Exercise 1. Build a class ***Employee*** modeling an employee and:

- has a **int** representing the age of employee

- has **setAge** and **getAge** method

- implements the **Comparable** interface

(Note:

+ ***Comparable*** *interface is used to make user defined class's objects comparable. This interface declares one method* ***compareTo****(Object object) and determines how objects can be compared to with each other*

+ Signature of compareTo method is,

public int compareTo(Object object).

compareTo method should return 0 if both objects are equal, 1 if first grater than other and -1 if first less than the other object of the same class.

)

Create two different Employee objects in the **main** method, so that we can compare them

Exercise2. For managing bill of using water, you will define "Bill" class and "BillList" class

1) Implement a "**Bill**" class enclosing four fields: customer **code,** customer **type, old index** of water meter, **new index** of water meter**,** and some methods:

2) Initiate the fields of "**Bill**" class

3) Implement setX and getX methods for all the fields

4) Implement a input method of "**Bill"** class

5) Implement a method named **price** in "Bill" class to calculate unit price of using water. It is known unit price of using water is calculated as following:

customer type	**unit price of using water**
"Resident"	500
"Business" or "Organization"	400
otherwise	300

6) Implement a method named **payment** in "Bill" class to calculate payment for using water. It is known:

payment = unit price x (new index of water meter – old index of water meter)

7) Define "BillList" class enclosing two fields: a array of "Bill" objects, the number of Bills in the array, and methods as following:

8) Write a constructor to initiate number of bills in the array and the array

9) Write a method named **total** to calculate the total bill payment of all customers in the array having the same specified customer type

10) Write the method **remove** in the class BillList that removes the bill having the specified customer code

11) Write the method named **maxPayment** to return a first-found bill having largest payment

12) Write the **main** method to test all the methods above

Exerise 3. For managing professors and students at a university, you will find "Person" classas following:

```
public abstract class Person {
    protected String code;
    protected String name;
    protected int rank;
    public Person (String code, String name, int rank){
        this.code = code;
        this.name = name;
        this.rank = rank;
    }
    public String getCode() {
        return code;
    }
    public String getName() {
```

```
        return name;
    }
    public int getRank() {
        return rank;
    }
    public abstract String description();
    public void display() {
        System.out.println("Employee code:" + code);
        System.out.println("Employee name:" + name);
        System.out.println("Rank:" + rank);
    }
}
```

In this question, you will define, "Professor" class and "Student" class, which are subclasses of "Person" superclass

1)

a) Implement a "**Professor**" class that extends "Person" class. "Professor" class is added one field: **department** to which the professor belongs, and several methods:

b) Initiate the fields of "Professor" class

c) Implement a method named **description** in "Professor" class for returning description of professor rank.

It is known the ranking description of the professor is as following:

Rank	Description	Rank	Description
0	None		
1	Assistant Professor	4	Assistant Teaching Professor
2	Associate Professor	5	Associate Teaching Professor
3	Professor	6	Teaching Professor

d) Implement a method named **display** in "Professor" class to display the information of professor

2)

a) Implement a "Student" class that extends "People" class. "Student" class is added two field: **degree program** to which student belongs, **year of graduation**, and several methods:

b) Initiate the fields of "Student" class

c) Implement a method named **description** in "Student" class for returning description of Student rank.

It is known the ranking description of the student is as following:

Rank	Description	Rank	Description
0	None	4	Senior
1	Freshman	5	Graduate
2	Sophomore	6	Masters Postgraduate
3	Junior	7	PhD Postgraduate

d) Implement a method named **display** in "Student" class to display information of student

3)

In this question, for managing professors and students at the university, you will define "PersonList" class that models a list of professors and students.

1) "PersonList" class encloses two fields: a array of "Person" objects, the number of people in the array, and methods as following:

2) Write a constructor to initiate the array and number of people in the array

3) Write a method named **displayAll** to display the information of all the people in the array having specified rank

4) Write a method named average to compute average of rank of all the people

5) Write the method named maxRank to return a last-found person having maximum rank

6) Write the method removePerson in the class PersonList that removes the person having the specified person's code

Exercise 4. Write classes to represent 3 different types of Ant - Worker, Queen and Drone.
Each Ant has a floating-point health property, which is not writable externally and upon creation is set to a value of 100 (percent).
Each Ant has a Damage() method that takes a single integer parameter that should be a value between 0 and 100. When this method is called, the health of the Ant is to be reduced by that percentage of their current health.
When a Worker has a health below 70% it cannot fly and therefore is pronounced Dead. When a Queen has a health below 20%, or a Drone below 50%, it is pronounced dead. This 'Dead' property should be readable from each Ant. When a Ant is dead, no further health deductions should be recorded by the Ant, although the Damage() method should still be invokable without error.
Your application should create a single list containing 10 instances of each type of Ant and store in a list or array. It must support methods to allow Damage() to be called for each Ant, and to return the health status of each Ant, including whether it is alive or not.
Your application interface must contains 2 functions (user press "1" or "2" to activate this function:
1 – Create Ant list – Clear current Ant list and create new random Ants, then display in the console windows
2 – Attack Ants - Attack current Ant list, a different random value between 0 and 80 should be selected for each Ant and applied with a call to Damage(). After attacked, the user interface should refresh to show the health status of the Ants in console windows

Expectation of User interface:

```
-----ANT-----
1. Create new ant list
2. Damage Ant
3. Quit
Choose your option:1
DRONE   |       100.0   |       Alive
DRONE   |       100.0   |       Alive
WORKER  |       100.0   |       Alive
QUEEN   |       100.0   |       Alive
DRONE   |       100.0   |       Alive
QUEEN   |       100.0   |       Alive
DRONE   |       100.0   |       Alive
QUEEN   |       100.0   |       Alive
QUEEN   |       100.0   |       Alive
QUEEN   |       100.0   |       Alive
-----ANT-----
1. Create new ant list
2. Damage Ant
3. Quit
Choose your option:2
DRONE   |       25.0    |       Dead
DRONE   |       71.0    |       Alive
WORKER  |       100.0   |       Alive
QUEEN   |       78.0    |       Alive
DRONE   |       81.0    |       Alive
QUEEN   |       36.0    |       Alive
DRONE   |       33.0    |       Dead
QUEEN   |       17.0    |       Dead
QUEEN   |       77.0    |       Alive
QUEEN   |       88.0    |       Alive
-----ANT-----
1. Create new ant list
2. Damage Ant
3. Quit
Choose your option:
```