***Name :***- Girishkumar Purushottam Naranje

***Roll No.*** :- 2051

***PRN No.*** :- 200941281049

**Assignments**

**OOPs with JAVA**

***Day 1*** :-

***Que*** .

Create Emp based organization structure --- Emp , Mgr , Worker

All of above classes must be in --com.app.org

2.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments.

Methods ---1.To get emp details -- override toString((inherited from Object class)

2.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;})

2.2 Mgr state ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----1. get mgr details : override toString (inherited from Emp class)

2. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

3. get performance bonus. --add a new method to return bonus.(getter)

2.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1. get worker details -- : override toString. (inherited from Emp class)

2. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) –override computeNetSalary

3. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)

Organize classes in inheritance hierarchy.

Write TestOrganization in "tester" package.

Create suitable array to store organization details.

Provide following options

1. Hire Manager

I/P : all manager details

2. Hire Worker

I/P : all worker details

3. Display information of all employees(toString) & display net salary (by invoking computeNetSal),

4. Exit

NOTE : Check boundary conditions properly.

***Ans :-***

**1.Employee Class**

**package** com.app.org;

**public** **class** Employee {

//Emp state--- id(int),name,deptId, basicSalary(double)

**private** **int** id;

**private** String name;

**private** **int** deptId;

**private** **double** basicSalary;

//constructor

**public** Employee(**int** id, String name, **int** deptId, **double** basicSalary) {

**this**.id=id;

**this**.name=name;

**this**.deptId=deptId;

**this**.basicSalary=basicSalary;

}

//net Salary

**public** **double** computeNetSalary() {

**return** 0;

}

//Details return string

**public** String toString() {

**return** "id:- "+id+"\nname: "+name+"\ndeptId:- "+deptId+"\nbasicSalary:- "+getBasicSalary();

}

**public** **double** getBasicSalary() {

**return** basicSalary;

}}

**2.Manager Class**

**package** com.app.org;

**public** **class** Manager **extends** Employee {

//Mgr state ---id,name,basic,deptId , perfBonus

**private** **double** perfBonus;

**public** Manager(**int** id, String name, **int** deptId, **double** basicSalary, **double** perfBonus) {

**super**(id, name, deptId, basicSalary);

**this**.perfBonus=perfBonus;

}

//compute net salary (formula: basic+perfBonus) -- override computeNetSalary

**public** **double** computeNetSalary() {

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

**return** getBasicSalary()+perfBonus;

}

// overriding version in Manager

**public** String toString() {

**return** "Manager:- \n"+**super**.toString()+"\nPerformance Bonus:- "+perfBonus+ "\nNet Salary:- "+ computeNetSalary()+"\n";

}}

**3.Worker Class**

**package** com.app.org;

**public** **class** Worker **extends** Employee {

//Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

**private** **double** hourWorked;

**private** **double** hourlyRate;

**public** Worker(**int** id, String name, **int** empId, **double** basicSalary,**double** hourWorked,**double** hourlyRate) {

**super**(id, name, empId, basicSalary);

**this**.hourWorked=hourWorked;

**this**.hourlyRate=hourlyRate;

}

//compute net salary (formula:= basic+(hoursWorked\*hourlyRate) --override computeNetSalary

**public** **double** computeNetSalary() {

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

**return** getBasicSalary()+(hourWorked\*hourlyRate);

}

//overriding in Worker

**public** String toString() {

**return** "Worker:- \n"+**super**.toString()+"\nhourWorked:- "+hourWorked+"\nhourlyRate:- "+hourlyRate+ "\nNet Salary :- "+computeNetSalary()+"\n";

}}

**4.Tester Class**

**package** tester;

**import** java.util.Scanner;

**import** com.app.org.Employee;

**import** com.app.org.Manager;

**import** com.app.org.Worker;

public class **public** **class** TestOrganization {

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

Scanner sc=**new** Scanner(System.***in***);

**boolean** exit=**false**;

System.***out***.print("How many Emplyees Requires :- ");

Employee[] empre=**new** Employee[sc.nextInt()];

**int** counter = 0;

**while**(!exit) {

System.***out***.println("1.Hire Manager 2.Hire Worker 3.Display information of all employees 4.Exit");

System.***out***.print("Choose Option :-");

**switch**(sc.nextInt()) {

**case** 1://Manager

**if**(counter<empre.length) {

System.***out***.println("Managers Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.PerformanceBonus ");

empre[counter]=**new** Manager(sc.nextInt(),sc.next(),sc.nextInt(),sc.nextDouble(),

sc.nextDouble());

counter++;

}

**else**

System.***out***.println("No Vacancy");

**break**;

**case** 2://Worker

**if**(counter<empre.length) {

System.***out***.println("Worker Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.hourWorked 6.hourlyRate");

empre[counter]=**new** Worker(sc.nextInt(),sc.next(),sc.nextInt(),sc.nextDouble(),

sc.nextDouble(), sc.nextDouble());

counter++;

}

**else**

System.***out***.println("No Vacancy");

**break**;

**case** 3://Display All details

System.***out***.println("\nDetails of All Employee ");

**for**(**int** i=0;i<counter;i++) {

System.***out***.println(empre[i]);

}

**break**;

**case** 4: exit=**true**;

**break**;

} }

sc.close();

}}

***Output :-***

How many Emplyees Requires :- 2

1.Hire Manager 2.Hire Worker 3.Display information of all employees 4.Exit

Choose Option :-1

Managers Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.PerformanceBonus

001 Girish 100 25000 500

1.Hire Manager 2.Hire Worker 3.Display information of all employees 4.Exit

Choose Option :-2

Worker Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.hourWorked 6.hourlyRate

002 Kiran 101 20000 2

300

1.Hire Manager 2.Hire Worker 3.Display information of all employees 4.Exit

Choose Option :-3

Details of All Employee

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

Manager:-

id:- 1

name:- Girish

deptId:- 100

basicSalary:- 25000.0

Performance Bonus:- 500.0

Net Salary:- 25500.0

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

Worker:-

id:- 2

name:- Kiran

deptId:- 101

basicSalary:- 20000.0

hourWorked:- 2.0

hourlyRate:- 300.0

Net Salary :- 20600.0

***Day 2*** :-

***Que*** .

1. Import day2.1 eclipse project in the lab.

2. Solve code samples

day2\_data\day2\_help\covariance\_demo\CovariantReturn.java

What will happen ?

Compile time err

Run time err

O/P ????

3. Solve code samples

day2\_data\day2\_help\method overloading

What will happen ?

Compile time err

Run time err

O/P ????

4. Solve : Optional BUT recommended assignment

Add another option

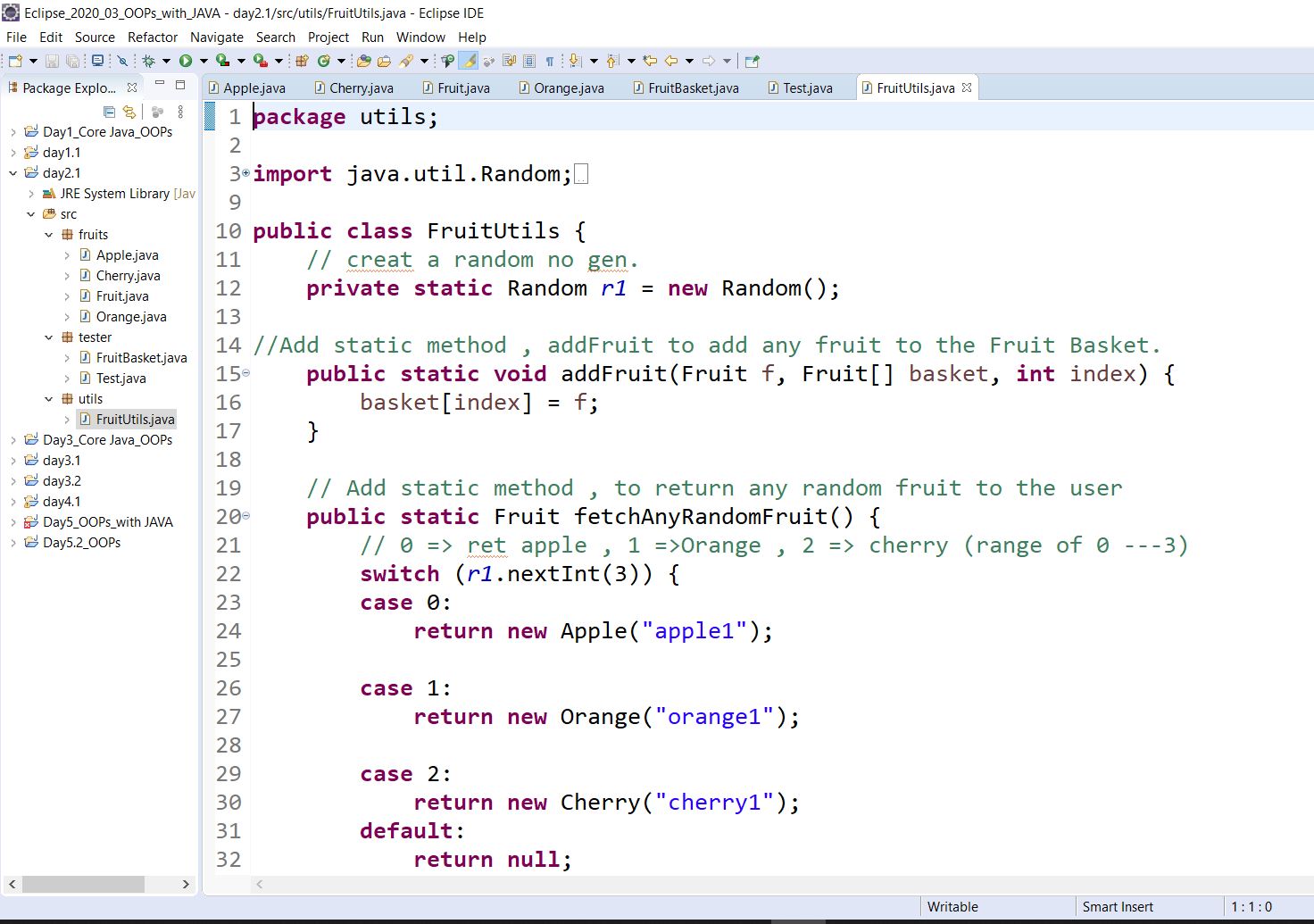
4 : Display emp specific details (i.e if its a manager display it's performance bonus or if it's worker display it's hourly rate)

I/P : emp id

O/P : bonus or rate or error message "invalid emp id"

***Ans :-***

**1**. Import day2.1 eclipse project in the lab



**2**. day2\_data\day2\_help\covariance\_demo\CovariantReturn.java

:- Compile time err

Error is **:-** The public type CovariantReturn must be defined in its own file.

**3**. day2\_data\day2\_help\method overloading

3.1 : EasyOver : O/P : long

3.2 : Test3 : Compile time err

Error is **:-** The method test(int, double) is ambiguous for the type Test3

3.3 : TestOverload : O/P : in string version

3.4 : TestOverload2 : O/P : in double version

**4.** Code

1.Employee class

**package** com.app.org;

**public** **class** Employee {

//Emp state--- id(int), name, deptId , basicSalary(double)

**private** **int** id;

**private** String name;

**private** **int** deptId;

**private** **double** basicSalary;

//constructor

**public** Employee(**int** id, String name, **int** deptId, **double** basicSalary) {

//super();

**this**.setId(id);

**this**.name=name;

**this**.deptId=deptId;

**this**.basicSalary=basicSalary;

}

//net Salary

**public** **double** computeNetSalary() {

**return** 0;

}

//Details return string

**public** String toString() {

**return** "id:- "+getId()+"\nname:- "+name+"\ndeptId:- "+deptId+"\nbasicSalary:- "+getBasicSalary();

}

**public** **double** getBasicSalary() {

**return** basicSalary;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}}

2.Manager Class

**package** com.app.org;

**public** **class** Manager **extends** Employee {

//Mgr state ---id,name,basic,deptId , perfBonus

**private** **double** perfBonus;

**public** Manager(**int** id, String name, **int** deptId, **double** basicSalary, **double** perfBonus ) {

**super**(id, name, deptId, basicSalary);

**this**.perfBonus=perfBonus;

}

//compute net salary (formula: basic+perfBonus) -- override computeNetSalary

**public** **double** computeNetSalary() {

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

**return** getBasicSalary()+perfBonus;

}

// overriding version in Manager

**public** String toString() {

**return** "Manager:- \n"+**super**.toString()+"\nPerformance Bonus:- "+perfBonus+ "\nNet Salary:- "+ computeNetSalary()+"\n";

}

**public** **double** getPerfBonus() {

**return** perfBonus;

}}

3.Worker Class

**package** com.app.org;

**public** **class** Worker **extends** Employee {

//Worker state--id,name,basic,deptId,hoursWorked,hourlyRate

**private** **double** hourWorked;

**private** **double** hourlyRate;

**public** Worker(**int** id, String name, **int** empId, **double** basicSalary, **double** hourWorked, **double** hourlyRate)

{

**super**(id, name, empId, basicSalary);

**this**.hourWorked=hourWorked;

**this**.hourlyRate=hourlyRate;

}

//compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

**public** **double** computeNetSalary(){

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

**return** getBasicSalary()+(hourWorked\*hourlyRate);

}

//overriding in Worker

**public** String toString(){

**return** "Worker:- \n"+**super**.toString()+"\nhourWorked:- "+hourWorked+"\nhourlyRate:- "+hourlyRate+ "\nNet Salary :- "+computeNetSalary()+"\n";

}

**public** **double** getHourlyRate() {

**return** hourlyRate;

}}

4.TestOrganization Class

**package** tester;

**import** java.util.Scanner;

**import** com.app.org.Employee;

**import** com.app.org.Manager;

**import** com.app.org.Worker;

**public** **class** TestOrganization {

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

Scanner sc=**new** Scanner(System.***in***);

**boolean** exit=**false**;

System.***out***.print("How many Emplyees Requires :- ");

Employee[] empre=**new** Employee[sc.nextInt()];

**int** counter = 0;

**while**(!exit) {

System.***out***.println("1.Hire Manager \n2.Hire Worker \n3.Display information of all employees \n4.Display emp specific details(Enter Employee id) \n5.Exit");

System.***out***.print("Choose Option :-");

**switch**(sc.nextInt()) {

**case** 1://Manager

**if**(counter<empre.length) {

System.***out***.println("Managers Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.PerformanceBonus ");

empre[counter]=**new** Manager(sc.nextInt(),sc.next(),sc.nextInt(),sc.nextDouble(),

sc.nextDouble());

counter++;

}

**else**

System.***out***.println("No Vacancy");

**break**;

**case** 2://Worker

**if**(counter<empre.length) {

System.***out***.println("Worker Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.hourWorked 6.hourlyRate");

empre[counter]=**new** Worker(sc.nextInt(),sc.next(),sc.nextInt(),sc.nextDouble(),

sc.nextDouble(),sc.nextDouble());

counter++;

}

**else**

System.***out***.println("No Vacancy");

**break**;

**case** 3://Display All details

System.***out***.println("\nDetails of All Employee ");

**for**(**int** i=0;i<counter;i++) {

System.***out***.println(empre[i]);

}

**break**;

**case** 4: System.***out***.println("Enter the iD of Emp");

**int** id2=sc.nextInt();

**for**(**int** i=0;i<counter;i++){

**if**(empre[i].getId()==id2) { //its req public access of id

**if**(empre[i] **instanceof** Manager) {

System.***out***.println("its is Maneger");

System.***out***.println("PerBonus="+((Manager)empre[i]).getPerfBonus()+"\n");

}

**else** **if**(empre[i] **instanceof** Worker) {

System.***out***.println("its is Worker");

System.***out***.println("Hourly Rate="+((Worker)empre[i]).getHourlyRate()+"\n");

}

**else** {

System.***out***.println("Enter valid Id ");

}}}

**break**;

**case** 5: System.***out***.println("NO Vacancy Now");

exit=**true**;

**break**;

}}

sc.close();

}}

***Output :-***

How many Emplyees Requires :- 2

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-1

Managers Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.PerformanceBonus

001 Girish 01 30000 5000

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-2

Worker Details: 1.ID 2.Name 3.EmpId 4.BasicSalary 5.hourWorked 6.hourlyRate

003 Kishan 03 20000 5 500

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-3

Details of All Employee

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

Manager:-

id:- 1

name:- Girish

deptId:- 1

basicSalary:- 30000.0

Performance Bonus:- 5000.0

Net Salary:- 35000.0

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

Worker:-

id:- 3

name:- Kishan

deptId:- 3

basicSalary:- 20000.0

hourWorked:- 5.0

hourlyRate:- 500.0

Net Salary :- 22500.0

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-4

Enter the iD of Emp

001

its is Maneger

PerBonus=5000.0

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-4

Enter the iD of Emp

003

its is Worker

Hourly Rate=500.0

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-4

Enter the iD of Emp

005

Enter valid Id

1.Hire Manager

2.Hire Worker

3.Display information of all employees

4.Display emp specific details(Enter Employee id)

5.Exit

Choose Option :-

***Day 3*** :-

***Que*** .

1. Import day3.1 in your workspace & revise abstract classes.

2. Import day3.2 & revise interfaces.

3. Solve assignment (to understand abstraction)

1. Create abstract class Shape --state : x,y

Abstract Method --public double area();

public String toString() : to ret x & y

Why will area() be abstract in Shape class ?????????

2. Circle -- x,y,radius

Concrete overriding Method --public double area() : ret area of circle

public String toString() : ret x, y & radius

3. Rectangle -- x,y,w,h

Concrete overriding Method --public double area() : ret area of rectangle

public String toString() : ret x, y , width & height

4. Square-- x,y,side

Concrete overriding Method --public double area() : ret area of square

public String toString() : ret x, y , side

5. Create a ShapeFactory class

Add a static method(generateShape) to return randomly generated shape.

Hint : random no generator

6. Create a Tester . Invoke ShapeFactory's generateShape() method , in a for-loop (5 times)

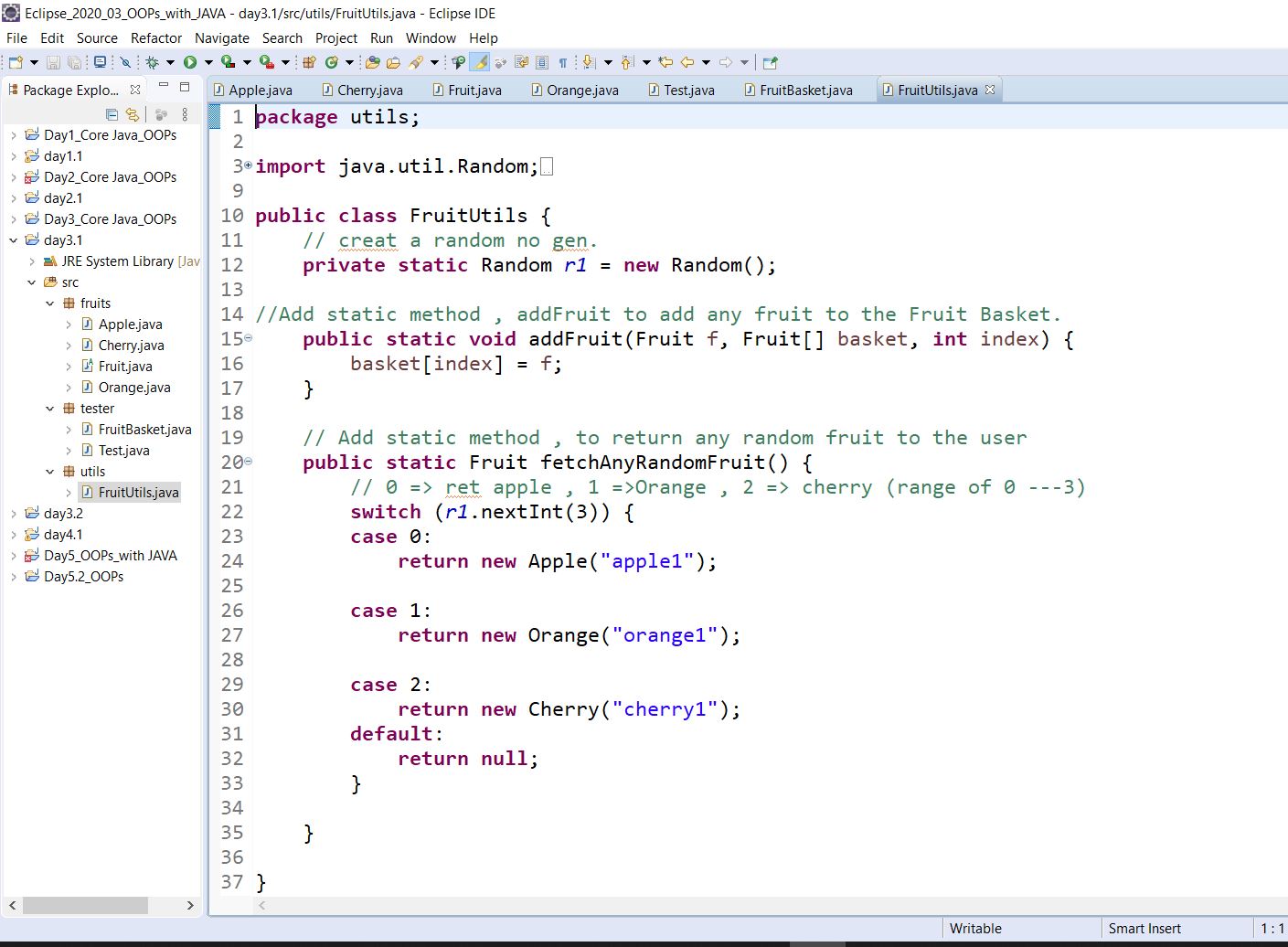
to display details & area of each shape.

***Ans :-***

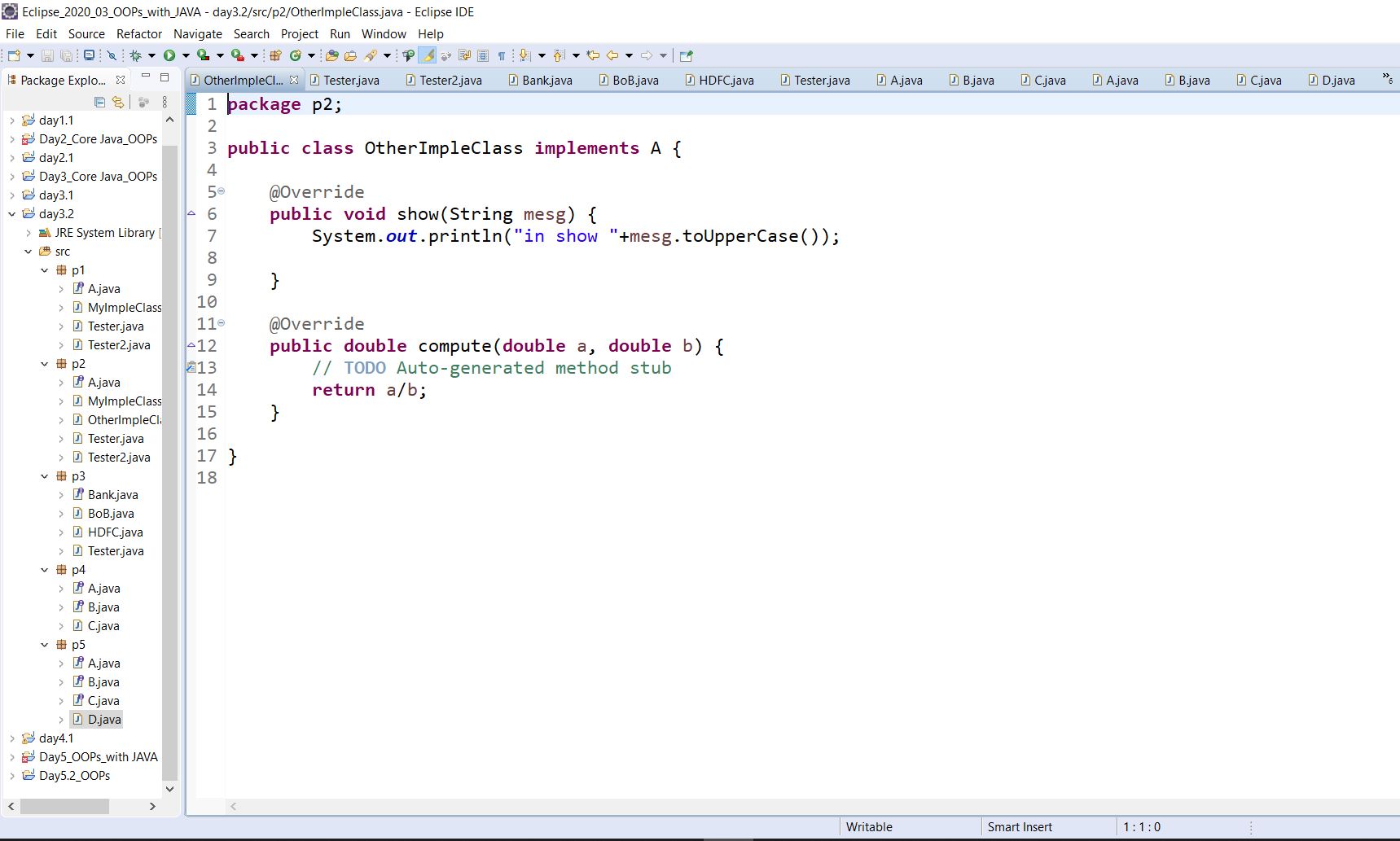
*Que.* Why will area() be abstract in Shape class ?

*Ans:*- Because area() is same factor in all 3 shape: Circle,Rectangular,Square. So it is override in Circle,Rectangle,Square.

**1.** Import day3.1 in your workspace



2.Import day3.2



3

3.1 Shape Class

**package** com.shape.abstrct;

//Abstract super class

**public** **abstract** **class** Shape {

//State x and y

**private** **double** x,y;

//parameterize Constructor

**public** Shape(**double** x, **double** y){

**this**.x=x;

**this**.y=y;

}

//Abstract Method area()

**public** **abstract** **double** area();

//Return state of x and y

**public** String toString(){

**return** "Shape:- x="+x+" y="+y;

}}

3.2 Circle Class

**package** com.shape.abstrct;

//Extends class

**public** **class** Circle **extends** Shape {

**private** **double** radius;

//Constructor

**public** Circle(**double** x, **double** y, **double** radius){

**super**(x,y);

**this**.radius=radius;

}

@Override

//Abstract Method area()

**public** **double** area() {

**return** ((22.0/7.0)\*radius\*radius);

}

@Override

**public** String toString()

{

**return** "Circle = "+**super**.toString()+" radius = "+radius;

}}

3.3 Rectangle Circle

**package** com.shape.abstrct;

**public** **class** Rectangle **extends** Shape {

**private** **double** w,h;

//Constructor

**public** Rectangle(**double** x, **double** y, **double** w, **double** h){

**super**(x,y);

**this**.w=w;

**this**.h=h;

}

@Override

//Abstract Method return area() -- concreate override method

**public** **double** area() {

**return** (w\*h);

}

@Override

**public** String toString()

{

**return** "Rectangle = "+**super**.toString()+" width = "+w+" height = "+h;

}}

3.4 Square Class

**package** com.shape.abstrct;

//entends sub class

**public** **class** Square **extends** Shape {

**private** **int** side;

//Constructor

**public** Square(**double** x, **double** y, **int** side){

**super**(x,y);

**this**.side=side;

}

//Concrete overriding Method --public double area() : ret area of square

**public** **double** area() {

**return** (side\*side);

}

**public** String toString() {

**return** "Square :- "+**super**.toString()+" Side = "+side ;

}}

3.5 ShapeFactory Class

**package** com.shape.abstrct;

**import** java.util.Random;

**public** **class** ShapeFactory {

**public** **static** Random *r2*= **new** Random();

**public** **static** Shape generateShape() {

**switch** (*r2*.nextInt(3)) {

**case** 0:

**return** **new** Circle(23.0, 55.3, 20.0);

**case** 1:

**return** **new** Rectangle(23.0, 55.3, 20.0, 10.0);

**case** 2:

**return** **new** Square(23.0, 55.3, 30);

**default** :

**return** **null**;

}}}

3.6 Tester Class

**package** com.shape.abstrct;

//import com.shape.abstrct.ShapeFactory.\*;

**public** **class** TesterShape {

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

Shape s;

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

s = ShapeFactory.*generateShape*();

//s=generateShape();

**if**(s **instanceof** Circle)

{

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

System.***out***.println("Area = "+s.area()+"\n");

}

**else** **if**(s **instanceof** Rectangle)

{

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

System.***out***.println("Area = "+s.area()+"\n");

}

**else** **if**(s **instanceof** Square)

{

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

System.***out***.println("Area = "+s.area()+"\n");

}}}}

**Output :-**

Rectangle = Shape:- x=23.0 y=55.3 width = 20.0 height = 10.0

Area = 200.0

Square :- Shape:- x=23.0 y=55.3 Side = 30

Area = 900.0

Square :- Shape:- x=23.0 y=55.3 Side = 30

Area = 900.0

Circle = Shape:- x=23.0 y=55.3 radius = 20.0

Area = 1257.142857142857

Circle = Shape:- x=23.0 y=55.3 radius = 20.0

Area = 1257.142857142857

***Day 4*** :-

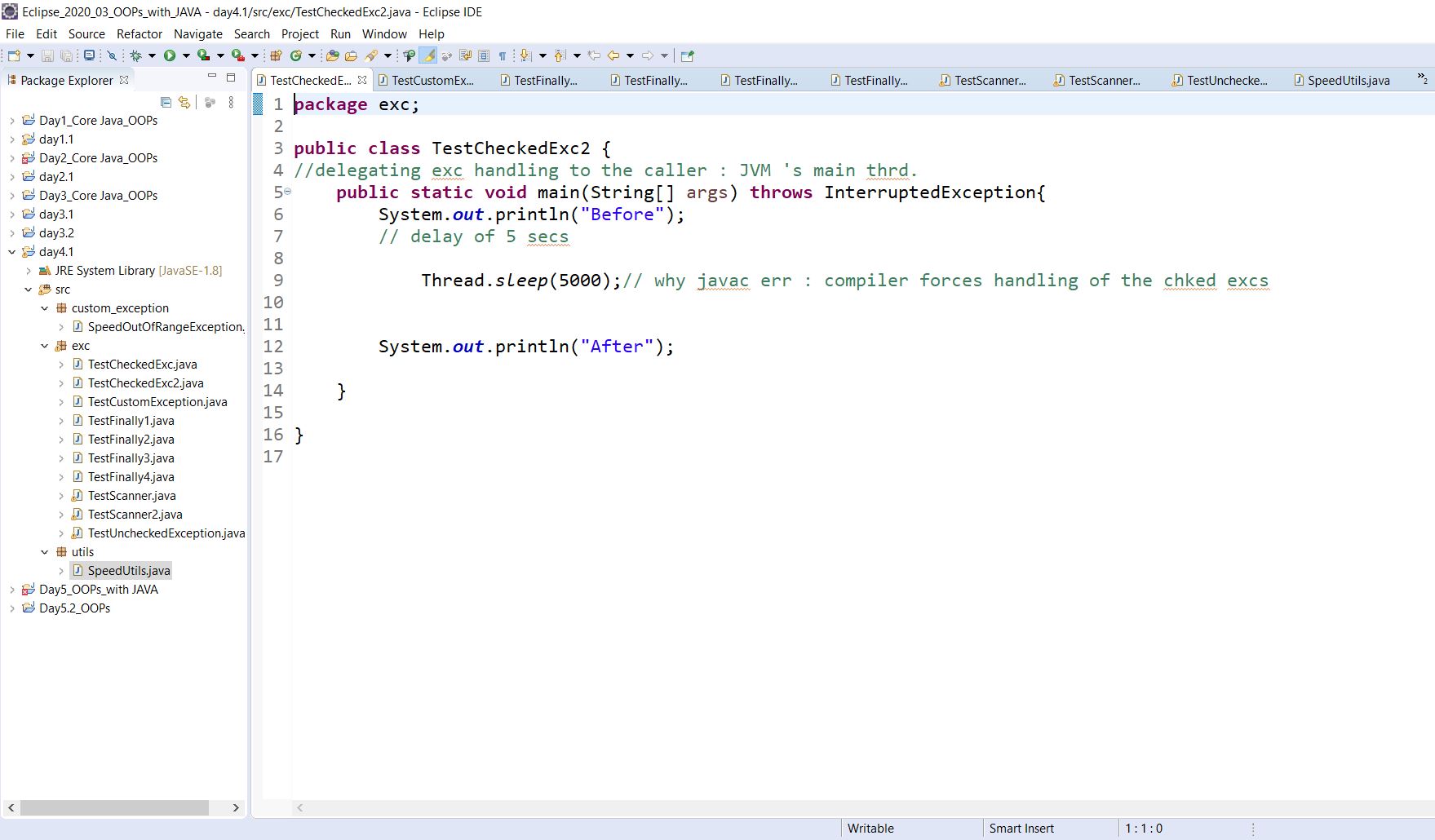
***Que*** .

-NO new assignments.

- importing eclipse project

***Ans :-***

: importing eclipse project



***Day 5*** :-

***Que*** .

Mini project (assignment based on all of earlier topics)

2.Create a customer management application along with validations.

2.1 Customer class

state --name,email,password,dateOfBirth(java.util.Date),custType(string),reg amount(double)

Add constructor n toString.

Unique ID : email

Override equals method correctly.

2.2 Create custom exception class(eg : InvalidInputException) , as a checked exception

2.3 Add I/P Validation rules in utils package : ValidationRules.java

Add them using different public static methods(eg : validateEmail, validatePassword etc)

Must use exception delegation (throws)

1.Customer email must contain "@" & email should end with ".com" (Use String class methods

contains , endsWith methods)

eg : public static String validateEmail(String email) throws InvalidInputException{...}

2. customer type must be either SILVER or GOLD or PLATUNUM (Use String class contains method)

3. customer's birth date must be after 1st Jan 1990.

(Hint : add a static method to parse(string--->date) n validate

For validation : Use java.util.Date class : before / after methods.

4. Reg amount must be in multiples of 500.

In case of validation failures , throw your custom exception , which will be actually handled in main(...)

of Tester class.

i.e Centralized exc handling is expected.

Tester

Supply following options

1. Register new customer

Accept customer details : name,email,password,dateOfBirth(dd-MM-yyyy pattern),custType,reg amount

Invoke validation rules.

In case of errors , show error message & application should continue.

In case of no erros , display "Customer registration successful" message.

Validation rule : Should not allow new registration with duplicate email

(Hint : use equals method)

2. Display all customer details (for-each)

3. Customer Login

I/P : email & password

In case of success , Display "Login Successful message" + Customer details

In case of failure : raise custom exception . Error mesg : Invalid email or Password.

(Hint : use equals method)

4. Change Password

I/P : email , old password , new password

In case of success , Display "Password changed for " customer name.

In case of failure : raise custom exception. Error mesg : "Password updation failed " : Invalid email or Password.

5 .Display sorted customers as per their email ids.

6. Exit

***Ans :-***

**1.Customer Class**

**package** com.customerManagement.aap;

**public** **class** Customer **implements** Comparable<Customer> {

**private** String name,email,password,custType,date;

**private** **double** reg\_amount;

**public** Customer(String name, String email, String password, String custType, String date, **double** reg\_amount) {

**super**();

**this**.name = name;

**this**.email = email;

**this**.password = password;

**this**.custType = custType;

**this**.date = date;

**this**.reg\_amount = reg\_amount;

}

@Override

**public** String toString() {

**return** "Customer [name=" + name + ", email=" + email + ", password=" + password + ", custType=" + custType + ", date=" + date + ", reg\_amount=" + reg\_amount + "]";

}

@Override

**public** **boolean** equals(Object o) {

**if**(o **instanceof** Customer) {

Customer c1=(Customer)o;

**return** **this**.email.equals(c1.email);

}

**return** **false**;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **double** getReg\_amount() {

**return** reg\_amount;

}

**public** **void** setReg\_amount(**double** reg\_amount) {

**this**.reg\_amount = reg\_amount;

}

@Override

**public** **int** compareTo(Customer second) {

**return** **this**.email.compareTo(second.email);

}}

**2.Account Class**

**package** utils;

**import** com.customerManagement.aap.Customer;

**public** **class** Account {

**public** **static** **void** login(Customer o[],String id,String passwd,**int** c) {

**for**(**int** i=0;i<c;i++) {

**if**(o[i].getEmail().equals(id) && o[i].getPassword().equals(passwd))

{

System.***out***.println(o[i].toString());

}

**else**

System.***out***.println("Check Email or Passwd");

}}

**public** **static** **boolean** changepass(Customer o[],String id,String passwd,String newPass,**int** c) {

**boolean** r=**false**;

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

{

**if**(o[i].getEmail().equals(id) && o[i].getPassword().equals(passwd))

{

System.***out***.println("Passwd Changed");

r= **true**;

}

**else**

System.***out***.println("Check Email or Passwd");

}

**return** r;

}

**public** **static** **void** display(Customer o[],**int** c) {

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

{

System.***out***.println(o[i].toString());

}}}

**3.InvalidInputHandling Class**

**package** exception;

**public** **class** InvalidInputHandling **extends** Exception {

String s;

**public** InvalidInputHandling(String s) {

//super(s);

**this**.s = s;

}

@Override

**public** String toString() {

**return** "Error [" + s +"]";

}}

**4.ValidationRules Class**

**package** utils;

**import** java.util.regex.Matcher;

**import** java.util.Date;

**import** java.util.regex.\*;

**import** java.text.SimpleDateFormat;

**import** exception.InvalidInputHandling;

**import** com.customerManagement.aap.Customer;

**public** **class** ValidationRules {

**static** String *str*="^[A-Za-z0-9+\_.]+@[A-Za-z.]+$";

**public** **static** **boolean** validEmail(String s) **throws** InvalidInputHandling {

Pattern p =Pattern.*compile*(*str*);

Matcher m=p.matcher(s);

**if**(m.matches()) {

**return** **true**;

}

**else** {

**throw** **new** InvalidInputHandling("Invalid Email");

}}

**public** **static** **boolean** validPassword(String p) **throws** InvalidInputHandling

{

**if**(p.length()>4) {

**return** **true**;

}

**else** {

**throw** **new** InvalidInputHandling("Password length must be 8 Charecter");

}}

**public** **static** **boolean** validDate(String d)**throws** InvalidInputHandling

{

**try** {

SimpleDateFormat dob=**new** SimpleDateFormat("dd-MM-yyyy");

Date date=dob.parse(d);

Date date1=**new** Date(1990);

**if**(date.after(date1)){

**return** **true**;

}

**else** {

**throw** **new** InvalidInputHandling("Date of Birth must be after 1990");

}}

**catch**(Exception e) {

**throw** **new** InvalidInputHandling("Date of Birth must be after 1990");

}}

**public** **static** **boolean** validAmount(**double** a) **throws** InvalidInputHandling {

**if**(a%500==0) {

**return** **true**;

}

**throw** **new** InvalidInputHandling("Amount must be Multiple of 500");

}

//customer type must be either SILVER or GOLD or PLATUNUM

**public** **static** **boolean** validCtype(String type) **throws** InvalidInputHandling {

String ss="SILVER GOLD PLATUNUM";

**if**(type.equalsIgnoreCase("SILVER") || type.equalsIgnoreCase("GOLD") || type.equalsIgnoreCase("PLATUNUM"))

**return** **true**;

**if**(type.contains(ss))

**return** **true**;

**throw** **new** InvalidInputHandling("customer type must be either SILVER or GOLD or PLATUNUM");

}

**public** **static** **boolean** validUid(Customer o, Customer[] oo,**int** counter) **throws** InvalidInputHandling

{

**int** count=0;

**for**(**int** i=0;i<=counter;i++) {

**if**(o.equals(oo[i])) {

count+=1;

}}

**if**(count==1) {

**return** **true**;

}

**throw** **new** InvalidInputHandling("duplicate Email ");

}

**public** **static** **boolean** Validate(String e,String p,String d,String type,**double** a) **throws** InvalidInputHandling

{

**if**(*validEmail*(e) && *validPassword*(p) && *validDate*(d) && *validAmount*(a) && *validCtype*(type))

{

**return** **true**;

}

**return** **false**;

}}

**5.Test Class**

**package** tester;

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.Scanner;

**import** **static** utils.ValidationRules.*Validate*;

**import** **static** utils.ValidationRules.*validUid*;

**import** utils.ValidationRules.\*;

**import** exception.InvalidInputHandling.\*;

**import** com.customerManagement.aap.Customer;

**import** **static** utils.Account.\*;

**public** **class** Test {

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

Scanner sc=**new** Scanner(System.***in***);

String n,e="",p="",d,ct,op1,op2,op3;

**double** a;

**boolean** flag=**false**,f=**false**;

**int** count=0;

System.***out***.println("Enter the SIZE of Customer");

Customer c[]=**new** Customer[sc.nextInt()];

**while**(!flag) {

System.***out***.println("\nEnter Choice\n1. Register new customer\n2. Display all customer details \n3. Customer Login\n4. Change Password\n5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).\r\n" +

"6. Exit");

**switch**(sc.nextInt()) {

**case** 1:

System.***out***.println("Enter Dateils:\n :name email password dateOfBirth(dd-MM-yyyy pattern) custType( SILVER or GOLD or PLATUNUM) reg amount(mul of 500)");

n=sc.next();

e=sc.next();

p=sc.next();

d=sc.next();

ct=sc.next();

a=sc.nextDouble();

**try** {

**if**(*Validate*(e,p,d,ct,a)) {

c[count]=**new** Customer(n, e, p, d, ct, a);

**for**(**int** i=0;i<=count;i++) {

**if**(*validUid*(c[count],c,count))

{

f=**true**;

}}

**if**(f) {

count++;

System.***out***.println("Customer registration successful");

}}}

**catch**(Exception ee) {

System.***out***.println(""+ee);

}

**break**;

**case** 2: *display*(c,count);

**break**;

**case** 3: System.***out***.println("Enter Gmail and Password");

op1=sc.next();

op2=sc.next();

*login*(c,op1,op2,count);

**break**;

**case** 4: System.***out***.println("I/P : email , old password , new password");

op1=sc.next();

op2=sc.next();

op3=sc.next();

**if**(*changepass*(c,op1,op2,op3,count)) {

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

{

**if**(c[i].getEmail().equals(op1) && c[i].getPassword().equals(op2))

{

c[i].setPassword(op3);

}}}

**break**;

**case** 5: Arrays.*sort*(c);

*display*(c,count);

**break**;

**case** 6: flag=**true**;

**break**;

}}

sc.close();

}}

***Output:-***

Enter the SIZE of Customer

3

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

1

Enter Dateils:

:name email password dateOfBirth(dd-MM-yyyy pattern) custType( SILVER or GOLD or PLATUNUM) reg amount(mul of 500)

Girish girish@gmail.com Girish@1993 21-11-1993 GOLD 500

Customer registration successful

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

1

Enter Dateils:

:name email password dateOfBirth(dd-MM-yyyy pattern) custType( SILVER or GOLD or PLATUNUM) reg amount(mul of 500)

Harish harish@gmail.com Harish@1993 11-10-1994 GOLD 500

Customer registration successful

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

1

Enter Dateils:

:name email password dateOfBirth(dd-MM-yyyy pattern) custType( SILVER or GOLD or PLATUNUM) reg amount(mul of 500)

Ravi ravi@gmail.com Ravi@1993 21-09-2000 GOLD 500

Customer registration successful

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

2

Customer [name=Girish, email=girish@gmail.com, password=Girish@1993, date=21-11-1993, custType =GOLD, reg\_amount=500.0]

Customer [name=Harish, email=harish@gmail.com, password=Harish@1993, date=11-10-1994, custType=GOLD, reg\_amount=500.0]

Customer [name=Ravi, email=ravi@gmail.com, password=Ravi@1993, date=21-09-2000, custType=GOLD, reg\_amount=500.0]

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

3

Enter Gmail and Password

ravi@gmail.com Ravi@1993

Check Email or Passwd

Check Email or Passwd

Customer [name=Ravi, email=ravi@gmail.com, password=Ravi@1993, date=21-09-2000, custType=GOLD, reg\_amount=500.0]

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

4

I/P : email , old password , new password

girish@gmail.com Girish@1993 9876543210

Passwd Changed

Check Email or Passwd

Check Email or Passwd

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit

5

Customer [name=Girish, email=girish@gmail.com, password=9876543210, date=21-11-1993, custType =GOLD, reg\_amount=500.0]

Customer [name=Harish, email=harish@gmail.com, password=Harish@1993, date=11-10-1994, custType =GOLD, reg\_amount=500.0]

Customer [name=Ravi, email=ravi@gmail.com, password=Ravi@1993, date=21-09-2000, custType=GOLD, reg\_amount=500.0]

Enter Choice

1. Register new customer

2. Display all customer details

3. Customer Login

4. Change Password

5 .Display sorted customers as per their email \*\*\*\*\*\*\*\*Need full array Initialaztion (if size is 5 then cutomer must be 5).

6. Exit