### 1: Adding a Method to the Box Class

class Rectangle

{

double width;

double height;

void volume()

{

System.out.println("Volume is ");

System.out.println(width \* height \*);

}

}

class Demo

{

public static void main (String args[])

{

Rectangle myRectangle1 = new Rectangle();

Rectangle my Rectangle 2 = new Rectangle ();

myRectangle1.height = 50;

myRectangle1.width = 70;

myRectangle2.width = 4;

myRectangle2.height = 7;

myRectangle1.volume();

myRectangle2.volume();

} }

**OUTPUT:**

Volume is 3500.0

Volume is 28.0

### 2: Demonstrating Static Variable

class StaticVar

{

static int p = 57;

static int r = 76;

static void callme()

{

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

}

}

class Variable

{

public static void main(String args[])

{

StaticDemo.callme();

System.out.println("r= " + StaticVar.r);

}

}

**OUTPUT**:

a=57

b=76

### 3:Write your own program and excecute for demonstrating Default Constructor (No-argument constructor)

class Rectangle

{

double width;

double height;

Rectangle ()

{

System.out.println("Constructing a Rectangle ");

width = 34;

height = 89;

}

double volume()

{

return width \* height;

}

}

**OUTPUT:**

Constructing a Rectangle

3026

### 4: Parameterized constructor

### class Rectangle

### {

double width; double height;

Box(double w, double h)

{

width = w;

height = h;

}

double volume()

{

return width \* height ;

}

}

class Demo

{

public static void main(String args[])

{

Rectangle myRectangle1 = new Rectangle(26,75);

Rectangle myRectangle2 = new Rectangle(6,8);

double vol;

vol = myRectangle1.volume();

System.out.println("Volume is " + vol);

vol = myRectangle2.volume();

System.out.println("Volume is " + vol); }

}

**OUTPUT:**

Volume is 1950.0

Volume is 48.0

## EXERCISE 5: Demonstrating finalize() method in java Example 1

public **class** Work1

{

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

{

Work1 obj = **new** Work1();

obj = **null**;

System.gc();

System.out.println("Done");

}

@Override

protected **void** finalize()

{

System.out.println("finilize() method called ");

}

}

**OUTPUT:**

Done

finalize() method called

## Exercise 6: Demonstrate Switch statement

**import** java.util.Scanner;

**public** **class** CurrentAge{

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

{

Scanner in = **new** Scanner(System.***in***); System.***out***.print("Input Current Age: ");

**int** age = in.nextInt();

**switch** (age)

{

**case** (1):

System.***out***.println("Your age is 12");

**break**;

**case** (2):

System.***out***.println("You age is 13");

**break**;

**case** (3):

System.***out***.println("You age is 15");

**break**;

**case** (4):

System.***out***.println("You age is 16");

**break**;

**case** (5):

System.***out***.println("You age is 17");

**break**;

**case** (6):

System.***out***.println("You age is 18");

**break**; **default**:

System.***out***.println("Please Enter your valid age below.");

**break**;

}

}

**OUTPUT:**

Input sem:18

You Current Age is 18.

## Exercise 7: Demonstrate for statement

**public** **class** Exercise17

{

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

{

**int** a[][]={{1,3,4},{2,4,3},{3,4,5}};

**int** b[][]={{1,3,4},{2,4,3},{1,2,4}};

**int** c[][]=**new** **int**[3][3];

**for**(**int** i=0;i<3;i++)

{

**for**(**int** j=0;j<3;j++){

c[i][j]=a[i][j]+b[i][j];

System.***out***.print(c[i][j]+"\t");

}

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

}

}

**OUTPUT:**

2 6 8

4 8 6

4 6 9

## 8: Demonstrate encapsulation concept

**class** Encapsulation{

**private** String name;

**private** **int** roll;

**private** **int** age;

**public** **int** getAge() { **return** age;}

**public** String getName() { **return** name;}

**public** **int** getRoll() { **return** roll; }

**public** **void** setAge(**int** newAge) { age = newAge; }

**public** **void** setName(String newName){ name = newName; }

**public** **void** setRoll(**int** newRoll){ roll = newRoll; }

}

### public class Demo{

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

{

Encapsulate obj = **new** Encapsulate();

obj.setName("Pruthvi");

obj.setAge(07);

obj.setRoll(57);

System.***out***.println("name: " + obj.getName());

System.***out***.println("age: " + obj.getAge());

System.***out***.println("roll: " + obj.getRoll());

}

}

**OUTPUT:**

name: Pruthvi

age: 07

roll: 57

**9:** **Implement the concept of importing classes from user defined package and creating packages**

*/\* CREATE A NEW FOLDER WITH NAME pkg1 INSIDE THE pkg1 FOLDER SAVE A.java\*/*

package lkg1;

public class P

{

public void displayP()

{

System.out.println("class P");

}

}

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

*/\* CREATE A NEW FOLDER WITH NAME pkg2 INSIDE THE pkg2 FOLDER SAVE B.java\*/* package lkg2;

public class R

{

protected int u=57;

public void displayR()

{

System.out.println("class R");

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

}

}

-------------------------------------------------------------------------------- import lkg1.P;

import pkg2.\*;

class Demo

{

public static void main(String args[])

{

P p= new P();

R r= new R();

p.displayP();

r.displayR();

}

}

**OUTPUT:**

class P

class R

u=57

## 10: write the program so that the class should have only one responsibility and should satisfy SRP [Single Responsibility Principle]

**class** Calculator1 {

**public** **static** **int** add(**int** x, **int** y) { **return** x + y; }

**public** **static** **int** sub(**int** x, **int** y) { **return** x - y; }

**public** **static** **int** mul(**int** x, **int** y) { **return** x \* y; }

**public** **static** **int** div(**int** x, **int** y) { **return** x / y; }

}

**class** ResultPrinter {

**public** **static** **void** printResult(**int** value)

{

System.***out***.println("The value is="+value);

}

}

**public** **class** Exercise10

{

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

{

**int** a = Calculator.*add*(79, 45);

ResultPrinter.*printResult*(a);

**int** b = Calculator.*sub*(79, 45);

ResultPrinter.*printResult*(b);

**int** c = Calculator.*mul*(79, 45);

ResultPrinter.*printResult*(c);

**int** d = Calculator.*div*(79, 45);

ResultPrinter.*printResult*(d);

}

}

**OUTPUT:**

The value is=124

The value is=34

The value is=35,866

The value is=1.75

## Exercise 11: Illustrate creating an array of objects

**class** Player

{

**public** **int** number;

**public** String name;

Student(**int** number, String name)

{

**this**.number = number;

**this**.name = name;

}

}

**public** **class** Sports

{

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

{

Player[] arr;

arr = **new** Player[5];

arr[0] = **new** Player (1,"Ronaldo");

arr[1] = **new** Player (2,"Messi");

arr[2] = **new** Player (3,"Neymar");

arr[3] = **new** Player (4,"Benjama ");

arr[4] = **new** Player (5,"Pele");

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

System.***out***.println("Element at " + i + " : " +

arr[i].roll\_no +" "+ arr[i].name);

}

}

**OUTPUT:**

Element at 0: 1 Ronaldo

Element at 1: 2 Messi

Element at 2: 3 Neymar

Element at 3: 4 Benajama

Element at 4: 5 Pele

## 12: Demonstration of String Class and its Methods

String

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

{

System.***out***.println("-------------STRING DEMO-------");

String s1=**new** String("Cricket");

String s2="cricket";

String s3="";

System.***out***.println("the string s1="+s1);

System.***out***.println("the string s2="+s2);

System.***out***.println("Char at ="+s2.charAt(2));

System.***out***.println("the length of the string s1="+s1.length());

System.***out***.println("SubString Begin index ="+s1.substring(1));

System.***out***.println("SubString Begin & end index ="+s1.substring(1,4));

System.***out***.println("s1 equals s2 is="+s1.equals(s2));

System.***out***.println("check isempty="+s3.isEmpty());

System.***out***.println("s1 concatination s2 is="+s1.concat(s2));

System.out.println("the length of the string s1="+s1.length());  
 System.***out***.println("Replace="+s1.replace("C", "T"));

System.***out***.println("Lower case is="+s1.toLowerCase());

System.***out***.println("Upper case is="+s1.toUpperCase());

System.***out***.println("s1 equals ignore case s2="+s1.equalsIgnoreCase(s2));

}

**OUTPUT:**

the string s1=Cricket

the string s2=cricket

Char at=p

the length of the string s1=7

SubString Begin index=ricket

SubString Begin & index =ricke

s1 equals s2 is=false

check isempty=true

s1 concatination s2 is=Cricketcricket

the length of the string s1=7

Replace=Ticket

Lower case is=cricket

Upper case is=CRICKET

s1 equals ignore case s2=false

## Exercise 13: Demonstration of Multilevel Inheritance

class GrandParent

{

public void gMethod() { System.out.println("GrandParent method"); }

}

class Parent`extends GrandParent

{

public void pMethod() { gMethod();

System.out.println("Parent method"); }

}

class Child extends Parent

{

public void cMethod() { pMethod();

System.out.println("Child method"); }

}

class MultilevelDemo

{

public static void main(String[] args)

{

Child cobj = new Child();

cobj.cMethod();

cobj.pMethod();

cobj.gMethod();

}

}

**OUTPUT:**

GrandParent method

Parent method

Child method

GrandParent method

Parent method

Child method

## Exercise 14: Demonstration an Open Closed Principle [OCP]

**interface** Shape

{

**public** **double** calculateArea();

}

**class** Rect **implements** Shape

{

**double** length;

**double** width;

**public** **double** calculateArea(){

**return** length \* width;

}

}

**class** Cir **implements** Shape

{

**public** **double** radius;

**public** **double** calculateArea(){

**return** (3.147\*radius\*radius);

}

}

**class** Square **implements** Shape

{

**double** l;

**public** **double** calculateArea() {

**return** (l\*l);

}

}

**class** AreaCal

{

**public** **double** calculateShapeArea(Shape shape){

**return** shape.calculateArea();

}

}

**public** **class** OCPDemo {

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

{

AreaCal a1 = **new** AreaCal();

Rect r = **new** Rect();

r.length=10;

r.width=20;

Cir c= **new** Cir();

c.radius=10;

Square s = **new** Square();

s.l=30;

**double** r1= a1.calculateShapeArea(r);

**double** c1= a1.calculateShapeArea(c);

**double** s1=a1.calculateShapeArea(s);

System.***out***.println("----------OUTPUT--------");

System.***out***.println("Area of Rectangle="+r1);

System.***out***.println("Area of Circle="+c1);

System.***out***.println("Area of Square="+s1);

}

}

**OUTPUT:**

----------OUTPUT--------

Area of Rectangle=200.0

Area of circle=314.7

Area of square=900.0

**Exercise 15: Demonstration of method OverLoading.**

class OverloadDemo

{

void method()

{

System.out.println("No parameters");

}

void method(int a)

{

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

}

void method(int a, int b)

{

System.out.println("a and b: " + a + " " + b);

}

double method(double a)

{

System.out.println("double a: " + a); return a\*a;

}

}

class OverloadExample

{

public static void main(String args[]) {

OverloadDemo ob = new OverloadDemo();

double result;

ob.method();

ob.method(10);

ob.method(10, 20);

result = ob.method(123.25);

System.out.println("Result of ob.method(123.25): " + result);

}

}

**OUTPUT:**

No parameters

a:10

a and b: 10 20

double a:123.25

result of ob.method(123.25): 15190.5625

**Exercise 16: Demonstration of method overriding.**

class A

{

int i, j;

A(int a, int b) { i = a; j = b; }

void show() { System.out.println("i and j: " + i + " " + j); }

} class B extends A

{ int k;

B(int a, int b, int c)

{ super(a, b); k = c;

}

void show()

{

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

}

}

class Override

{ public static void main(String args[])

{

B subOb = new B(1, 2, 3);

subOb.show();

}

}

**OUTPUT:**

## 

## 17: Demonstration of Abstract class

abstract class A

{

abstract void callme();

}

class B extends A

{

void callme()

{

System.out.println("this is call me inside child.");

}

public static void main(String[] args)

{

B b = new B();

b.callme();

}

}

**OUTPUT:**

## Exercise 36: Demonstration of Interface concept in Java

## interface Area

{

final static float pi=3.14f;

double compute(double x, double y);

}

interface Display extends Area

{

void display\_result(double result);

}

class Rectangle implements Display

{

public double compute(double x, double y)

{

return(pi\*x\*y);

}

public void display\_result(double result)

{

System.out.println("The area is:"+result);

}

}

class InterfaceDemo

{

public static void main(String args[])

{

Rectangle r1=new Rectangle();

double res=r1.compute(10.2,20.4);

r1.display\_result(res);

}

}

**OUTPUT:**

**Exercise 19:Demonstration of file writer in java**

Import java.io.\*;

Class FileWriter {

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

FileWriter fw=new FileWriter(“Shashi.txt”);

String s=”Sri Jaya chamarajendra polytechnic Bangalore”;

Char ch[]=s.toCharArray();

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

fw.write(ch[i]);

fw.close();

}

}

**OUTPUT:**

## 

**Exercise 20:Demonstration of file reader class in java**

Import java.io.\*;

Class MyFileReader {

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

Int i = 0;

FileReader fr=new FileReader(“Shashi.txt”);

While ((i=fr.read()) != -1)

System.out.println((char) i);

fr.close();

}

}

**OUTPUT:**

**21: Multiple Catch Block in Exception Handling.**

**public** **class** MultipleCatchBlock1 {

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

**try**{

**int** a[]=**new** **int**[5];

a[5]=30/0;

}

**catch**(ArithmeticException e)

{

System.out.println("Arithmetic Exception occurs");

}

**catch**(ArrayIndexOutOfBoundsException e)

{

System.out.println("ArrayIndexOutOfBounds Exception occurs");

}

**catch**(Exception e)

{

System.out.println("Parent Exception occurs");

}

System.out.println("rest of the code");

}

}

**OUTPUT:**

**Exercise 22: Demonstration of Throwing Our Own Exceptions.**

**public** **class** TestThrow1{

**static** **void** validate(**int** age)

{

**if**(age<18)

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

**else**

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

}

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

{

validate(13);

System.out.println("rest of the code...");

}

}

## 23: Design an interface & implement it like one that builds different types of toys and check compliance with ISP [ Interface Segregation principle]

**interface** Toy

{ **void** price(**double** p); **void** color(String c);

}

**interface** Move

{ **void** move();

}

**interface** flying

{

**void** fly();

}

**class** car **implements** Toy,Move

{

**double** p; String c;

**public** **void** price(**double** p)

{

**this**.p=p;

System.***out***.println("The car Toy"); System.***out***.println("price =" +p);

}

**public** **void** color(String c)

{

**this**.c=c;

System.***out***.println("color =" +c);

}

**public** **void** move()

{

System.***out***.println("start to move forward and backward ");

}

}

**class** plane **implements** Toy, flying

{

**double** p; String c;

**public** **void** price(**double** p)

{

**this**.p=p;

System.***out***.println("The Plane Toy");

System.***out***.println("price =" +p);

}

**public** **void** color(String c)

{

**this**.c=c;

System.***out***.println("color =" +c);

}

**public** **void** move()

{

System.***out***.println("plane can move forward and backward ");

}

**public** **void** fly()

{

System.***out***.println("starts flying");

}

}

**public** **class** Toyb1

{

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

{

car c=**new** car();

c.price(12.0);

c.color("red");

c.move();

plane p=**new** plane();

p.price(15.0);

p.color("yellow");

p.move();

p.fly();

}

}

OUTPUT:

The Car Toy

Price= 12.0

Color=red

Plane can move forward and backward

Starts flying

## Exercise 24: Demonstrating Enum in Java

**class** Workingdays

{

**enum** Day

{

***Sunday***, ***Monday***, ***Tuesday***, ***Wednesday***, ***Thursday***, ***Friday***, ***Saturday***

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

{ **for** (Day d : Day.*values*())

{

*weekend*(d);

}

}

**static** **void** weekend (Day d)

{

**if** (d.equals (Day.***Sunday***))

System.***out***.println("value = "+ d + "is a Holiday"); **else**

System.***out***.println("value ="+ d + " is a working day"); }}

OUTPUT:  
 value = Sunday is Holiday

value = Monday is Working day

value = Tuesday is Working day

value = Wednesday is Working day

value = Thursday is Working day

value = Friday is Working day

value = Saturday is Working day