**Experiment No :01**

Title: Design a class Complex with data members for real and imaginary part. Provide default and Parameterized constructors. Write a program to perform arithmetic operations of two complex numbers.

CODE: \*\*/

class ComplexNumber

{

int real, image;

public ComplexNumber(int r, int i)

{

this.real = r;

this.image =i;

}

public void showC()

{

System.out.print(this.real + "+i" + this.image);

}

public static ComplexNumber add(ComplexNumber n1,

ComplexNumber n2)

{

ComplexNumber res = new ComplexNumber(0, 0);

res.real = n1.real + n2.real;

res.image = n1.image + n2.image;

return res;

}

public static void main(String arg[])

{

ComplexNumber c1 = new ComplexNumber(4, 5);

ComplexNumber c2 = new ComplexNumber(10, 5);

System.out.print("first Complex number: ");

c1.showC();

System.out.print("\nSecond Complex number: ");

c2.showC();

ComplexNumber res = add(c1, c2);

System.out.println("\nAddition is :");

res.showC();

}

}

/\*\* OUTPUT:

first Complex number: 4+i5

Second Complex number: 10+i5

Addition is :

14+i10BUILD SUCCESSFUL (total time: 0 seconds) \*\*/

Experiment no:2

Identify commonalities and differences between Publication, Book and Magazine classes. Title, Price, Copies are common instance variables and saleCopy is common method. The differences are, Book class has author and orderCopies(). Magazine Class has orderQty, Currentissue, reciveissue(). Write a program to find how many copies of the given books are

ordered and display total sale of publication

\*/

class Publication

{

String title;

int copies, price;

void saleCopy()

{

System.out.println("Total Sale of Publication is: "+(copies\*price));

}

}

class Book extends Publication

{

String author;

Book(String n, String t, int c, int p)

{

this.author=n;

this.title=t;

this.copies=c;

this.price=p;

}

void orderCopies()

{

System.out.println("No of ordered book copies"+this.copies);

}

void saleCopy()

{

int total=this.price\*this.copies;

System.out.println("Amount generated from sale of books "+total);

}

}

class Magazine extends Publication

{

int orderqty, issue,receiveissue;

Magazine(String t, int c, int p, int ri, int i)

{

this.title=t;

this.copies=c;

this.price=p;

receiveissue=ri;

}

void orderCopies()

{

System.out.println("No of magazines ordered "+this.copies);

}

void saleCopy()

{

int total=this.price\*this.copies;

System.out.println("Total amount generated from sale of magazines "+total);

}

}

class Total

{

public static void main(String args[])

{

Book b1= new Book("ABC","XYZ",2,240);

b1.orderCopies();b1.saleCopy();

Magazine m=new Magazine("XYZ",10,200,10,15);

m.orderCopies();m.saleCopy();

}

}

/\*

Output:

No of ordered book copies5

Amount generated from sale of books 600

No of magazines ordered 15

Total amount generated from sale of magazines 3600

\*/

Experiment no:3

Design and develop inheritance for a given case study, identify objects and relationships and implement inheritance wherever applicable. Employee class with Emp\_name, Emp\_id, Address, Mail\_id, and Mobile\_no as members. Inherit the classes, Programmer, Team Lead, Assistant Project Manager and Project Manager from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.

\*/

class Employee

{

String name,add,mail;

float id,mobile,basic;

void salary()

{

float da,hra,pf,cf,gross;

da = basic\*97/100;

hra = basic \* 10/100;

pf = basic \*12/100;

cf = basic\*0.1f/100;

gross = basic+da+hra-pf-cf;

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

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

System.out.println("Gross Salary:"+gross);

System.out.println("mailid :"+mail);

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

}

}

class Programmer extends Employee

{

Programmer(String name,int sal,String mail)

{

System.out.println("Programmer Salary Details:");

this.name = name;

this.mail = mail;

basic = sal;

}

}

class TeamLead extends Employee

{

TeamLead(String name,int sal,String mail)

{

System.out.println("Team Leader Salary Details:");

this.name = name;

this.mail = mail;

basic = sal;

}

}

class APM extends Employee

{

APM(String name,int sal,String mail)

{

System.out.println("Assistant Project Manager Salary Details:");

this.name = name;

this.mail = mail;

basic = sal;

}

}

class ProjectManager extends Employee

{

ProjectManager(String name,int sal,String mail)

{

System.out.println("Project Manager Salary Details:");

this.name = name;

this.mail = mail;

basic = sal;

}

}

public class Inheritance

{

Programmer P = new Programmer("k.A Joshipura",30000,"kajoshi@gmail.com");

P.salary();

TeamLead t = new TeamLead("Mohan Agarwal",40000,"mohan@gmail.com");

t.salary();

APM A = new APM("Ramesh Patil",50000,"rp@gmail.com");

A.salary();

ProjectManager PM = new ProjectManager("sita mane",30000,"sita@hotmaill.com");

PM.salary();

}

}

/\*\*

Output:

Programmer Salary Details:

Name:k.A Joshipura

Basic salary:30000.0

Gross Salary:58470.0

mailid :kajoshi@gmail.com

Team Leader Salary Details:

Name:Mohan Agarwal

Basic salary:40000.0

Gross Salary:77960.0

mailid :mohan@gmail.com

Assistant Project Manager Salary Details:

Name:Ramesh Patil

Basic salary:50000.0

Gross Salary:97450.0

mailid :rp@gmail.com

Project Manager Salary Details:

Name:sita mane

Basic salary:30000.0

Gross Salary:58470.0

mailid :sita@hotmaill.com

Experiment no:4

Design a base class shape with two double type values and member functions to input the data and compute\_area() for calculating area of figure. Derive two classes’

triangle and rectangle. Make compute\_area() as abstract function and redefine this function in the derived class to suit their requirements. Write a program that accepts dimensions of triangle/rectangle and display calculated area. Implement dynamic binding for given case study.\*/

abstract class Shape

{

double length, breadth;

void input()

{

Scanner s = new Scanner(System.in);

System.out.println("Enter first value");

length = s.nextDouble();

System.out.println("Enter Second value");

breadth = s.nextDouble();

}

abstract void compute\_area();

}

class Triangle extends Shape

{

public void compute\_area()

{

double area;

area = 0.5 \* length \* breadth;

System.out.println("Traiangle Area is :"+area);

}

}

class Rectangle extends Shape

{

public void compute\_area()

{

double area;

area = length \* breadth;

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

}

}

public class Dynamic

{

public static void main(String args[])

{

Triangle t = new Triangle();

Rectangle r = new Rectangle();

t.input();

t.compute\_area();

r.input();

r.compute\_area();

}

}

/\*

Output

Enter first value

40

Enter Second value

20

Traiangle Area is :400.0

Enter first value

15

Enter Second value

15

Rectangle area is:225.0

\*/

Experiment no:5

Design and develop a context for given case study and implement an interface for Vehicles

Consider the example of vehicles like bicycle, car, and bike. All Vehicles have common

functionalities such as Gear Change, Speed up and apply breaks . Make an interface and put all

these common functionalities. Bicycle, Bike, Car classes should be implemented for all these

functionalities in their own class in their own way.

\*/

import java.io.\*;

interface Vehicle

{

void changeGear(int a);

void speedUp(int a);

void applyBrakes(int a);

}

class Bicycle implements Vehicle

{

int speed;

int gear;

public void changeGear(int newGear)

{

gear = newGear;

}

public void speedUp(int increment)

{

speed = speed + increment;

}

public void applyBrakes(int decrement)

{

speed = speed - decrement;

}

public void printStates()

{

System.out.println("speed: " + speed + " gear: " + gear);

}

}

class Bike implements Vehicle

{

int speed;

int gear;

public void changeGear(int newGear)

{

gear = newGear;

}

public void speedUp(int increment)

{

speed = speed + increment;

}

public void applyBrakes(int decrement)

{

speed = speed - decrement;

}

public void printStates() {

System.out.println("speed: " + speed + " gear: " + gear);

}

}

class Car implements Vehicle

{

int speed;

int gear;

public void changeGear(int newGear)

{

gear = newGear;

}

public void speedUp(int increment)

{

speed = speed + increment;

}

public void applyBrakes(int decrement)

{

speed = speed - decrement;

}

public void printStates() {

System.out.println("speed: " + speed + " gear: " + gear);

}

}

class Interface

{

public static void main (String[] args)

{

Bicycle bicycle = new Bicycle();

bicycle.changeGear(2);

bicycle.speedUp(3);

bicycle.applyBrakes(1);

System.out.println("Bicycle present state :");

bicycle.printStates();

// creating instance of bike.

Bike bike = new Bike();

bike.changeGear(1);

bike.speedUp(4);

bike.applyBrakes(3);

System.out.println("Bike present state :");

bike.printStates();

Car car = new Car();

car.changeGear(3);

car.speedUp(5);

car.applyBrakes(3);

System.out.println("Car present state :");

car.printStates();

}

}

/\*

Output:

Bicycle present state :

speed: -1 gear: 3

Bike present state :

speed: 2 gear: 3

Car present state :

speed: 5 gear: 5

\*/

Interface online course

interface OnlineCourse

{

void getDetails();

}

class DataStructure implements OnlineCourse

{

public void getDetails()

{

System.out.println("Oldest");

}

}

class Java implements OnlineCourse

{

public void getDetails()

{

System.out.println("widely used");

}

}

class Python implements OnlineCourse

{

public void getDetails()

{

System.out.println("Emerging");

}

}

class CourseFactory

{

public OnlineCourse chooseCourse(String str)

{

if(str.equals("Mostly used"))

return new Java();

else if(str.equals("Machine Learning"))

return new Python();

else

return new DataStructure();

}

}

public class OnlineCourseMain

{

public static void main (String [] args)

{

CourseFactory cf= new CourseFactory();

OnlineCourse obj=cf.chooseCourse ("Mostly used");

obj.getDetails();

}

}

/\*public class OnlineCourseMain

{

public static void main (String [] args)

{

OnlineCourse obj= new Python();

System.out.println(obj.getDetails());

}

}\*/

Output:

Widely used

interface OS

{

void spec();

}

class Android implements OS

{

public void spec()

{

System.out.println("Most powerful OS");

}

}

class iOS implements OS

{

public void spec()

{

System.out.println("Most secure OS");

}

}

class Lumia implements OS

{

public void spec()

{

System.out.println("Least used OS, about to die");

}

}

class OperatingSystemFactory

{

public OS getInstance (String str)

{

if (str.equals("Open"))

return new Android();

else if (str.equals("Closed"))

return new iOS();

else

return new Lumia();

}

}

class FactoryMain

{

public static void main (String [] args)

{

OperatingSystemFactory osf=new OperatingSystemFactory();

OS obj= osf.getInstance("Open");

obj.spec();

}

}

Output:

Most powerful OS

Program:

interface Subject

{

void getDetails();

}

class DataStructure implements Subject

{

public void getDetails()

{

System.out.println ("this is the most difficult sub");

}

}

class OOP implements Subject

{

public void getDetails()

{

System.out.println ("this is the most easiest sub");

}

}

class DS implements Subject

{

public void getDetails()

{

System.out.println ("Good for concepts");

}

}

class SubjectFactory

{

public Subject chooseCourse(String str)

{

if(str.equals("Challenging"))

return new DS();

else if(str.equals("Powerful"))

return new OOP();

else

return new DataStructure();

}

}

class SubjectMain

{

public static void main (String [] args)

{

SubjectFactory sf= new SubjectFactory();

Subject sub=sf.chooseCourse("Powerful");

sub.getDetails();

}

}

/\*class SubjectMain

{

public static void main (String [] args)

{

Subject sf= new OOP();

System.out.println(sf.getDetails());

}

}\*/

Output:

this is the most easiest sub

Template.jaava

import java.util.List;

import java.util.\*;

interface SomeSpecialNumber<T>

{

public boolean match(T t);

}

class PrimeNumber implements SomeSpecialNumber<Integer>

{

public boolean match(Integer t)

{

for (int i = 2; i < t; ++i)

{

if (t % i == 0) {

return false;

}

}

return true;

}

}

class EvenNumber implements SomeSpecialNumber<Integer>

{

public boolean match(Integer t)

{

return (t % 2 == 0);

}

}

class Algorithm

{

public <T> int count(List<T> list, SomeSpecialNumber<T> s)

{

int count = 0;

for (T type : list)

{

if (s.match(type))

{

++count;

}

}

return count;

}

}

public class Template

{

public void testSpecificElement()

{

List<Integer> list = Arrays.asList(1, 4, 5, 7, 8, 9, 13, 11);

PrimeNumber pn = new PrimeNumber();

EvenNumber en = new EvenNumber();

Algorithm a = new Algorithm();

System.out.println(" Count of prime numbers: " + a.count(list, pn));

System.out.println(" Count of even numbers: " + a.count(list, en));

}

public static void main(String[] args)

{

Template tq = new Template();

tq.testSpecificElement();

}

}

Output:

Count of prime numbers: 5

Count of even numbers: 2

Experiment no :6

Implement a program to handle Arithmetic exception, Array Index

Out of Bounds. The user enters two numbers Num1 and Num2. The

division of Num1 and Num2 is displayed. If Num1 and Num2 are

not integers, the program would throw a Number Format

Exception. If Num2 were zero, the program would throw an

Arithmetic Exception. Display the exception.

\*/

program :

import java.util.\*;

class NumberException

{

public static void main(String args[])

{

Scanner scr=new Scanner(System.in);

int num1,num2,q;

try

{

System.out.println("\nEnter the value of first integer : ");

num1=Integer.parseInt(scr.next());

System.out.println("\nEnter the value of second integer : ");

num2=Integer.parseInt(scr.next());

q=num1/num2;

System.out.println("\nQuotient is : "+q);

}

catch(NumberFormatException e)

{

System.out.println(e);

}

catch(ArithmeticException e)

{

System.out.println(e);

}

}

}

Output :

/\*

Enter the value of first integer :

16

Enter the value of second integer :

4

Quotient is : 4

\*/

program :

class NestedTry

{

public static void main(String args[])

{

try {

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

System.out.println(arr[5]);

try {

int x = arr[2] / 0; // performe division by zero

}

catch (ArithmeticException e2) {

System.out.println("division by zero is not possible");

}

}

catch (ArrayIndexOutOfBoundsException e1) {

System.out.println("ArrayIndexOutOfBoundsException");

System.out.println("Element at such index does not exists");

}

}

}

Output :

ArrayIndexOutOfBoundsException

Element at such index does not exists

\*/

12 age exception program :

import java.util.Scanner;

class AgeException extends Exception {

public AgeException(String str) {

System.out.println(str);

}

}

public class AgeExceptionDemo {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.print("Please enter your age: ");

int age = s.nextInt();

try {

if(age < 18)

throw new AgeException("Invalid age");

else

System.out.println("Valid age");

}

catch (AgeException a) {

System.out.println(a);

}

}

}

Output :

Please enter your age: 23

Valid age

Please enter your age: 15

Invalid age

AgeException

**Experiment no :7**

Implement a generic program using any collection class to count

the number of elements in a collection that have a specific property

such as even numbers, odd number, prime number and

palindromes.

\*/

program :

import java.util.List;

import java.util.\*;

interface SomeSpecialNumber<T>

{

public boolean match(T t);

}

class PrimeNumber implements SomeSpecialNumber<Integer>

{

public boolean match(Integer t)

{

for (int i = 2; i < t; ++i)

{

if (t % i == 0) {

return false;

}

}

return true;

}

}

class EvenNumber implements SomeSpecialNumber<Integer>

{

public boolean match(Integer t)

{

return (t % 2 == 0);

}

}

class Algorithm

{

public <T> int count(List<T> list, SomeSpecialNumber<T> s)

{

int count = 0;

for (T type : list)

{

if (s.match(type))

{

++count;

}

}

return count;

}

}

public class Template

{

public void testSpecificElement()

{

List<Integer> list = Arrays.asList(1, 4, 5, 7, 8, 9, 13, 11);

PrimeNumber pn = new PrimeNumber();

EvenNumber en = new EvenNumber();

Algorithm a = new Algorithm();

System.out.println(" Count of prime numbers: " + a.count(list, pn));

System.out.println(" Count of even numbers: " + a.count(list, en));

}

public static void main(String[] args)

{

Template tq = new Template();

tq.testSpecificElement();

}

}

Output :

Count of prime numbers: 5

Count of even numbers: 2

**Experiment No: 08**

\*/

import java.io.\*;

import java.util.\*;

class Database

{

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

public void addRecords() throws IOException

{

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("sample.txt",true)));

String studentname, address,s;//declaration of studentname , address ,s

int studentid, rollno, Class;//declaration of studentid , rollno, Class

float marks;//declaration of marks

boolean addMore = false; //declaration of addmore

do {

System.out.print("\nEnter Student Name: "); //printing on console

studentname = br.readLine(); //taking input from user

System.out.print("Student Id: "); //printing on console

studentid = Integer.parseInt(br.readLine()); //taking input from user

System.out.print("Roll no: ");//printing on console

rollno = Integer.parseInt(br.readLine()); //taking input from user

System.out.print("Address: "); //printing on console

address = br.readLine(); //taking input from user

System.out.print("Class: ");//printing on console

Class = Integer.parseInt(br.readLine()); //taking input from user

System.out.print("Marks : "); //printing on console

marks = Float.parseFloat(br.readLine()); //taking input from user

pw.println(studentname+" "+studentid+" "+rollno+" "+address+" "+Class+" "+marks);

//appending data into to file

System.out.print("\nRecords added successfully !\n\nDo you want to add more records ? (y/n) : ");

s = br.readLine();//take input from user

if(s.equalsIgnoreCase("y")){

addMore = true;//modify addmore

System.out.println();

}

else

addMore = false; //modify addmore

}

while(addMore);

pw.close();

}

public void readRecords() throws IOException

{

try {

BufferedReader file = new BufferedReader(new FileReader("sample.txt"));

String name;

int i=1;

while((name = file.readLine()) != null) {

System.out.println(name);

System.out.println("");

} file.close();

}

catch(FileNotFoundException e)

{

System.out.println("\nERROR : File not Found !!!");

}

}

public void searchRecords() throws IOException

{

try {

BufferedReader file = new BufferedReader(new FileReader("sample.txt"));

String name;

int flag=0; //intizing value of flag=0

Scanner sc=new Scanner(System.in);

System.out.print("Enter an id of the student you want to search: ");

String searchname=sc.next();

while((name = file.readLine()) != null) {

String[] line = name.split(" ");

if(searchname.equalsIgnoreCase(line[1])){

System.out.println("Record found"); //printing on console

System.out.println(name); //printing record on console

System.out.println("");

flag=1; //modify value

break;

}

}

if(flag==0)

System.out.println("Record not found"); //printing on console

file.close();

}

catch(FileNotFoundException e) {//Exception handling

System.out.println("\nERROR : File not Found !!!");//printing on console

}

}

public void deleteRecords() throws IOException

{

try { // Open the file

BufferedReader file1 = new BufferedReader(new FileReader("sample.txt"));

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("new.txt",true)));

String name; //declaration of string name

int flag=0; //intizing value of flag=0

Scanner sc=new Scanner(System.in); //creating obj of scanner class

System.out.print("Enter the name of the student you want to delete: ");

String searchname=sc.next(); // Read records from the file

while((name = file1.readLine()) != null) {

String[] line = name.split(" ");

if(!searchname.equalsIgnoreCase(line[0])){

pw.println(name);

flag=0; //modify value

}

else{

System.out.println("Record found"); //printing on console

flag=1;//modify value

}

} file1.close();//closing file

pw.close();

File delName = new File("sample.txt");//creating obj of sample.txt

File oldName = new File("new.txt"); //creating obj of new.txt

File newName = new File("sample.txt"); //creating obj of sample.txt

if(delName.delete())

System.out.println("deleted successfully"); //printing on console

else

System.out.println("Error");//printing on console

if (oldName.renameTo(newName))

System.out.println("Renamed successfully"); //printing on console

else

System.out.println("Error"); //printing on console

}

catch(FileNotFoundException e) {//Exception handling

System.out.println("\nERROR : File not Found !!!");

}

}

public void updateRecords() throws IOException

{

try {

BufferedReader file1 = new BufferedReader(new FileReader("sample.txt"));

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("new.txt",true)));

String name;//declaration of string name

int flag=0; //intizing flag to 0

Scanner sc=new Scanner(System.in); //creating obje of scanner class

System.out.print("Enter the name of the student you want to update: "); //printing on console

String searchname=sc.next(); // Read records from the file

while((name = file1.readLine()) != null) { //check condition

String[] line = name.split(" ");

if(!searchname.equalsIgnoreCase(line[0])){ //check condition

pw.println(name);

flag=0; //modify value of flag

}

else

{

System.out.println("Record found"); //printing on console

System.out.print("Enter updated marks: "); //printing on console

String up\_mark=sc.next(); //taking input from user

pw.println(line[0]+" "+line[1]+" "+line[2]+" "+line[3]+" "+line[4]+" "+up\_mark);

flag=1; //modify value of flag

}

}

file1.close();

pw.close();

File delName = new File("sample.txt");

File oldName = new File("new.txt");

File newName = new File("sample.txt");

if(delName.delete())

System.out.println("record updated successfully");

else

System.out.println("Error");

if (oldName.renameTo(newName))

System.out.println("Renamed successfully");

else

System.out.println("Error");

}

catch(FileNotFoundException e) {

System.out.println("\nERROR : File not Found !!!"); //printing on console

}

}

public void clear(String filename) throws IOException

{

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(filename)));

pw.close(); //closing PrintWriter object

System.out.println("\nAll Records cleared successfully !");

//printing on console

}

}

public class FileHandlingMain

{

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

{

Database f = new Database(); //creating obj of Database class

Scanner sc =new Scanner(System.in);//creating object of scanner class

System.out.println("");

while(true) {

//menu driven

System.out.print("1. Add Records\n2. Display Records\n3. Clear All Records\n4. Search Records"

+ "\n5. Delete Records\n6. Update Records \n7. Exit\n\nEnter your choice : ");

int choice = sc.nextInt();//taking input from user

System.out.println("");

//switch Case

switch(choice)

{

case 1: f.addRecords();

break;

case 2: f.readRecords();

break;

case 3: f.clear("sample.txt");

break;

case 4: f.searchRecords();

break;

case 5: f.deleteRecords();

break;

case 6: f.updateRecords();

break;

case 7: System.exit(0);

break;

default: System.out.println("\nInvalid Choice !");

break;

}

}

}

}

/\* Output

C:\ Java Porgrams>javac FileHandlingMain.java

C:\ Java Porgrams>java FileHandlingMain

1. Add Records

2. Display Records

3. Clear All Records

4. Search Records

5. Delete Records

6. Update Records

7. Exit

Enter your choice : 1

Enter Student Name: ABC

Student Id: 1

Roll no: 1

Address: Pune

Class: 1

Marks : 10

Records added successfully !

Do you want to add more records ? (y/n) : n

1. Add Records

2. Display Records

3. Clear All Records

4. Search Records

5. Delete Records

6. Update Records

7. Exit

Enter your choice : 2

ABC 1 1 Pune 1 10.0

1. Add Records

2. Display Records

3. Clear All Records

4. Search Records

5. Delete Records

6. Update Records

7. Exit

Enter your choice : 4

Enter an id of the student you want to search: 1

Record found

ABC 1 1 Pune 1 10.0

1. Add Records

2. Display Records

3. Clear All Records

4. Search Records

5. Delete Records

6. Update Records

7. Exit

Enter your choice : 5

Enter the name of the student you want to delete: 1

deleted successfully

1. Add Records

2. Display Records

3. Clear All Records

4. Search Records

5. Delete Records

6. Update Records

7. Exit

Enter your choice : 7

\*/

Program No: 9

Case Study:

Using concepts of Object Oriented programming develop solution for any one

application

1) Banking system having following operations :

1. Create an account 2. Deposit money 3. Withdraw money 4. Honor daily withdrawal

limit 5.

Check the balance 6. Display Account information.

2) Inventory management system having following operations :

1. List of all products 2. Display individual product information 3. Purchase 4.

Shipping 5.

Balance stock 6. Loss and Profit calculation.

Program

\*/

import java.util.Scanner;

class Customer {

private String customerName;

private int customerAge;

public void setCustomerName(String customerName){

this.customerName=customerName;

}

public String getCustomerName(){

return customerName;

}

public void setCustomerAge(int customerAge){

this.customerAge=customerAge;

}

public int getCustomerAge(){

return customerAge;

}

}

abstract class Account { //creating abstract class account

protected double balance; //declaration of balance

protected int accountId; //declaration of accountId

protected String accountType; //declaration of accountType

protected Customer custobj; //declaration of customer obj

void setBalance(double balance){

this.balance=balance;

}

double getBalance(){

return balance;}

void setAccountId(int accountId){

this.accountId=accountId;

}

int getAccountId(){

return accountId;

}

void setAccountType(String accountType){

this.accountType=accountType;

}

String getAccountType(){

return accountType; //returning value of accountType

}

void setCustomerObject(Customer custobj){

this.custobj=custobj;

}

Customer getCustomerObject(){

return custobj;

}

public abstract boolean withdraw(double amount);

}

//SAVING ACCOUNT CLASS

class SavingsAccount extends Account{

//inheriting Account class in SavingAccount

private double minimumBalance;

public void setMinimumBalance(double minimumBalance){

this.minimumBalance=minimumBalance;

//setting minimumBalance

}

public double getMinimumBalance(){

return minimumBalance;

}

public boolean withdraw(double amount){

//method to return true or false

if((balance-amount)>minimumBalance){

//check whether withdraw amount is greater than

minimumBalance

balance-=amount;

return true;

}

else

return false; //returning false}

}

//BANK CLASS

class Bank {

public static Scanner sc=new Scanner(System.in);

public SavingsAccount a=new SavingsAccount();

public Customer c=new Customer();

public SavingsAccount createAccount(){

sc.nextLine();

System.out.print("Enter your name: ");

String customername=sc.nextLine();

c.setCustomerName(customername);

System.out.print("Enter your age: ");

int customerage=sc.nextInt();

if(customerage<18){//check whether the age is less than 18

do{

System.out.print("Minimum age should be 18 to create an

account.\nPlease enter valid age: ");

customerage=sc.nextInt();

}while(customerage<18); //if age is less than 18

}

c.setCustomerAge(customerage); //calling setCustomerName method

a.setCustomerObject(c);//calling setCustomerName method

System.out.print("Enter your account Id: ");

int accountid=sc.nextInt();

a.setAccountId(accountid);

System.out.print("Enter your account type: ");

String accounttype=sc.next();

a.setAccountType(accounttype);

System.out.print("Enter balance: ");

double balance=sc.nextDouble();//taking balance as input from user

a.setBalance(balance);//calling setBalance method

System.out.print("Enter minimum balance: ");

double minbalance=sc.nextDouble();

a.setMinimumBalance(minbalance);

return a;

}void getWithdrawAmount(){

System.out.print("Enter the amount you want to withdraw: ");

double amount=sc.nextDouble();

//taking amount as input from

user

if(amount>20000){ //check whether amount is greater than 20000

System.out.println("Withdrawal failed. Maximum limit of withdrawal in

one transaction is Rs.20000.");

}

else{ //if amount is less than 20000

if(a.withdraw(amount)==true){

System.out.println("Withdrawal successful. Balance is:

"+a.getBalance());

}

else

System.out.println("Sorry!!! Not enough balance"); //if not enough

balance

}

}

public void depositAmount(double amount){ //method to deposit Amount

double bal=a.getBalance()+amount;

//previous balance + amount

a.setBalance(bal); //call setBalance method

System.out.println("Amount deposited successfully. Balance is:

"+a.getBalance());

}

public void checkBalance(){

//method to check Balance

System.out.println("Balance is: "+a.getBalance());//calling

getbalance method

}

public void displayAccountInformation(){

//method to display Account

Information

System.out.println("Welcome "+c.getCustomerName()+"! Following are your

account details:");

//display name of customer

System.out.println("Age :"+c.getCustomerAge()); //display Age of

customer

System.out.println("Account Id: "+a.getAccountId());

//display

Account Id of customer

System.out.println("Account Type: "+a.getAccountType()); //display

Account Type of customer

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

//display Balance of

customer

System.out.println("Minimum balance: "+a.getMinimumBalance()); //display

Minimum balance of customer

}

}//MAIN CLASS

public class BankOperations{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

SavingsAccount a; //cresting object of SavingsAccount class

Bank bm=new Bank(); //cresting object of Bank class

do{

//menu driven program

System.out.println("\n\t1.Create Account\n\t2.Display

Account\n\t3.Check Balance"

+ "\n\t4.Deposit Amount\n\t5.Withdraw

Amount\n\t6.Exit");

System.out.print("Enter your choice: "); //printing on

console

int choice=sc.nextInt();

//taking input from user

System.out.println("");

switch(choice)

//switch case

{

case 1:

a=bm.createAccount();

System.out.println("=============================================

====");

break;

case 2:

bm.displayAccountInformation();

System.out.println("=============================================

====");

break;

case 3:

bm.checkBalance();

System.out.println("=============================================

====");

break;

case 4:

System.out.print("Enter the amount you want to deposit: ");

double amount=sc.nextDouble();

bm.depositAmount(amount);

System.out.println("=============================================

====");

break;

case 5:

bm.getWithdrawAmount();

System.out.println("=============================================

====");

break;

case 6:

System.out.println("=============================================

====");

return ; //stop execution of programdefault:

System.out.println("INVALID INPUT !!");//printing invalid input

System.out.println("=============================================

====");

break;

}

}while(true);

}

}

/\*OUTPUT:

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.Exit

Enter your choice: 1

Enter your name: Siddharth

Enter your age: 20

Enter your account Id: 3210213

Enter your account type: Saving

Enter balance: 500

Enter minimum balance: 100

=================================================

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.Exit

Enter your choice: 2

Welcome Siddharth! Following are your account details:

Age :20

Account Id: 3210213

Account Type: Saving

Balance: 500.0

Minimum balance: 100.0

=================================================

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.ExitEnter your choice: 3

Balance is: 500.0

=================================================

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.Exit

Enter your choice: 4

Enter the amount you want to deposit: 250

Amount deposited successfully. Balance is: 750.0

=================================================

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.Exit

Enter your choice: 5

Enter the amount you want to withdraw: 250

Withdrawal successful. Balance is: 500.0

=================================================

1.Create Account

2.Display Account

3.Check Balance

4.Deposit Amount

5.Withdraw Amount

6.Exit

Enter your choice: 6

Experiment no 10

Progragm : Design and implement Factory design pattern for the given context. Consider Car building

process, which requires many steps from allocating accessories to final makeup. These steps

should be written as methods and should be called while creating an instance of a specific car type.

Hatchback, Sedan, SUV could be the subclasses of Car class. Car class and its subclasses,

CarFactory and TestFactoryPattern should be implemented.

Experiment : 10

\*/

code :

enum CarType {

SMALL, SEDAN, LUXURY

}

abstract class Car {

public Car(CarType model) {

this.model = model;

arrangeParts();

}

private void arrangeParts() {

}

protected abstract void construct();

private CarType model = null;

public CarType getModel() {

return model;

}

public void setModel(CarType model) {

this.model = model;

}

}

class LuxuryCar extends Car {

LuxuryCar() {

super(CarType.LUXURY);

construct();

}

@Override

protected void construct() {

System.out.println("Building luxury car&quot");

}

}

class SmallCar extends Car {

SmallCar() {

super(CarType.SMALL);

construct();

}

@Override

protected void construct() {

System.out.println("Building small car");

}

}

class SedanCar extends Car {

SedanCar() {

super(CarType.SEDAN);

construct();

}

@Override

protected void construct() {

System.out.println("Building sedan car");

// add accessories

}

}

class CarFactory {

public static Car buildCar(CarType model) {

Car car = null;

switch (model) {

case SMALL:

car = new SmallCar();

break;

case SEDAN:

car = new SedanCar();

break;

case LUXURY:

car = new LuxuryCar();

break;

default:

// throw some exception

break;

}

return car;

}

}

class TestFactoryPattern {

public static void main(String[] args) {

System.out.println(CarFactory.buildCar(CarType.SMALL));

System.out.println(CarFactory.buildCar(CarType.SEDAN));

System.out.println(CarFactory.buildCar(CarType.LUXURY));

}

}

output :

Building small car

SmallCar@4617c264

Building sedan car

SedanCar@7a81197d

Building luxury car&quot

LuxuryCar@24d46ca6

**Experiment : 10**

\*/

code :

interface Currency {

String getSymbol();

}

class Rupee implements Currency {

@Override

public String getSymbol() {

return "Rs";

}

}

class SGDDollar implements Currency {

@Override

public String getSymbol() {

return "SGD";

}

}

class USDollar implements Currency {

@Override

public String getSymbol() {

return "USD";

}

}

class CurrencyFactory {

public static Currency createCurrency (String country) {

if (country. equalsIgnoreCase ("India")){

return new Rupee();

}else if(country. equalsIgnoreCase ("Singapore")){

return new SGDDollar();

}else if(country. equalsIgnoreCase ("US")){

return new USDollar();

}

throw new IllegalArgumentException("No such currency");

}

}

public class Factory {

public static void main(String args[]) {

String country = args[0];

Currency rupee = CurrencyFactory.createCurrency(country);

System.out.println(rupee.getSymbol());

}

}

output :

Rs

**Experiment No:-11**

Program On:Implement and apply Strategy Design pattern for simple Shopping Cart where three payment strategies are used such as Credit Card, PayPal, Bit Coin. Create an interface for strategy pattern and give concrete implementation for payment.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.List;

interface PaymentStrategy {

public void pay(int amount);

}

class ShoppingCart {

//List of items

List<Item> items;

public ShoppingCart(){

this.items=new ArrayList<Item>();

}

public void addItem(Item item){

this.items.add(item);

}

public void removeItem(Item item){

this.items.remove(item);

}

public int calculateTotal(){

int sum = 0;

for(Item item : items){

sum += item.getPrice();

}

return sum;

}

public void pay(PaymentStrategy paymentMethod){

int amount = calculateTotal();

paymentMethod.pay(amount);

}

}

class Item {

private String upcCode;

private int price;

public Item(String upc, int cost){

this.upcCode=upc;

this.price=cost;

}

public String getUpcCode() {

return upcCode;

}

public int getPrice() {

return price;

}

}

class PaypalStrategy implements PaymentStrategy {

private String emailId;

private String password;

public PaypalStrategy(String email, String pwd){

this.emailId=email;

this.password=pwd;

}

@Override

public void pay(int amount) {

System.out.println(amount + " paid using Paypal.");

}

}

class CreditCardStrategy implements PaymentStrategy {

private String name;

private String cardNumber;

private String cvv;

private String dateOfExpiry;

public CreditCardStrategy(String nm, String ccNum, String cvv, String expiryDate){

this.name=nm;

this.cardNumber=ccNum;

this.cvv=cvv;

this.dateOfExpiry=expiryDate;

}

@Override

public void pay(int amount) {

System.out.println(amount +" paid with credit/debit card");

}

}

public class ShoppingCartTest {

public static void main(String[] args) {

ShoppingCart cart = new ShoppingCart();

Item item1 = new Item("1234",60);

Item item2 = new Item("5678",40);

cart.addItem(item1);

cart.addItem(item2);

//pay by paypal

cart.pay(new PaypalStrategy("myemail@example.com", "pwd"));

//pay by credit card

cart.pay(new CreditCardStrategy("Aparna Bindage", "1234567890123456", "786", "12/15"));

}

}

/\*

OUTPUT:

100 paid using Paypal.

100 paid with credit/debit card

\*/

**13.** Define a class student with four data members such as name, roll no, sub1 and sub2. Define appropriate methods to INITITALIZE AND DISPLAY VALUES OF DATA MEMBERS also calculate total marks and percentage scored by student in three subjects.

**import** java.lang.\*;

**import** java.io.\*;

**class** **Student** {

String name;

**int** roll\_no;

**int** sub1,sub2;

**void** **getdata**() **throws** IOException {

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

System.out.println ("Enter Name of Student");

name = br.readLine();

System.out.println ("Enter Roll No. of Student");

roll\_no = Integer.parseInt(br.readLine());

System.out.println ("Enter marks out of 100 of 1st subject");

sub1 = Integer.parseInt(br.readLine());

System.out.println ("Enter marks out of 100 of 2nd subject");

sub2 = Integer.parseInt(br.readLine());

}

**void** **show**() {

**int** total = sub1+sub2;

**float** per = (total \* 100) / 200;

System.out.println ("Roll No. = "+roll\_no);

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

System.out.println ("Marks of 1st Subject = "+sub1);

System.out.println ("Marks of 2nd Subject = "+sub2);

System.out.println ("Total Marks = "+total);

System.out.println ("Percentage = "+per+"%");

}

}

**public** **class** **StudentDemo** {

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

Student s=**new** Student();

s.getdata();

s.show();

}

}

**OUTPUT**

C:\>javac StudentDemo.java

C:\>java StudentDemo

Enter Name of Student

Subhash Patel

Enter Roll No. of Student

13

Enter marks out of 100 of 1st subject

56

Enter marks out of 100 of 2nd subject

78

Roll No. = 13

Name = Subhash Patel

Marks of 1st Subject = 56

Marks of 2nd Subject = 78

Total Marks = 134

Percentage = 67.0%

14.linked list

//linked list

import java.util.\*;

// Main class

public class Linkedlistcrude {

public static void main(String args[])

{

// Creating object of the

// class linked list

LinkedList<String> ll = new LinkedList<String>();

// Adding elements to the linked list

ll.add("A");

ll.add("B");

ll.addLast("C");

ll.addFirst("D");

ll.add(2, "E");

System.out.println(ll);

ll.remove("B");

ll.remove(3);

ll.removeFirst();

ll.removeLast();

System.out.println(ll);

System.out.println(ll.size());

}

}

// Linked list operations in Java

class LinkedList {

Node head;

// Create a node

class Node {

int data;

Node next;

Node(int d) {

data = d;

next = null;

}

}

// Insert at the beginning

public void insertAtBeginning(int new\_data) {

// insert the data

Node new\_node = new Node(new\_data);

new\_node.next = head;

head = new\_node;

}

// Insert after a node

public void insertAfter(Node prev\_node, int new\_data) {

if (prev\_node == null) {

System.out.println("The given previous node cannot be null");

return;

}

Node new\_node = new Node(new\_data);

new\_node.next = prev\_node.next;

prev\_node.next = new\_node;

}

// Insert at the end

public void insertAtEnd(int new\_data) {

Node new\_node = new Node(new\_data);

if (head == null) {

head = new Node(new\_data);

return;

}

new\_node.next = null;

Node last = head;

while (last.next != null)

last = last.next;

last.next = new\_node;

return;

}

// Delete a node

void deleteNode(int position) {

if (head == null)

return;

Node temp = head;

if (position == 0) {

head = temp.next;

return;

}

// Find the key to be deleted

for (int i = 0; temp != null && i < position - 1; i++)

temp = temp.next;

// If the key is not present

if (temp == null || temp.next == null)

return;

// Remove the node

Node next = temp.next.next;

temp.next = next;

}

Arraylist

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

ArrayList<String> animals = new ArrayList<>();

// add elements in the array list

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

System.out.println("ArrayList: " + animals);

// remove element from index 2

String str = animals.remove(2);

System.out.println("Updated ArrayList: " + animals);

System.out.println("Removed Element: " + str);

// checks if the ArrayList is empty

boolean result = animals.isEmpty(); // false

System.out.println("Is the ArrayList empty? " + result);

}

}

16. multilevel

import java.lang.\*;

class Account {

String cust\_name;

int acc\_no;

Account(String a, int b) {

cust\_name=a;

acc\_no=b;

}

void display() {

System.out.println ("Customer Name: "+cust\_name);

System.out.println ("Account No: "+acc\_no);

}

}

class Saving\_Acc extends Account {

int min\_bal,saving\_bal;

Saving\_Acc(String a, int b, int c, int d) {

super(a,b);

min\_bal=c;

saving\_bal=d;

}

void display() {

super.display();

System.out.println ("Minimum Balance: "+min\_bal);

System.out.println ("Saving Balance: "+saving\_bal);

}

}

class Acct\_Details extends Saving\_Acc {

int deposits, withdrawals;

Acct\_Details(String a, int b, int c, int d, int e, int f) {

super(a,b,c,d);

deposits=e;

withdrawals=f;

}

void display() {

super.display();

System.out.println ("Deposit: "+deposits);

System.out.println ("Withdrawals: "+withdrawals);

}

}

class MultiLevelInh {

public static void main(String[] args) {

Acct\_Details A = new Acct\_Details("Pankaj",666,1000,5000,500,9000);

A.display();

}

}

18. Array of object

class GFG {

public static void main(String args[])

{

// Declaring an array of student

Student[] arr;

// Allocating memory for 2 objects

// of type student

arr = new Student[2];

// Initializing the first element

// of the array

arr[0] = new Student( "Satyabrata");

// Initializing the second element

// of the array

arr[1] = new Student("Omm Prasad");

// Displaying the student data

System.out.println(

"Student data in student arr 0: ");

arr[0].display();

System.out.println(

"Student data in student arr 1: ");

arr[1].display();

}

}

// Creating a student class with

// id and name as a attributes

class Student {

public String name;

// Student class constructor

Student(String name)

{

this.name = name;

}

// display() method to display

// the student data

public void display()

{

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

System.out.println();

}

}

19 method overloading

public class Add{

// Overloaded sum(). This sum takes two int parameters

public int Add(int x, int y)

{

return (x + y);

}

// Overloaded sum(). This sum takes three int parameters

public int Add(int x, int y, int z)

{

return (x + y + z);

}

// Overloaded sum(). This sum takes two double

// parameters

public double sum(double x, double y)

{

return (x + y);

}

// Driver code

public static void main(String args[])

{

Sum s = new Sum();

System.out.println(s.sum(10, 20));

System.out.println(s.sum(10, 20, 30));

System.out.println(s.sum(10.5, 20.5));

}

}

20 bank

//Java Program to demonstrate the real scenario of Java Method Overriding

//where three classes are overriding the method of a parent class.

//Creating a parent class.

class Bank{

int getRateOfInterest(){return 0;}

}

//Creating child classes.

class SBI extends Bank{

int getRateOfInterest(){return 8;}

}

class ICICI extends Bank{

int getRateOfInterest(){return 7;}

}

class AXIS extends Bank{

int getRateOfInterest(){return 9;}

}

//Test class to create objects and call the methods

class Test2{

public static void main(String args[]){

SBI s=new SBI();

ICICI i=new ICICI();

AXIS a=new AXIS();

System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());

}

}

Getersetter

/ Java Program to Illustrate Getter and Setter

// Importing input output classes

import java.io.\*;

// Class 1

// Helper class

class GetSet {

    // Member variable of this class

    private String name;

    // Method 1 - Getter

    public String getName() { return name; }

    // Method 2 - Setter

    public void setName(String N)

    {

        // This keyword refers to current instance itself

        this.name = N;

    }

}

// Class 2

// Main class

class GFG {

    // Main driver method

    public static void main(String[] args)

    {

        // Creating an object of class 1 in main() method

        GetSet obj = new GetSet();

        // Setting the name by calling setter method

        obj.setName("Geeks for Geeks");

        // Getting the name by calling getter method

        System.out.println(obj.getName());

    }

}