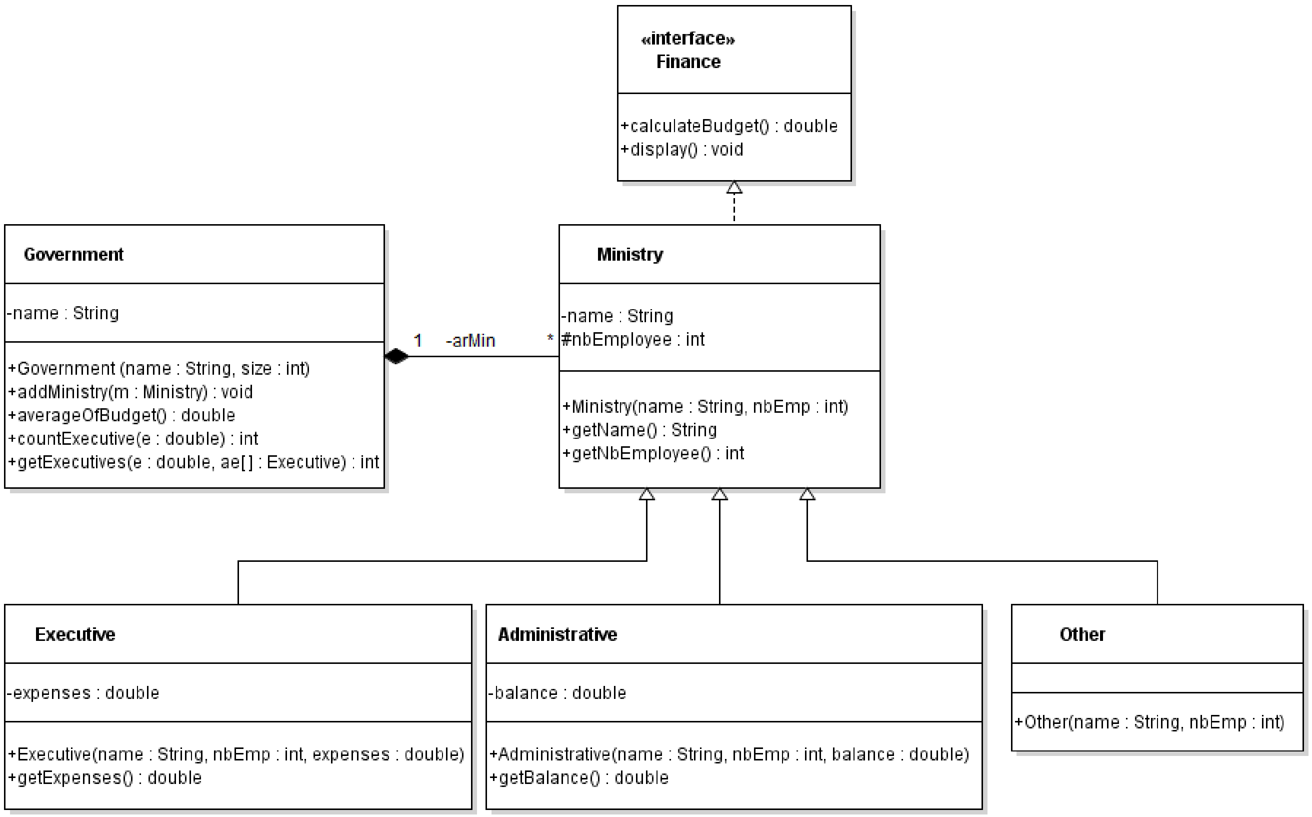
King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Midterm 2 Exam – Spring 2015

Exercise1:


«interface»
Finance
+calculateBudget() : double
+display() : void
Government
-name : String
+Government (name : String, size : int)
+addMinistry(m : Ministry) : void
+averageOfBudget() : double
+countExecutive(e : double) : int
+getExecutives(e : double, ae[ ] : Executive) : int
1
-arMin
*
Ministry
-name : String
#nbEmployee : int
+Ministry(name : String, nbEmp : int)
+getName() : String
+getNbEmployee() : int
Executive
-expenses : double
+Executive(name : String, nbEmp : int, expenses : double)
+getExpenses() : double
Administrative
-balance : double
+Administrative(name : String, nbEmp : int, balance : double)
+getBalance() : double
Other
+Other(name : String, nbEmp : int)

Finance interface:

- Methods:
  - ***calculateBudget ()***:This method calculates and returns the budget of a Ministry. The budget of a Ministry is calculated as follows:
    - ***Executive***:
      *Budget = expenses + nbEmployee * 1.5*
    - ***Administrative***:
      *Budget = nbEmployee * 10000 - balance.*
    - ***Other***
      *Budget = nbEmployee * 10000.*
  - ***display()*****:** display all the attributes

Ministry class:

- Attributes:
  - ***name***: the name of the ministry.
  - ***nbEmployee:*** number of employees in the ministry
- Methods:
  - ***Ministry(name: String, nbEmp: int)***: constructor
  - ***getName* ():** this method returns the name of the ministry.
  - ***getNbEmployee* ():** this method returns the number of employees of the ministry.

Executive class

- Attributes:
  - ***expenses***: the expenses of the ministry.
- Methods:
  - ***Executive(name: String, nbEmp : int, expenses : double)***: constructor.
  - ***getExpenses()***: this method returns the amount of expenses.

Administrative class:

- Attributes:
  - ***balance***: the basic salary allocated for the role.
- Methods:
  - ***Administrative (name: String , nbEmp : int, balance : double):*** constructor.
  - ***getBalance ():*** This method returns the balance of the administrative ministry.

Other class:

- Methods:
  - ***Other (name: String , nbEmp : int):*** constructor.

QUESTION: Translate into Java code the following:

- The interface ***Finance.*** **/2**

```
public interface Finance {

	public double calculateBudget();---------1
	public void display();---------1
}
```

- The class ***Ministry.*** **/3**

```
public abstract class Ministry implements Finance{---------1

	protected int nbEmployee;
	private String name;

	Ministry(String name, int nbEmployee)
	{
		this.name = name;
		this.nbEmployee = nbEmployee;
	}

	Ministry(Ministry m) ---------1
	{
		name = m.name;
		nbEmployee = m.nbEmployee;
```

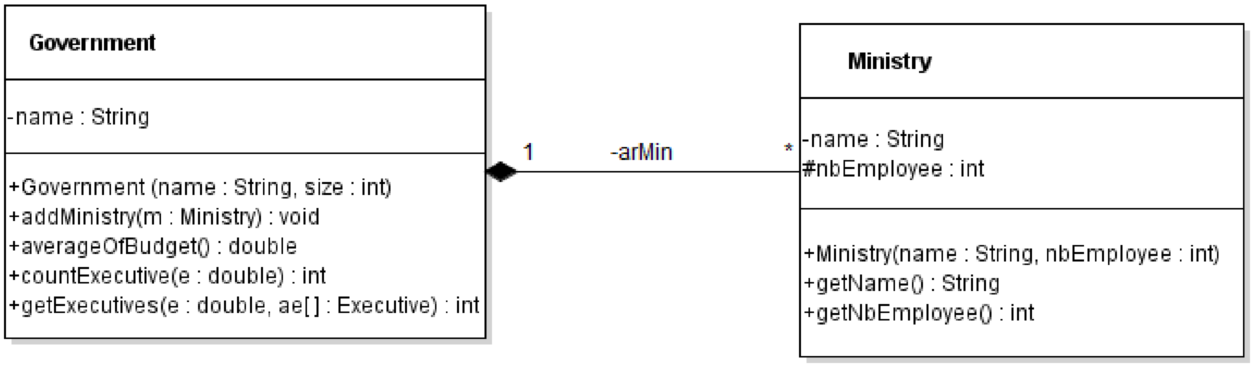
```
        }

        public int getNbEmployee() {
                return nbEmployee;
        }

        public String getName() {
                return name;
        }

        public void display() {---------1
                System.out.print(name + nbEmployee);
        }

}
```

- The class ***Executive.*** /5

```
public class Executive extends Ministry{---------1

        private double expenses;


        Executive(String name, int nbEmployee, double expenses)
        {
                super(name,nbEmployee); ---------1
                this.expenses = expenses;
        }

        Executive(Executive e) {---------1
                super(e); ---------1
                expenses = e.expenses;
        }

        public double calculateBudget() {---------1
                return expenses + nbEmployee * 1.5;
        }

        public double getExpenses() {
                return expenses;
        }


}
```

Exercise 2:

Let’s consider the same class *Ministry* described in exercise 1.


Government
-name : String
+Government (name : String, size : int)
+addMinistry(m : Ministry) : void
+averageOfBudget() : double
+countExecutive(e : double) : int
+getExecutives(e : double, ae[] : Executive) : int
1
-arMin
*
Ministry
-name : String
#nbEmployee : int
+Ministry(name : String, nbEmployee : int)
+getName() : String
+getNbEmployee() : int

Government class

- Attributes:
  - ***name***: the name of the Government.
- Methods:
  - ***Goverment (name: String, size: int)***: constructor.
  - ***addMinistry(m: Ministry)*: **this method adds the *Ministry* ***m*** to the Government.
  - ***averageOfBudget ():*** this method calculates and returns the average budget of all ministries of the government.
  - ***countExecutive(e: double):*** this method returns the number of **Executive** ministries with **expenses** greater than **e**.
  - ***getExecutives(e : double, ae[ ]: Executive):*** This method inserts into the array **ae** all ***Executive*** ministries having expenses greater than **e** and budget greater than the average budget. Also this method returns the number of ***Executive*** ministries added to **ae**.

QUESTION: Translate into Java code the class ***Government.***

```
public class Government { /30

	private String name;
	private Ministry arMin[];
	private int nb;

	Government(String name, int size) /2
	{
		this.name = name;
		arMin = new Ministry[size]; ---------1
		nb = 0; ---------1
	}
```

```
public void addMinistry(Ministry m) /7
{
        if(nb >= arMin.length) ---------1
                return;

        if(m instanceof Executive) ---------1
                arMin[nb] = new Executive( (Executive)m ); ---------1
        else if(m instanceof Administrative) ---------1
                arMin[nb] = new Administrative( (Administrative)m ); ---------1
        else
                arMin[nb] = new Other( (Other)m ); ---------1
        nb++;---------1
}

public double averageOfBudget() /5
{
        if(nb == 0) ---------1
                return 0;

        double sum =0; ---------1
        for(int i=0; i<nb; i++)---------1

        {
                sum+=arMin[i].calculateBudget();---------1
        }

        return sum/nb; ---------1
}

public int countExecutive(double e) /6
{
        int count = 0; ---------1
        for(int i=0;i<nb;i++)---------1
                if(arMin[i] instanceof Executive) ---------1
                {       //alt: if(((Executive) arMin[i]).getExpenses > e)
                        Executive x = (Executive) arMin[i]; ---------1
                        if(x.getExpenses() > e) ---------1
                                count++;
                }

        return count; ---------1
}
```

```
        public int getExecutives(double e, Executive ae[]) /10
        {
                double avg = averageOfBudget();----------1
                int j=0; ----------1


                for(int i=0; i<nb; i++)----------1

                {
                        if(arMin[i] instanceof Executive) ----------1
                                if(arMin[i].calculateBudget() > avg) ----------1
                                {
                                        Executive x = (Executive) arMin[i]; ----------1
                                        if(x.getExpenses() > e) ----------1
                                        {
                                                ae[j]=x; ----------1
                                                j++;----------1
                                        }

                                }
                }

                return j; ----------1
        }


}
```