observable

{

void attend(int count)

{

for(int i=count;i>=0;i--)

{

setChanged();

notifyObservers(String.valueOf(i));

}

}

}

class PIBMEvent

{

public static void main(String a[])

{

Organizer o=new Organizer();

Attendee attendee=new Attendee();

attendee.addObserver(o);

attendee.attend(10);

}

}

**Output:**

**Text, table

Description automatically generated**

**AWT and Swing:**

**27.Display text on label “core Java Programming” in larger size with bold and Times new roman font, take three buttons with captions red, green and blue. On Click event of each button respective color should change for the background of label (Use swing).**

**Code:**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

class SwingExample

{

public static void main(String[] args) {

JFrame f=new JFrame("SwingExample");

JLabel l1=new JLabel("Core Java Programming:");

//l1.setFont(new Font("Serif", Font.BOLD, 20));

l1.setBounds(30,5,150,50);

//l1.setBackground(Color.red);

l1.setOpaque(true);

l1.setFont(new Font("TimesNewRoman",Font.BOLD,10));

JButton btnred=new JButton("Red");

btnred.setBounds(35,150,90,30);

JButton btngreen=new JButton("green");

btngreen.setBounds(145,150,90,30);

JButton btnblue=new JButton("blue");

btnblue.setBounds(245,150,90,30);

btnred.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

l1.setBackground(Color.red);

}

});

btngreen.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

l1.setBackground(Color.green);

}

});

btnblue.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

l1.setBackground(Color.blue);

}

});

f.add(l1);

f.add(btnred);

f.add(btngreen);

f.add(btnblue);

f.setSize(400,400);

//f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setLayout(null);

f.setResizable(false);

f.setVisible(true);

}

}

**Output:**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**JDBC**

**30.write JDBC application to register students for convocation in MCA (Use PreparedStatement, assume suitable table structure).**

Code:

StudConvo.html

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Student Convocation!!!</title>

</head>

<body>

<h1 style="text-align:center;">Student Registration for Convocation!!</h1><br>

<fieldset style="text-align:center;">

<form action="StudentRegConvo" method="post">

Enter Name:<input type="text" name="name"><br><br>

Enter Email\_ID:<input type="text" name="emailid"><br><br>

Enter Mobile Number:<input type="text" name="mobileno"><br><br>

Enter Roll Number:<input type="text" name="rollno"><br><br>

Select Degree:<select name="studDegree" id="StudDegree">

<option value="MCA">MCA</option>

<option value="MBA">MBA</option>

</select>

<br><br>

Any improvement/Any Suggestion:

s<br><textarea name="comments"></textarea><br><br>

<input type="reset" value="Reset">

<input type="Submit" value="Submit">

</form>

</fieldset>

</body>

</html>

StudentRegConvo.java

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class StudentRegConvo

\*/

@WebServlet("/StudentRegConvo")

public class StudentRegConvo extends HttpServlet {

private static final long serialVersionUID = 1L;

public StudentRegConvo() {

super();

// TODO Auto-generated constructor stub

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String name=request.getParameter("name");

String emailid=request.getParameter("emailid");

String mobileno=request.getParameter("mobileno");

String roll\_no=request.getParameter("rollno");

String degree=request.getParameter("studDegree");

String comments=request.getParameter("comments");

try {

Class.forName("com.mysql.jdbc.Driver");

java.sql.Connection con =DriverManager.getConnection("jdbc:mysql://localhost:3306/stud","root","");

PreparedStatement pst=con.prepareStatement("insert into stdconvo values(?,?,?,?,?,?)");

pst.setString(1,name);

pst.setString(2,emailid);

pst.setString(3,mobileno);

pst.setString(4,roll\_no);

if(degree.equals("MCA"))

{

pst.setString(5,"MCA");

}

if(degree.equals("MBA"))

{

pst.setString(5,"MBA");

}

else if(degree.equals(" "))

{

out.println("<h3>Please Select Degree!!</h3>");

}

pst.setString(6,comments);

int row = pst.executeUpdate();

}

catch(Exception e)

{

out.println(e);

}

out.println("<h1>Data Successfully Added!!</h1>");

}

}

Output:

Text, timeline

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**Servlet and JSP & JDBC:**

**31.Write servlet program to accept student details to store in database and display in tabular format. Assume suitable table structure (Write HTML interface, web.xml & servlet class) Validation parameters: a. All fields are compulsory. b. Book number should be numeric c. Book name should not contain special symbol d. Price should be decimal with two precisions e. Validate the data by using a servlet**

Code:

Output;

**32.Write a JSP program to accept patient details from HTML and display patient details in proper format to update current details. If details not found display error page.**

Code:

Output:

**33.Write JSP code to accept registration details for placement portal from candidate and insert the details in database (assume suitable table structure).**

Code:

Register.html

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<h1 style="text-align:center;">Register Form</h1>

<fieldset>

<form action="register.jsp" >

<table >

<tr>

<td>User Name:</td>

<td><input type="text" name="username" /></td>

</tr>

<tr>

<td>Email ID:</td>

<td><input type="text" name="emailid" /></td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="password" /></td>

</tr>

<tr>

<td>Contact\_no</td>

<td><input type="text" name="contactno" /></td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="address"></textarea></td>

</tr>

<tr>

<td>Degree</td>

<td><input type="text" name="degree"/></td>

</tr>

<tr>

<td>Skills</td>

<td><textarea name="skills"></textarea></td>

</tr></table>

<input type="submit" value="Submit" />

</form>

</fieldset>

</body>

</html>

Register\_ui.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@page import="java.sql.\*,java.util.\*"%>

<%

String username=request.getParameter("username");

String emailid=request.getParameter("emailid");

String pass=request.getParameter("password");

String contact\_no=request.getParameter("contactno");

String address=request.getParameter("address");

String degree=request.getParameter("degree");

String skills=request.getParameter("skills");

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspprg", "root", "");

PreparedStatement pst=con.prepareStatement("insert into placement values(?,?,?,?,?,?,?)");

pst.setString(1,username);

pst.setString(2, emailid);

pst.setString(3, pass);

pst.setString(4, contact\_no);

pst.setString(5, address);

pst.setString(6, degree);

pst.setString(7, skills);

int row = pst.executeUpdate();

out.println("Data is successfully inserted!");

}

catch(Exception e)

{

e.printStackTrace();

}

%>

Output:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

**34.Write JSP application to insert record into student table & display record in tabular format assume suitable table structure. [Write HTML, web.xml]**

Code:

Home.html

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<form method="post" action="process.jsp">

First name:<br>

<input type="text" name="first\_name">

<br>

Last name:<br>

<input type="text" name="last\_name">

<br>

City name:<br>

<input type="text" name="city\_name">

<br>

Email Id:<br>

<input type="email" name="email">

<br><br>

<input type="submit" value="submit">

</form>

</body>

</html>

Process.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@page import="java.sql.\*,java.util.\*"%>

<%

String first\_name=request.getParameter("first\_name");

String last\_name=request.getParameter("last\_name");

String city\_name=request.getParameter("city\_name");

String email=request.getParameter("email");

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspprg", "root", "");

Statement st=conn.createStatement();

int i=st.executeUpdate("insert into users(first\_name,last\_name,city\_name,email)values('"+first\_name+"','"+last\_name+"','"+city\_name+"','"+email+"')");

out.println("Data is successfully inserted!");

}

catch(Exception e)

{

e.printStackTrace();

}

%>

Output:

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated