Class Employee

```
public abstract class Employee {
	private int id;
	private String name;

	public Employee(int id, String name) {
		this.id = id;
		this.name = name;
	}
	public Employee(Employee e){
		this.id = e.id;
		this.name = e.name;
	}
	public int getId() {
		return id;
	}
	public String getName() {
		return name;
	}

	public abstract double calcSalary();

	public String toString(){
		return id + " : " + name;
	}
}
```

Class FullTime

```
public class FullTime extends Employee{
	private double salary;

	public FullTime(int id, String name, double salary) {
		super(id, name);
		this.salary = salary;
	}
	public FullTime(FullTime ft){
		super(ft);
		this.salary = ft.salary;
	}
	public String toString(){
		return super.toString() + " : " + salary;
	}
	public double calcAnnSalary(){ return salary * 12; }
}
```

Class PartTime

```
public class PartTime extends Employee{
	private int weeklyHours;
	private double hourlyRate;

	public PartTime(int id, String name, int weeklyHours,
					double hourlyRate) {
		super(id, name);
		this.weeklyHours = weeklyHours;
		this.hourlyRate = hourlyRate;
	}
	public PartTime(PartTime pt){
		super(pt);
		this.weeklyHours = pt.weeklyHours;
		this.hourlyRate = pt.hourlyRate;
	}
	public int getWeeklyHours() {
		return weeklyHours;
	}
	public double getHourlyRate() {
		return hourlyRate;
	}
	public String toString(){
		return super.toString() + " : " + weeklyHours + " : "
								+ hourlyRate;
	}
	public double calcAnnSalary(){
		return weeklyHours * hourlyRate * 4 * 12;
	}
}
```

Class Company

```
public class Company {
	private String name;
	private Employee [] arrEmp;
	private int nbEmp;
	public Company(String name, int size) {
		this.name = name;
		arrEmp = new Employee[size];
		nbEmp = 0;
	}
	public int search(Employee e){
		for(int i = 0; i < nbEmp; i++)
			if(arrEmp[i].getId() == e.getId())
				return i;
		return -1;
	}
```

```
        public boolean addEmployee(Employee e){
              if(search(e) != -1 || nbEmp == arrEmp.length)
                    return false;
              if(e instanceof FullTime)
                    arrEmp[nbEmp++] = new FullTime((FullTime) e);
              else
                    arrEmp[nbEmp++] = new PartTime((PartTime) e);
              return true;
        }
        public Employee getMaxSalary(){
              if(nbEmp == 0) return null;
              Employee max = arrEmp[0];
              for(int i = 1; i < nbEmp; i++)
                    if(arrEmp[i].calcAnnSalary() > max.calcAnnSalary())
                          max = arrEmp[i];
              return max;
        }
        public int countPartTime(){
              int count = 0;
              for(int i = 0; i < nbEmp; i++)
                    if(arrEmp[i] instanceof PartTime)
                          count++;
              return count;
        }
        public Employee[] getEmployees(int hours){
              Employee temp[] = new Employee[countPartTime()];
              int counter = 0;
              for(int i = 0; i < nbEmp; i++){
                    if(arrEmp[i] instanceof PartTime
                    && ((PartTime)arrEmp[i]).getWeeklyHours() > hours)
                          temp[counter++] = arrEmp[i];
              }
              return temp;
        }
        public int splitEmployees(FullTime FT[], PartTime PT[]){
              int countFT = 0, countPT = 0;
              for(int i = 0; i < nbEmp; i++){
                    if(arrEmp[i] instanceof FullTime)
                          FT[countFT++] = (FullTime) arrEmp[i];
                    else
                          PT[countPT++] = (PartTime) arrEmp[i];
              }
              return countPT;
        }

  }
```