**Write a program to generate the following output in java?**

**\***

**\*\***

**\*\*\*\***

**\*\*\*\*\***

**\*\*\*\*\*\***

**Ans:**

import java.util.Scanner;

class Star{

public static void main(String[]args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

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

for(int j=1;j<=1;j++){

System.out.println(“\*”);

}

System.out.println( );

}

}

}

**Write a Java program to show a NullPointerException**

Class Null{

public static void main(String[]args){

int[]a=null;

a[0]=10;

System.out.println(a[0]);

}

}

**Constructor Overloading**

Constructor overloading is enables a single class to have more than one constructor that varies by the list of arguments passed.

public class Student {

int id,passoutYear;

String name,contactNo,collegeName;

Student(String contactNo, String collegeName, int passoutYear){

this.contactNo = contactNo;

this.collegeName = collegeName;

this.passoutYear = passoutYear;

}

Student(int id, String name){

this("987654321", "SRM", 2021);

this.id = id;

this.name = name;

}

public static void main(String[] args) {

Student s = new Student(420, "Sara");

System.out.println("Printing Student Information: \n");

System.out.println("Name: "+s.name+"\nId: "+s.id+"\nContact No.: "+s.contactNo+"\nCollege Name: "+s.contactNo+"\nPassing Year: "+s.passoutYear);

}

}

**Find duplicate elements in a string**

import java.util.HashMap;

public static void findDuplicates(String s) {

HashMap<Character, Integer> map = new HashMap<Character, Integer>();

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

char c = s.charAt(i);

if (map.containsKey(c)) {

map.put(c, map.get(c) + 1);

} else {

map.put(c, 1);

}

}

System.out.print("Duplicate characters in string: ");

for (char c : map.keySet()) {

if (map.get(c) > 1) {

System.out.print(c + " ");

}

}

}

**Implement multiple inheritances using an interface:-**

While achieving multiple inheritance using interface we can convert one parent type reference to another parent type with the help of cast operated.

Example for:

Interface I1

{

Void test1();

}

Interface I2

{

Void test2();

}

Class C implements I1,I2

{

Public void test1()

{

System.out.println(“hi”);

}

Public void test2()

{

System.out.println(“bye”);

}}

Class DriverMain{

Public static void main (String[] args){

C obj1.test1();

Obj1.test1();

Obj1.test2();

I1 obj2=obj1;//upcasting

Obj2.test1();

I2 obj3=obj1;

Obj3.test2();

I1 obj4=(I1)obj3;//I2 type to I1 type

Obj4.test1();

I2 obj5=(I2)obj4;//I1 type to I2 type

Obj5.test2();

}}

**Implement a program for encapsulation:-**

Class Student

{

String name ;

Int id;

Int age;

Public void Studying()

{

System.out.println(name+”is Studying”);

}

Student(String name,int id,int age)

{

This.name=name;

This.id=id;

This.age=age;

}}

Class Driver1

{

public static void main(String[] args)

{

Student S1=new Student(“sam”,100,21);

Student S2=new Student(“sai”,101,22);

Student S3=new Student(“sathish”,103,23);

DisplayDetails(S1);

DisplayDetails(S2);

DisplayDetails(S3);

}

Public static void displayDetails(Student S)

{

System.out.println(“name:”+S.name);

System.out.println(“id:”+S.id);

System.out.println(“age:”+S.age);

S.Studying();

System.out.println(“\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”);

}}

**Implement a program for abstraction:-**

Abstract class A

{

Abstract void test1()

}

Abstract class B extends A

{

Void test2()

{

System.out.println(“hi”);

}}

Class C extends B

{

Public void test1()

{

System.out.println(“hello”);

}}

Class Driver1

{

Public static void main(String[] args)

{

C obj1=newC();

Obj1.test1();

Obj2.test2();

B obj2=obj1;

Obj2 .test1();

Obj2.test2();

A obj3=obj1;

Obj3.test1();

Obj3.test2();

}}

**Print Multiplication table Program in java:-**

import java.util.Scanner;

public class MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

for (int i = 1; i <= 10; i++) {

int result = number \* i;

System.out.println(number + " x " + i + " = " + result);

}

scanner.close();

}

}

**How do you write an interface with default and static method**

public interface MyInterface {

void myMethod();

default void myDefaultMethod() {

System.out.println("This is a default method.");

}

static void myStaticMethod() {

System.out.println("This is a static method.");

}

}

public class MyClass implements MyInterface {

public void myMethod() {

System.out.println("MyClass implementation of myMethod.");

}

public void myDefaultMethod() {

System.out.println("MyClass implementation of myDefaultMethod.");

}

}

public class Main {

public static void main(String[] args) {

MyInterface.myStaticMethod();

MyClass obj = new MyClass();

obj.myMethod();

obj.myDefaultMethod();

}

}

**Write a Java program to create and throw custom exceptions?**

Class TestException extends RuntimeException

{

}

class A

{

Public static void Main(String[] args)

{

throw new TestException ();

}

}

**Write a program to demonstrate method overriding?**

Class Card

{

Public void makepayment(){

System.out.println(“payment done”);

}

}

Class Debitcard extends card

{

Public void makepayment(){

{

System.out.println(“payment done using debit card”);

}

}

Class Creditcard extends card

{

Public void makepayment(){

System.out.println(“payment done using Creditcard”);

}

}

Class swipMachine

{

Public void swipcard(card c){

c.makepayment();

}

}

Class Driver

{

Public static void main(String[] args){

Debitcard c1=new Debitcard();

Creditcard c2=new Creditcard();

swipMachine m=new swipMachine();

m. swipMachine(c1); //payment done using debitcard

m. swipMachine(c2);// payment done using creditcard

}

}

**Write a Java program that sorts HashMap by value.?**

Import java.util.HashMap;

Import java.util.set;

Class c

{

Public static void main(String[] args)

{

HashMap h1=new HashMap();

h1.put(1,”Naveen”);

h1.put(2,”rakesh”);

h1.put(3,”Pramod”);

set s=h1.keyset();

for(object i:s)

{

System.out.println(i);

}

}

}

**Write a Java Program for Fibonacci series?**

Class Fibonacci

{

Public static void power(int x)

{

int f1=0;

int f2=1;

int f3=0;

while(f1<=x)

system.out.println(f1);

f3=f1+f2;

f1=f2;

f2=f3;

}

}

Public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println(“Enter value”);

int a=sc.nextInt();

power(a);

}

}

}

**How to check Odd and Even Number in java.**

Class L

{

Public static void main(String[] args)

{

int i=1;

while(i<=10)

{

If(i%2==0)

{

System.out.println(i+”is an even number”);

}

else

{

System.out.println(i+”is an even number”);

}

i++

}

}

}

**Check no is Armstrong or not in java Program?**

Class R

{

Public static int countDigit(int n)

{

int count=0;

while(n>0)

{

n=n/10;

count++;

}

return count;

}

Public static void main(String[] args)

{

int num=375;

int num1=num;

int count=countDigit(num);

int ld;

int p;

int sum=0;

while(num>0)

{

ld=num%10;

p=power(ld,count);

sum=sum+p;

num=num/10;

}

If(num1==sum)

{

System.out.println(“Armstrong number”);

}

else

{

System.out.println(“ Not an Armstrong number”);

}

}

Public static int power(int x,int n)

{

int p=1;

for(int i=1;i<n;i++)

{

P=p\*x;

}

return p;

}