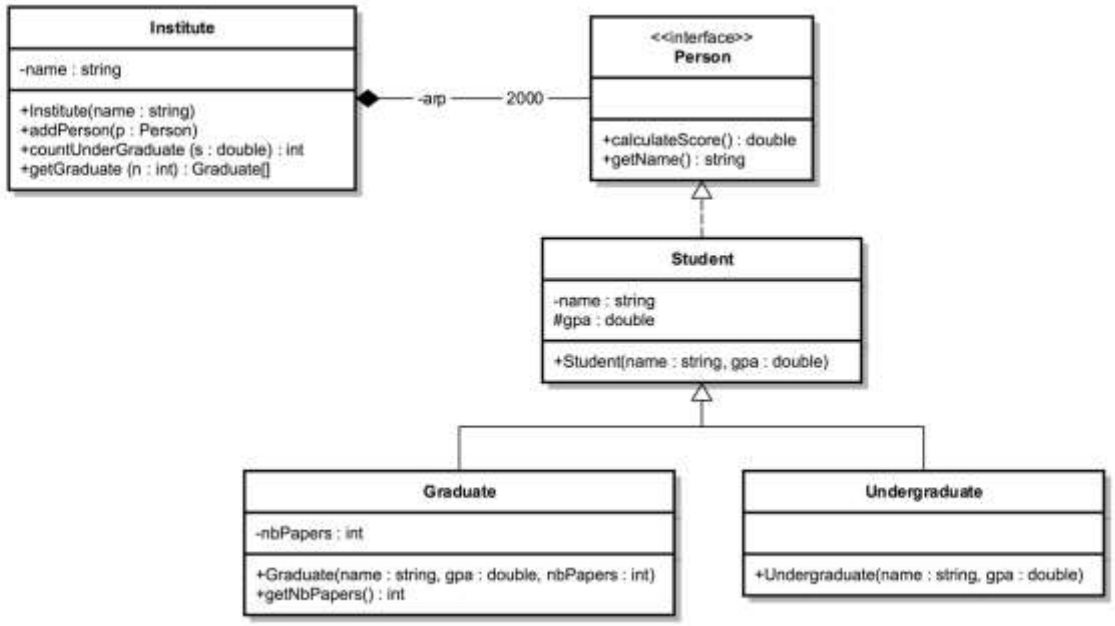
King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Abstract Classes and Interfaces Lab – Spring 2017



Interface Person:

- METHODS:
  - ***calculateScore ():*** calculated as:
    - ***for Graduate*** : Score = nbPapers * gpa
    - ***for UnderGraduate***: Score = gpa * 3 + 5
  - ***getName ():*** returns the ***name*** of the student.

Student class

- METHODS:
  - ***Student (name: String, gpa : double):*** constructor.

Graduate class

- METHODS:
  - ***Graduate (name: String, gpa : double, nbPpapers : int):*** constructor.
  - ***getNbPapers():*** getter for attribute ***nbPapers***.

UnderGraduate class

- METHODS:
  - ***UnderGraduate (name: String, gpa : double)***: constructor.

Institute class

- METHODS:
  - ***Institute(name: String):*** constructor.
  - ***addPerson(p: Person):*** add a person to the institute.
  - ***countUnderGraduate (s : double):*** count the number of ***UnderGraduate*** in the institute with score grater or equal to ***s.***
  - ***getGraduate (n : int):*** this method will return an array containing all the ***Graduate*** with number of papers greater than ***n***.

QUESTION: Translate into Java code **<u>all the classes</u>**

Solution:

```
public interface Person {
	double calculateScore();

	String getName();
}

public abstract class Student implements Person {
	private String name;
	protected double gpa;

	public Student(String name, double gpa) {
		this.name = name;
		this.gpa = gpa;
	}

	public Student(Student s) {
		name = s.name;
		gpa = s.gpa;
	}

	public String getName() {
		return name;
	}
}

public class Graduate extends Student {
	private int nbPapers;

	public Graduate(String name, double gpa, int nbPapers) {
		super(name, gpa);
		this.nbPapers = nbPapers;
	}
```

```
	public Graduate(Graduate g) {
		super(g);
		nbPapers = g.nbPapers;
	}

	public double calculateScore() {
		return nbPapers * gpa;
	}

	public int getNbPapers() {
		return nbPapers;
	}
}


public class Undergraduate extends Student {
	public Undergraduate(String name, double gpa) {
		super(name, gpa);
	}

	public Undergraduate(Undergraduate p) {
		super(p);
	}

	public double calculateScore() {
		return gpa * 3 + 5;
	}
}
public class Institute {
	private String name;
	private Person arp[];
	private int nb;

	Institute(String name) {
		this.name = name;
		arp = new Person[2000];
		nb = 0;
	}

	public void addPrson(Person p) {
		if (nb >= arp.length)
			return;
		if (p instanceof Graduate)
			arp[nb] = new Graduate((Graduate) p);
		else
			arp[nb] = new Undergraduate((Undergraduate) p);
		nb++;
	}

	public int countUnder(double s) {
		int count = 0;
		for (int i = 0; i < nb; i++)
			if (arp[i] instanceof Undergraduate)
				if (arp[i].calculateScore() >= s)
```

```
                        count++;
                return count;
        }

        public Graduate[] getGraduate(int n) {
                Graduate[] g = new Graduate[nb];
                int j = 0;
                for (int i = 0; i < nb; i++) {
                        if (arp[i] instanceof Graduate) {
                                Graduate x = (Graduate) arp[i];
                                if (x.getNbPapers() > n) {
                                        g[j] = x;
                                        j++;
                                }
                        }
                }
                return g;
        }
}
```