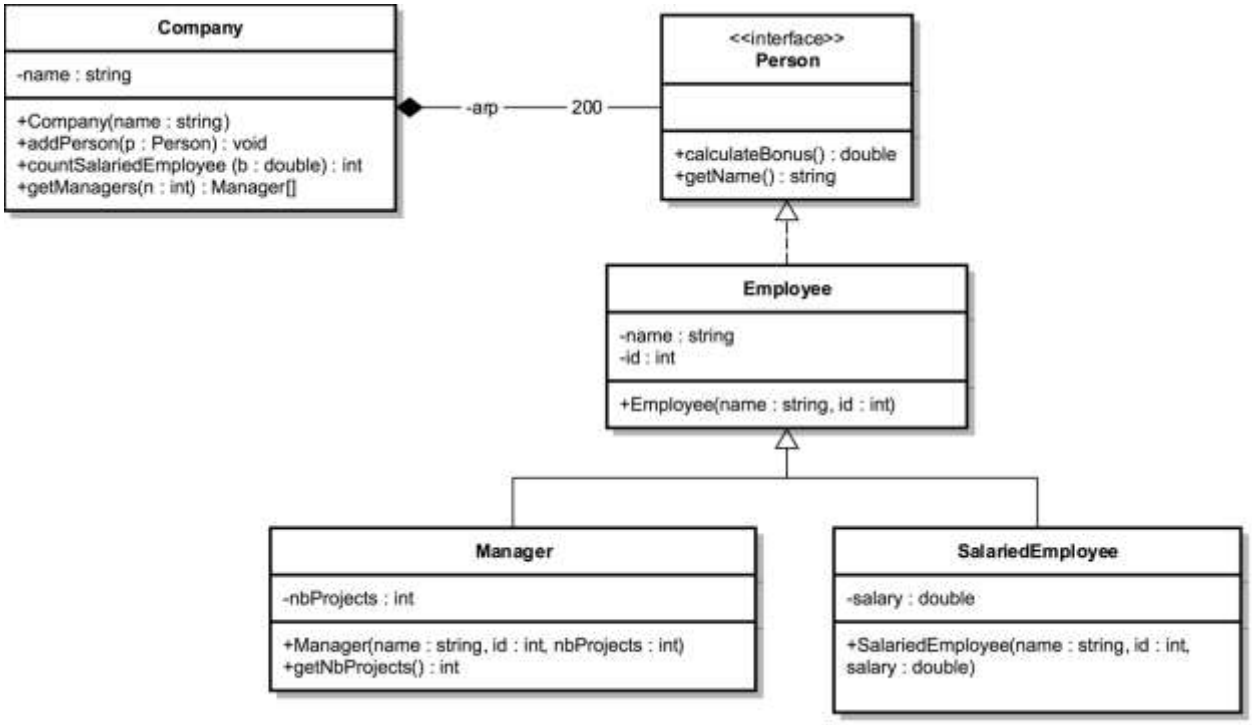
King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Abstract Classes and Interfaces Tutorial – Spring 2017



Interface Person:

- METHODS:
  - ***calculateBonus ():*** calculated as:
    - ***for Manager*** : Bonus = **nbProjects** * 10,000
    - ***for SalariedEmployee***: Bonus = **salary** * 2
  - ***getName ():*** returns the ***name*** of the Employee.

Employee class

- METHODS:
  - ***Employee (name: String, id : int):*** constructor.

Manager class

- METHODS:
  - ***Manager (name: String, id : int, nbProjects : int):*** constructor.
  - ***getNbProjects():*** getter for attribute ***nbProjects***.

SalariedEmployee class

- METHODS:
  - ***SalariedEmployee (name: String, id : int)***: constructor.

Company class

- METHODS:
  - ***Company(name: String):*** constructor.
  - ***addPerson(p: Person):*** add a person to the Company.
  - ***countSalariedEmployee (b : double):*** count the number of ***SalariedEmployee*** in the **Company** with **Bonus** grater or equal to ***b.***
  - ***getManagers(n : int):*** this method will return an array containing all the ***Manager*** with number of papers greater than ***n***.

QUESTION: Translate into Java code **all the classes**

Solution:

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

	String getName();
}

public abstract class Employee implements Person {
	private String name;
	private int id;

	public Employee(String name, int id) {
		this.name = name;
		this.id = id;
	}

	public Employee(Employee e) {
		name = e.name;
		id = e.id;
	}

	public String getName() {
		return name;
	}
}

public class Manager extends Employee {
	private int nbProjects;

	public Manager(String name, int id, int nbProjects) {
		super(name, id);
		this.nbProjects = nbProjects;
	}
```

```
	public Manager(Manager m) {
		super(m);
		nbProjects = m.nbProjects;
	}

	public double calculateBonus() {
		return nbProjects * 10000;
	}

	public int getNbProjects() {
		return nbProjects;
	}
}

public class SalariedEmployee extends Employee {
	private double salary;

	public SalariedEmployee(String name, int id, double salary) {
		super(name, id);
		this.salary = salary;
	}

	public SalariedEmployee(SalariedEmployee s) {
		super(s);
	}

	public double calculateBonus() {
		return salary * 2;
	}
}
public class Company {
	private String name;
	private Person arp[];
	private int nb;

	Company(String name) {
		this.name = name;
		arp = new Person[200];
		nb = 0;
	}

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

	public int countSalariedEmployee(double s) {
		int count = 0;
```

```
		for (int i = 0; i < nb; i++)
			if (arp[i] instanceof SalariedEmployee)
				if (arp[i].calculateBonus() >= s)
					count++;
		return count;
	}

	public Manager[] getManagers(int n) {
		Manager[] m = new Manager[nb];
		int j = 0;
		for (int i = 0; i < nb; i++) {
			if (arp[i] instanceof Manager) {
				Manager x = (Manager) arp[i];
				if (x.getNbProjects() > n) {
					m[j] = x;
					j++;
				}
			}
		}
		return m;
	}
}
```