Task 1

Create a class Point2D , under package "com.cdac.geometry" for representing a point in x-y co-ordinate system.

1.1 Create a parameterized constructor to accept x & y co-ords.

1.2 Add public String show()) --to return point's x & y co-ords

1.3 Add isEqual method to Point2D class : boolean returning method : must return true if both points are having same x,y co-ords or false otherwise.

1.4 Add a method , calculateDistance , to calc distance between 2 points

Hint : use distance formula.

1.5 Create a driver class(for UI) , in the "tester" package "TestPoints" , with main(..)

Ask user , how many points to plot? :

Create suitable array.

1.6 Accept x,y co-ordinates for all the points n store it suitably.

1.7 Display x,y co-ordinates of all the points plotted so far ,using for-each loop.

1.8 Accept 2 indices from user .

Find out if the points at these indices are same or different (Hint : isEqual)

Print the message accordingly.

If points are not same , display distance between these 2 points.

Task 2

2.1 Can you arrange Fruit,Apple,Orange,Mango in inheritance hierarchy ?

Use tight encapsulation.

2.2 Properties (instance variables) : color : String , weight : double , name:String, fresh : boolean

2.3 Add suitable constructor.

2.4 Override toString correctly to return state of all fruits (return only : name ,color , weight )

2.5 Add a taste() method to return String form of the taste of the Fruit

eg : public String taste()

For Fruit : Can you identify taste of any general fruit ?

So return : "no specific taste"

Apple : should return "sweet n sour"

Mango : should return "sweet"

Orange : should return "sour"

2.6 Add specific functionality , in the sub classes

In Mango : public void pulp() {Display name n color of the fruit + a mesg creating pulp!}

In Orange : public void juice() {Display name n weight of the fruit + a mesg extracting juice!}

In Apple : public void jam() {Display name of the fruit + a mesg making jam!}

2.7 Add all of above classes under the package "com.app.fruits"

2.8 Create java application FruitBasket , with main method , as a tester

2.9 Prompt user for the basket size n create suitable data structure

2.10 Supply options

1. Add Mango

2. Add Orange

3. Add Apple

NOTE : You will be adding a fresh fruit in the basket , in all of above options.

4. Display names of all fruits in the basket.

5. Display name,color,weight , taste of all fresh fruits , in the basket.

6. Mark a fruit in a basket , as stale(=not fresh)

i/p : index

o/p : error message (in case of invalid index) or mark it stale

7. Mark all sour fruits stale

Hint : Use equals() method of the String class.

8. Invoke fruit specific functionality (pulp / juice / jam)

i/p : index

Invoke correct functionality (pulp / juice / jam)

10. Exit

TAsk 3

3. Solve this.

Fresh business scenario to apply inheritance , polymorphism n abstraction to emp based organization scenario.

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

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

3.1 Emp state--- id(int), firstName, lastName , deptId , basic(double)

emp id MUST be automatically generated.

Behaviour ---1. get emp details -- using toString.

2. compute net salary

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

Behaviour ----1. get mgr details : using toString.

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

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

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

Behaviour---

1. get worker details -- : override toString.

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.

NOTE : toString method SHOULD NOT include the net salary of the employee

Write TestOrganization in "tester" package.

Create suitable array to store organization details.

Provide following options

1. Hire Manager

i/p : manager details , except id

2. Hire Worker

i/p : worker details , except id

3. Display information of all employees(toString) including net salary(computeNetSalary) using single for-each loop.

4. Update basic salary

i/p : emp id , salary increment

In case of invalid emp id , either : display error message directly

OR (still better)

throw custom exception n handle it in centralized manner

(This can be added later!)

10 Exit