// 1.simple class

public class Main {

int x = 5;

public static void main(String[] args) {

Main myObj = new Main();

System.out.println(myObj.x);

}

}

// 2.member variable and member function

import java.io.\*;

public class Employee {

public String name;

private double salary;

public Employee (String empName) {

name = empName;

}

public void setSalary(double empSal) {

salary = empSal;

}

public void printEmp() {

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

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

}

public static void main(String args[]) {

Employee empOne = new Employee("Rajat");

empOne.setSalary(82000);

empOne.printEmp();

}

}

// 3.enum in java

public class Main {

enum Level {

LOW,

MEDIUM,

HIGH

}

public static void main(String[] args) {

Level myVar = Level.MEDIUM;

System.out.println(myVar);

}

}

// 4.single inheritance

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class TestInheritance{

public static void main(String args[]){

Dog d=new Dog();

d.bark();

d.eat();

}}

// 5.multilevel inheritance

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class BabyDog extends Dog{

void weep(){System.out.println("weeping...");}

}

class TestInheritance2{

public static void main(String args[]){

BabyDog d=new BabyDog();

d.weep();

d.bark();

d.eat();

}}

//6.hierarchical inheritance

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class Cat extends Animal{

void meow(){System.out.println("meowing...");}

}

class TestInheritance3{

public static void main(String args[]){

Cat c=new Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}}

// 7.Java Program to create and call a default constructor

class Bike1{

//creating a default constructor

Bike1(){System.out.println("Bike is created");}

//main method

public static void main(String args[]){

//calling a default constructor

Bike1 b=new Bike1();

}

}

//8.Let us see another example of default constructor

//which displays the default values

class Student3{

int id;

String name;

//method to display the value of id and name

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){

//creating objects

Student3 s1=new Student3();

Student3 s2=new Student3();

//displaying values of the object

s1.display();

s2.display();

}

}

// 9.Java Program to demonstrate the use of the parameterized

constructor.

class Student4{

int id;

String name;

//creating a parameterized constructor

Student4(int i,String n){

id = i;

name = n;

}

//method to display the values

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){

//creating objects and passing values

Student4 s1 = new Student4(111,"Karan");

Student4 s2 = new Student4(222,"Aryan");

//calling method to display the values of object

s1.display();

s2.display();

}

}

// 10.java destructor

public class DestructorExample

{

public static void main(String[] args)

{

DestructorExample de = new DestructorExample ();

de.finalize();

de = null;

System.gc();

System.out.println("Inside the main() method");

}

protected void finalize()

{

System.out.println("Object is destroyed by the Garbage Collector");

}

}

// 11.run time polymorphism in java

class Bike{

void run(){System.out.println("running");}

}

class Splendor extends Bike{

void run(){System.out.println("running safely with 60km");}

public static void main(String args[]){

Bike b = new Splendor();//upcasting

b.run();

}

}

// 12.operator overloading

class OverloadingExample{

static int add(int a,int b){return a+b;}

static int add(int a,int b,int c){return a+b+c;}

}

// 13.function overriding

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void eat(){System.out.println("eating bread...");}

}

// 24.friend function in java

public class A {

private int privateInt = 31415;

public class SomePrivateMethods {

public int getSomethingPrivate() { return privateInt; }

private SomePrivateMethods() { } // no public constructor

}

public void giveKeyTo(B other) {

other.receiveKey(new SomePrivateMethods());

}

}

public class B {

private A.SomePrivateMethods key;

public void receiveKey(A.SomePrivateMethods key) {

this.key = key;

}

public void usageExample() {

A anA = new A();

// int foo = anA.privateInt; // doesn't work, not accessible

anA.giveKeyTo(this);

int fii = key.getSomethingPrivate();

System.out.println(fii);

}

}

// 15.virtual function

class Parent {

void v1() //Declaring function

{

System.out.println("Inside the Parent Class");

}

}

public class Child extends Parent{

void v1() // Overriding function from the Parent class

{

System.out.println("Inside the Child Class");

}

public static void main(String args[]){

Parent ob1 = new Child(); //Refering the child class object

using the parent class

ob1.v1();

}

}

// 16. stack in java

// Java code for stack implementation

import java.io.\*;

import java.util.\*;

class Test

{

// Pushing element on the top of the stack

static void stack\_push(Stack<Integer> stack)

{

for(int i = 0; i < 5; i++)

{

stack.push(i);

}

}

// Popping element from the top of the stack

static void stack\_pop(Stack<Integer> stack)

{

System.out.println("Pop Operation:");

for(int i = 0; i < 5; i++)

{

Integer y = (Integer) stack.pop();

System.out.println(y);

}

}

// Displaying element on the top of the stack

static void stack\_peek(Stack<Integer> stack)

{

Integer element = (Integer) stack.peek();

System.out.println("Element on stack top: " + element);

}

// Searching element in the stack

static void stack\_search(Stack<Integer> stack, int element)

{

Integer pos = (Integer) stack.search(element);

if(pos == -1)

System.out.println("Element not found");

else

System.out.println("Element is found at position: " +

pos);

}

public static void main (String[] args)

{

Stack<Integer> stack = new Stack<Integer>();

stack\_push(stack);

stack\_pop(stack);

stack\_push(stack);

stack\_peek(stack);

stack\_search(stack, 2);

stack\_search(stack, 6);

}

}

// 17.queue in java

import java.util.\*;

class Book implements Comparable<Book>{

int id;

String name,author,publisher;

int quantity;

public Book(int id, String name, String author, String publisher, int

quantity) {

this.id = id;

this.name = name;

this.author = author;

this.publisher = publisher;

this.quantity = quantity;

}

public int compareTo(Book b) {

if(id>b.id){

return 1;

}else if(id<b.id){

return -1;

}else{

return 0;

}

}

}

public class LinkedListExample {

public static void main(String[] args) {

Queue<Book> queue=new PriorityQueue<Book>();

//Creating Books

Book b1=new Book(121,"Let us C","Yashwant Kanetkar","BPB",8);

Book b2=new Book(233,"Operating System","Galvin","Wiley",6);

Book b3=new Book(101,"Data Communications &

Networking","Forouzan","Mc Graw Hill",4);

//Adding Books to the queue

queue.add(b1);

queue.add(b2);

queue.add(b3);

System.out.println("Traversing the queue elements:");

//Traversing queue elements

for(Book b:queue){

System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"

"+b.quantity);

}

queue.remove();

System.out.println("After removing one book record:");

for(Book b:queue){

System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"

"+b.quantity);

}

}

}

// 18.sum of two different datatype using parameterized constructor .

class Add

{

int a;

Double b;

Add(int x,Double y)

{

a=x;

b=y;

}

void ans()

{

System.out.println("The Addition is :- "+(a+b));

System.out.println("The substraction is :- "+(a-b));

System.out.println("The multiplication is :- "+(a\*b));

System.out.println("The division is :- "+(a/b));

}

public static void main(String args[])

{

Add a1 = new Add(5,4.5);

a1.ans();

}

}

//19 . arithmetic operators

public class ArithmeticOperator

{

public static void main(String args[])

{

int a=10;

int b=20;

System.out.println("a + b = "+(a+b));

System.out.println("b - a = "+(b-a));

System.out.println("a x b = "+(a\*b));

System.out.println("b / a = "+(b/a));

}

}

//20 . passing data using def constructor & parameterized constructor .

class bca

{

int id;

String name;

bca(int i,String n)

{

id=i;

name=n;

}

bca()

{

}

void display()

{

System.out.println(id+" "+name);

}

public static void main(String args[])

{

bca b1=new bca(101,"ajith");

bca b2=new bca();

b2.id=b1.id;

b2.name=b1.name;

b1.display();

b2.display();

}

}

//21 . multilevel inheritance

class Bikes

{

void speed()

{

System.out.println("Various speed of Bikes: :-)");

}

}

class Splendor extends Bikes

{

void speed()

{

System.out.println("Splendor Runs at 45km/hr !");

}

}

class Shine extends Bikes

{

void speed()

{

System.out.println("Shine Runs at 55km/hr !");

}

}

class CT100 extends Bikes

{

void speed()

{

System.out.println("CT100 Runs at 60km/hr !");

}

public static void main(String args[])

{

Bikes b1,b2,b3,b4;

b1 = new Bikes();

b2 = new Splendor();

b3 = new Shine();

b4 = new CT100();

b1.speed();

b2.speed();

b3.speed();

b4.speed();

}

}

// 22. bitwise operators

public class BitwiseOperator

{

public static void main(String args[])

{

int a=2;

int b=3;

System.out.println("a & b = "+(a&b));

System.out.println("a | b = "+(a|b));

System.out.println("a ^ b = "+(a^b));

System.out.println(" ~a = "+(~a));

a&=b;

System.out.println("a = "+a);

}

}

// 23 . conditonal operators

public class ConditionalOperator

{

public static void main(String args[])

{

int a,b;

a=5;

b=(a==1)?5:7;

System.out.println(b);

b=(a==5)?5:7;

System.out.println(b);

}

}

// 24 . do while program

public class DoWhile

{

public static void main(String args[])

{

int x=21,sum=0;

do

{

sum+=x;

x--;

}

while(x<10);

{

System.out.println("the summation is "+sum);

}

}

}

// 25. for loop program

public class ForLoop

{

public static void main(String args[])

{

int[] numbers={10,20,30,40,50};

for(int x : numbers)

{

System.out.println(x);

System.out.println(",");

}

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

String[] names={"james","larry","tom","lacy"};

for(String name : names)

{

System.out.println(name);

System.out.println(",");

}

}

}

// 26. if else program

public class IfElse

{

public static void main(String args[])

{

int a=10;

if(a<5)

System.out.println("a is less than 5 .");

else

System.out.println("a is greater than 5 .");

}

}

// 27. run time polymorphism

class Bank

{

float getRateOfInterest()

{

return 0;

}

}

class SBI extends Bank

{

float getRateOfInterest()

{

return 8.4f;

}

}

class ICICI extends Bank

{

float getRateOfInterest()

{

return 7.3f;

}

}

class AXIS extends Bank

{

float getRateOfInterest()

{

return 9.7f;

}

}

class TestPolymorphism

{

public static void main(String args[])

{

Bank b;

b=new SBI();

System.out.println("sbi rate of interest "+b.getRateOfInterest());

b=new ICICI();

System.out.println("ICICI rate of interest "+b.getRateOfInterest());

b=new AXIS();

System.out.println("AXIS rate of interest "+b.getRateOfInterest());

}

}

// 28. static variable use program

class math

{

int a;

double b;

static double c = 5.5;

math(int x,double y)

{

a=x;

b=y;

}

void sum()

{

System.out.println("a x b x c = "+(a\*b\*c));

}

public static void main(String args[])

{

math m1=new math(5,2.5);

m1.sum();

}

}

// 29. sum of two digits using user input

import java.util.\*;

class UserInputDemo

{

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in); //System.in is a standard input stream

System.out.print("Enter first number- ");

int a= sc.nextInt();

System.out.print("Enter second number- ");

int b= sc.nextInt();

System.out.print("Enter third number- ");

int c= sc.nextInt();

int d=a+b+c;

System.out.println("Total= " +d);

}

}

// 30. string user input

import java.util.\*;

class UserInputDemo1

{

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in); //System.in is a standard input stream

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

String str= sc.nextLine(); //reads string

System.out.print("You have entered: "+str);

}

}

// 31 . prime number program

public class PrimeExample{

public static void main(String args[]){

int i,m=0,flag=0;

int n=3;//it is the number to be checked

m=n/2;

if(n==0||n==1){

System.out.println(n+" is not prime number");

}else{

for(i=2;i<=m;i++){

if(n%i==0){

System.out.println(n+" is not prime number");

flag=1;

break;

}

}

if(flag==0) { System.out.println(n+" is prime number"); }

}//end of else

}

}

// 32. factorial of n number

class FactorialExample{

public static void main(String args[]){

int i,fact=1;

int number=5;//It is the number to calculate factorial

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

fact=fact\*i;

}

System.out.println("Factorial of "+number+" is: "+fact);

}

}

// 33. right triangle pattern program in java

public class RightTrianglePattern

{

public static void main(String args[])

{

//i for rows and j for columns

//row denotes the number of rows you want to print

int i, j, row=6;

//outer loop for rows

for(i=0; i<row; i++)

{

//inner loop for columns

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

{

//prints stars

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

}

//throws the cursor in a new line after printing each line

System.out.println();

}

}

}

// 34 . left triangle pattern program in java

public class LeftTrianglePattern

{

public static void main(String args[])

{

//i for rows and j for columns

//row denotes the number of rows you want to print

int i, j, row = 6;

//Outer loop work for rows

for (i=0; i<row; i++)

{

//inner loop work for space

for (j=2\*(row-i); j>=0; j--)

{

//prints space between two stars

System.out.print(" ");

}

//inner loop for columns

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

{

//prints star

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

}

//throws the cursor in a new line after printing each line

System.out.println();

}

}

}

// 35 . pyramid pattern program in java

public class PyramidPattern

{

public static void main(String args[])

{

//i for rows and j for columns

//row denotes the number of rows you want to print

int i, j, row = 6;

//Outer loop work for rows

for (i=0; i<row; i++)

{

//inner loop work for space

for (j=row-i; j>1; j--)

{

//prints space between two stars

System.out.print(" ");

}

//inner loop for columns

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

{

//prints star

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

}

//throws the cursor in a new line after printing each line

System.out.println();

}

}

}

// 36 . diamond pattern program in java

import java.util.Scanner;

public class DiamondPattern

{

public static void main(String args[])

{

int row, i, j, space = 1;

System.out.print("Enter the number of rows you want to print: ");

Scanner sc = new Scanner(System.in);

row = sc.nextInt();

space = row - 1;

for (j = 1; j<= row; j++)

{

for (i = 1; i<= space; i++)

{

System.out.print(" ");

}

space--;

for (i = 1; i <= 2 \* j - 1; i++)

{

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

}

System.out.println("");

}

space = 1;

for (j = 1; j<= row - 1; j++)

{

for (i = 1; i<= space; i++)

{

System.out.print(" ");

}

space++;

for (i = 1; i<= 2 \* (row - j) - 1; i++)

{

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

}

System.out.println("");

}

}

}

// 37. check the no. weather its positive or negative

public class CheckPositiveOrNegativeExample1

{

public static void main(String[] args)

{

//number to be check

int num=912;

//checks the number is greater than 0 or not

if(num>0)

{

System.out.println("The number is positive.");

}

//checks the number is less than 0 or not

else if(num<0)

{

System.out.println("The number is negative.");

}

//executes when the above two conditions return false

else

{

System.out.println("The number is zero.");

}

}

}

// 38 . check the no. weather its positive or negative via user input

import java.util.Scanner;

public class CheckPositiveOrNegativeExample2

{

public static void main(String[] args)

{

int num;

//object of the Scanner class

Scanner sc = new Scanner(System.in);

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

//reading a number from the user

num = sc.nextInt();

//checks the number is greater than 0 or not

if(num>0)

{

System.out.println("The number is positive.");

}

//checks the number is less than 0 or not

else if(num<0)

{

System.out.println("The number is negative.");

}

//executes when the above two conditions return false

else

{

System.out.println("The number is zero.");

}

}

}

// 39. reverse number in java

public class ReverseNumberExample1

{

public static void main(String[] args)

{

int number = 987654, reverse = 0;

while(number != 0)

{

int remainder = number % 10;

reverse = reverse \* 10 + remainder;

number = number/10;

}

System.out.println("The reverse of the given number is: " + reverse);

}

}

// 40. fibonacci series program in java

class FibonacciExample1{

public static void main(String args[])

{

int n1=0,n2=1,n3,i,count=10;

System.out.print(n1+" "+n2);//printing 0 and 1

for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{

n3=n1+n2;

System.out.print(" "+n3);

n1=n2;

n2=n3;

}

}

}

// 41. print ascii values in java

public class PrintAsciiValueExample1

{

public static void main(String[] args)

{

// character whose ASCII value to be found

char ch1 = 'a';

char ch2 = 'b';

// variable that stores the integer value of the character

int asciivalue1 = ch1;

int asciivalue2 = ch2;

System.out.println("The ASCII value of " + ch1 + " is: " + asciivalue1);

System.out.println("The ASCII value of " + ch2 + " is: " + asciivalue2);

}

}

// 42. palindrome number program in java

class PalindromeExample{

public static void main(String args[]){

int r,sum=0,temp;

int n=454;//It is the number variable to be checked for palindrome

temp=n;

while(n>0){

r=n%10; //getting remainder

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

System.out.println("palindrome number ");

else

System.out.println("not palindrome");

}

}