

```

public abstract class Employee {

    private String name; private int id;

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

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public void display(){
        System.out.println("Name: " +name);
        System.out.println("Id: " +id);
    }

    public abstract double calculatePay();
}

public class FullTimeEmp extends Employee{

    private int salary;

    public FullTimeEmp(String name, int age, int salary) {
        super(name, age);
        this.salary = salary;
    }

    public FullTimeEmp(FullTimeEmp f) {
        super(f);
        this.salary = f.salary;
    }

    public void display(){
        super.display();
        System.out.println("Total monthly pay: " +calculatePay());
    }
}

```

```

        public double calculatePay(){
            return salary;
        }
    }

public class PartTimeEmp extends Employee {

    private int nbWorkHours;
    private int rate;

    public PartTimeEmp(String name, int id, int nbHours, int rate){
        super(name, id);
        this.nbWorkHours = nbHours;
        this.rate = rate;
    }

    public PartTimeEmp(PartTimeEmp p) {
        super(p);
        this.nbWorkHours = p.nbWorkHours;
        this.rate = p.rate;
    }

    public void display(){
        super.display();
        System.out.println("Number of weekly hours: " +nbWorkHours);
        System.out.println("Rate: " +rate);
        System.out.println("Total monthly pay: " +calculatePay());
    }

    public int getNbWorkHours() { return
        nbWorkHours;
    }

    public int getRate() { return
        rate;
    }

    public double calculatePay(){
        return nbWorkHours * 4 * rate;
    }
}

```

```

public class Company {

    private String name;
    private Employee[] arrEmployee; private
    int nbEmployee;

    public Company(String name, int size) throws NegativeArraySizeException{
        this.name = name;
        arrEmployee = new Employee[size];
        nbEmployee = 0;
    }

    public void displayAll(){
        System.out.println("Company Name: "+ name);
        for(int i = 0 ; i < nbEmployee ; i++){
            arrEmployee[i].display();
        }
    }

    public void addEmployee(Employee e) {
        if (nbEmployee < arrEmployee.length) {
            if (e instanceof PartTimeEmp)
                arrEmployee[nbEmployee++] = new PartTimeEmp((PartTimeEmp) e);
            else
                arrEmployee[nbEmployee++] = new FullTimeEmp((FullTimeEmp) e);
        }
        else
            throw new IllegalStateException("Can't add more employee");
    }

    public void deleteEmployee(String name) throws IndexOutOfBoundsException
    {
        int empId = searchName(name); arrEmployee[empId] =
        arrEmployee[nbEmployee-1]; nbEmployee--;
    }

    public int searchName(String name) {
        for (int i = 0; i < nbEmployee; i++)
            if (arrEmployee[i].getName().equalsIgnoreCase(name))
                return i;

        return -1;
    }
}

```

```
public double getYearlyPay(String name){
    int index = searchName(name);
        return arrEmployee[index].calculatePay() * 12;
}

public double calAvgPayForPartTime() throws ArithmeticException {
    int sum = 0;
    int nbPartTime = 0;
        for (int i = 0; i < nbEmployee; i++)
            if (arrEmployee[i] instanceof PartTimeEmp) {
                nbPartTime++;
                sum += arrEmployee[i].calculatePay();
            }
        return sum / nbPartTime;
}

}
```