112 學年度　第一學期　　課號：I4390

進階程式開發技術

程式作業報告

程式作業 **02**

姓名：李信均

座號：I4B18

(1) Create a class called **Employee**. An employee is described by a first name, a last name, an employee ID, and a monthly salary. Provide instance variables to describe an employee. Provide a constructor that initializes the instance variables. Provide a *set* and a *get* method for each instance variable except the instance variable of the month salary. Provide a method to print out the values of all instance variables. Write a test application to demonstrate class capabilities. (25 分)

主要想法說明:

創建一個 class Employee，

```
 5 public class Employee {
 6     private String firstName;
 7     private String lastName;
 8     private String employeeId;
 9     private int monthlySalary;
```

創建 construture，

```
11    public Employee() {
12    }
13
14    public Employee(String firstName, String lastName, String employeeId, int monthlySalary) {
15        this.firstName = firstName;
16        this.lastName = lastName;
17        this.employeeId = employeeId;
18        this.monthlySalary = monthlySalary;
19    }
```

增加 member 的 set/get function，

```
21    public void setFirstName(String firstName) {
22        this.firstName = firstName;
23    }
24
25    public String getFirstName() {
26        return firstName;
27    }
28
29    public void setLastName(String lastName) {
30        this.lastName = lastName;
31    }
32
33    public String getLastName() {
34        return lastName;
35    }
36
37    public void setEmployeeId(String employeeId) {
38        this.employeeId = employeeId;
39    }
40
41    public String getEmployeeId() {
42        return employeeId;
43    }
44
45    public void setMonthlySalary(int monthlySalary) {
46        this.monthlySalary = monthlySalary;
47    }
48
49    public int getMonthlySalary() {
50        return monthlySalary;
51    }
```

增加輸出和輸入的 function。

```
53●     public void print() {
54          System.out.println("First Name: " + firstName);
55          System.out.println("Last Name: " + lastName);
56          System.out.println("Employee ID: " + employeeId);
57          System.out.println("Monthly Salary: " + monthlySalary);
58      }
59
60●     public Employee scan(Scanner scan) {
61          System.out.println("---------------Input Employee Information ------------");
62          System.out.print("First Name: ");
63          this.setFirstName(scan.next());
64          System.out.print("Last Name: ");
65          this.setLastName(scan.next());
66          System.out.print("Employee ID: ");
67          this.setEmployeeId(scan.next());
68          System.out.print("Monthly Salary: ");
69          this.setMonthlySalary(scan.nextInt());
70          return this;
71      }
```

測試程式:

設置預設的 employee 並輸出

```
Employee employee_default = new Employee("Matt", "Lee", "A001", 85000);

System.out.println("Default Employee Information:");
employee_default.print();
```

設置一個 array list 存放 employee，宣告 cont 來決定是否繼續加入 employee，先宣告一個空的 employee，呼叫 employee function scan()輸入 employee 資料，將新宣告的 employee 存入 arrayList 並決定是否繼續加入，

```
ArrayList <Employee> employees = new ArrayList<>();
String cont = "Y";
Scanner scan = new Scanner(System.in);
while ( cont.equalsIgnoreCase("y")) {
    Employee employee = new Employee();
    employee = employee.scan(scan);
    employees.add(employee);
    System.out.print("Input next?(y/n): ");
    cont = scan.next();
}
scan.close();
```

呼叫 employee function print()輸出所有存放到 arrayList 的 employee。

```
for (Employee e: employees) {
    System.out.println("---------------Show Employee Information ------------");
    e.print();
}
```

輸出:

最上方為預設的資料，輸入兩筆資料(藍色)並輸出。

```
Default Employee Information:
First Name: Matt
Last Name: Lee
Employee ID: A001
Monthly Salary: 85000
---------------Input Employee Information ------------
First Name: Jane
Last Name: Chang
Employee ID: A002
Monthly Salary: 24000
Input next?(y/n): y
---------------Input Employee Information ------------
First Name: Jing
Last Name: Feng
Employee ID: A003
Monthly Salary: 59000
Input next?(y/n): n
---------------Show Employee Information ------------
First Name: Jane
Last Name: Chang
Employee ID: A002
Monthly Salary: 24000
---------------Show Employee Information ------------
First Name: Jing
Last Name: Feng
Employee ID: A003
Monthly Salary: 59000
```

(2) An employee can be a full time or part time employee. Using the **Employee** class in exercise (2) as the base class. Creates two classes the **fullTimeEmployee** class which represents the full time employee and the **partTimeEmployee** class which represents the part time employee. The **fullTimeEmployee** class and the **partTimeEmployee** class are derived class of the **Employee** class. The difference between a full time employee and a part time employee is the computation of month salary. The full time employee has additional description such as base payment(底薪), overtime work hours(加班時數) and overtime work hour pay(每小時加班費). The overtime payment has additional description such as the number of working hours(工作時數) and hour pay(時薪). The month salary of employee is computed as follows:

full time employee's month payment = base payment + overtime work hours * overtime work hours pay

part time employee's month payment = number of working hours * hour pay

The derived class **fullTimeEmployee** class and the **partTimeEmployee** class should inherit all the methods of base class **Employee**. Please add method to compute the month salary for each derived class. Write a test application to demonstrate the inheritance property. (25 分)

主要想法說明:

創建兩個 class，皆繼承於 Employee，兩個 class 分別為 FullTimeEmployee 跟 PartTimeEmployee，member 和 function 皆有些許不同。

創建 class FullTimeEmployee 繼承 Employee，

```
5 public class FullTimeEmployee extends Employee {
6     private int basePayment;
7     private double overtimeWorkHours;
8     private int overtimeWorkHoursPay;
```

創建 construture(後面還有參數，礙於長度關係裁切掉)，月薪部分直接設為 0，

```
public FullTimeEmployee() {

}

public FullTimeEmployee(String firstName, String lastN
    super(firstName, lastName, employeeId, 0);
    this.basePayment = basePayment;
    this.overtimeWorkHours = overtimeWorkHours;
    this.overtimeWorkHoursPay = overtimeWorkHoursPay;
}
```

創建 member 的 setter/getter，

```
public void setBasePayment(int basePayment) {
    this.basePayment = basePayment;
}

public int getBasePayment() {
    return basePayment;
}

public void setOvertimeWorkHours(double overtimeWorkHours) {
    this.overtimeWorkHours = overtimeWorkHours;
}

public double getOvertimeWorkHours() {
    return overtimeWorkHours;
}

public void setOvertimeWorkHoursPay(int overtimeWorkHoursPay) {
    this.overtimeWorkHoursPay = overtimeWorkHoursPay;
}

public int getOvertimeWorkHoursPay() {
    return overtimeWorkHoursPay;
}
```

增加輸出和輸入的 function，這裡有使用到繼承 super，不直接輸入月薪，

```
@Override
public void print() {
    super.print();
    System.out.println("Base Payment: " + basePayment);
    System.out.println("Overtime Work Hours: " + overtimeWorkHours);
    System.out.println("Overtime Hour Pay: " + overtimeWorkHoursPay);
}

public FullTimeEmployee scan(Scanner scan) {
    System.out.println("---------------Input FullTimeEmployee Information ------------");
    System.out.print("First Name: ");
    this.setFirstName(scan.next());
    System.out.print("Last Name: ");
    this.setLastName(scan.next());
    System.out.print("Employee ID: ");
    this.setEmployeeId(scan.next());
    System.out.print("Base payment: ");
    this.setBasePayment(scan.nextInt());
    System.out.print("Overtime work hours: ");
    this.setOvertimeWorkHours(scan.nextDouble());
    System.out.print("Overtime work hour pay: ");
    this.setOvertimeWorkHoursPay(scan.nextInt());
    return this;
}
```

新增 member function 計算月薪，公式: 底薪 + 加班時數 * 加班時薪。

```
public void computeMonthlySalary() {
    setMonthlySalary((int)(basePayment + overtimeWorkHours * overtimeWorkHoursPay));
}
```

創建 class PartTimeEmployee 繼承 Employee，

```
public class PartTimeEmployee extends Employee {
    private double workingHours;
    private int hourPay;
```

創建 construture(後面還有參數，礙於長度關係裁切掉)，月薪部分直接設為 0，

```
public PartTimeEmployee() {

}

public PartTimeEmployee(String firstName, String
    super(firstName, lastName, employeeId, 0);
    this.workingHours = workingHours;
    this.hourPay = hourPay;
}
```

創建 member 的 setter/getter，

```
public void setWorkingHours(double workingHours) {
    this.workingHours = workingHours;
}

public double getWorkingHours() {
    return workingHours;
}

public void setHourPay(int hourPay) {
    this.hourPay = hourPay;
}

public double getHourPay() {
    return hourPay;
}
```

增加輸出和輸入的 function，這裡有使用到繼承 super，不直接輸入月薪，

```
@Override
public void print() {
    super.print();
    System.out.println("Working Hours: " + workingHours);
    System.out.println("Hour Pay: " + hourPay);
}

public PartTimeEmployee scan(Scanner scan) {
    System.out.println("---------------Input PartTimeEmployee Information ------------");
    System.out.print("First Name: ");
    this.setFirstName(scan.next());
    System.out.print("Last Name: ");
    this.setLastName(scan.next());
    System.out.print("Employee ID: ");
    this.setEmployeeId(scan.next());
    System.out.print("Work hours: ");
    this.setWorkingHours(scan.nextDouble());
    System.out.print("Hour pay: ");
    this.setHourPay(scan.nextInt());
    return this;
}
```

新增 member function 計算月薪，公式: 時數 * 時薪。

```
public void computeMonthlySalary() {
    setMonthlySalary((int)(workingHours * hourPay));
}
```

測試程式:

先預設好正職和工讀各一位，並呼叫計算月薪的 function 後輸出，

```
FullTimeEmployee fullTimeEmployee = new FullTimeEmployee("Matt", "Lee", "A001", 36000, 30, 200);
PartTimeEmployee partTimeEmployee = new PartTimeEmployee("Jennessa", "Chang", "A002", 120, 200);

fullTimeEmployee.computeMonthlySalary();
partTimeEmployee.computeMonthlySalary();

System.out.println("Default fullTimeEmployee Information:");
fullTimeEmployee.print();

System.out.println("\nDefault partTimeEmployee Information:");
partTimeEmployee.print();
```

宣告輛個不同的 array list，分別存放 FullTimeEmployee、PartTimeEmployee，Slc 為選擇增加 FullTimeEmployee 或 PartTimeEmployee 或結束輸入，

```
ArrayList <FullTimeEmployee> fullTimeEmployees = new ArrayList<>();
ArrayList <PartTimeEmployee> partTimeEmployees = new ArrayList<>();
int slc = 1;
Scanner scan = new Scanner(System.in);
System.out.print("Input next?\n0 to quit, 1 to fullTimeEmployees, 2 to partTimeEmployees: ");
slc = scan.nextInt();
```

如 slc=1 為加入正職，輸入並計算月薪後存放在正職的 array list，

如 slc=2 為加入正職，輸入並計算月薪後存放在工讀的 array list，

如 slc=0 為結束輸入，

```
while ( slc == 1 || slc == 2) {
    if (slc == 1) {
        FullTimeEmployee employee = new FullTimeEmployee();
        employee = employee.scan(scan);
        fullTimeEmployee.computeMonthlySalary();
        fullTimeEmployees.add(employee);
    }
    else {
        PartTimeEmployee employee = new PartTimeEmployee();
        employee = employee.scan(scan);
        partTimeEmployee.computeMonthlySalary();
        partTimeEmployees.add(employee);
    }

    System.out.print("Input next?\n0 to quit, 1 to fullTimeEmployees, 2 to partTimeEmployees: ");
    slc = scan.nextInt();
}
scan.close();
```

呼叫 employee function print()輸出所有存放到 arrayList 的正職和工讀。

```
for (FullTimeEmployee e: fullTimeEmployees) {
    System.out.println("---------------Show FullTimeEmployee Information ------------");
    e.print();
}
for (PartTimeEmployee e: partTimeEmployees) {
    System.out.println("---------------Show PartTimeEmployee Information ------------");
    e.print();
}
```

輸出:

預設輸出員工輸出，

```
Default fullTimeEmployee Information:
First Name: Matt
Last Name: Lee
Employee ID: A001
Monthly Salary: 42000
Base Payment: 36000
Overtime Work Hours: 30.0
Overtime Hour Pay: 200

Default partTimeEmployee Information:
First Name: Jennessa
Last Name: Chang
Employee ID: A002
Monthly Salary: 24000
Working Hours: 120.0
Hour Pay: 200
Input next?
0 to quit, 1 to fullTimeEmployees, 2 to partTimeEmployees: 1
```

自行輸入兩筆資料分別為正職和工讀(藍色)後結束並輸出。

```
---------------Input FullTimeEmployee Information ------------
First Name: Jane
Last Name: Chang
Employee ID: A003
Base payment: 28000
Overtime work hours: 10
Overtime work hour pay: 180
Input next?
0 to quit, 1 to fullTimeEmployees, 2 to partTimeEmployees: 2
---------------Input PartTimeEmployee Information ------------
First Name: Jing
Last Name: Feng
Employee ID: A004
Work hours: 120
Hour pay: 230
Input next?
0 to quit, 1 to fullTimeEmployees, 2 to partTimeEmployees: 0
---------------Show FullTimeEmployee Information ------------
First Name: Jane
Last Name: Chang
Employee ID: A003
Monthly Salary: 29800
Base Payment: 28000
Overtime Work Hours: 10.0
Overtime Hour Pay: 180
---------------Show PartTimeEmployee Information ------------
First Name: Jing
Last Name: Feng
Employee ID: A004
Monthly Salary: 27600
Working Hours: 120.0
Hour Pay: 230
```

(3) Design a program to sort an array of integers into ascending order. Your program must input ***k*** integers, sort the integers by using the **selection sort**, and output the sorting result. Your program must represent the integers by the objects (i.e. integers) of **Integer class**, store the objects of **Integer class** in an array, and use the methods of Integer class to perform the sorting computation. (25 分)

主要想法說明:

先輸入 k，

```
public class selectionSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of k: ");
        int k = scanner.nextInt();
```

創建一個存放 k 個 Integer 的 array，輸入 k 個數字存進去並將其輸出，

```
Integer[] integers = new Integer[k];

for (int i = 0; i < k; i++) {
    System.out.print(i + ": " );
    integers[i] = scanner.nextInt();
}

scanner.close();

System.out.print("Unsorted order: ");
for (Integer num : integers) {
    System.out.print(num + " ");
}
```

呼叫 function 進行 selection sort，使用到 compareTo 找最小的數字，如果比較的數字大於被比較的回傳 1、相等回傳 0、小於則回傳-1，因此只要得到回傳值為小於 0 時，將這個位置儲存起來再和後面的比較，最後得出一個最小值和 i 的位置交換數字，i 會由 0 到 n-1，最後這個 array 會變為由小排到大的 array。

```
29          sort(integers);
```

```
37●     public static void sort(Integer[] integers) {
38          int n = integers.length;
39
40          for (int i = 0; i < n-1; i++) {
41              int min = i;
42
43              for (int j = i + 1; j < n; j++) {
44                  if (integers[j].compareTo(integers[min]) < 0) {
45                      min = j;
46                  }
47              }
48
49              Integer temp = integers[i];
50              integers[i] = integers[min];
51              integers[min] = temp;
52          }
53      }
```

最後將排序過後的 array 輸出。

```
System.out.print("\nSorted order: ");
for (Integer num : integers) {
    System.out.print(num + " ");
}
```

輸出:

```
Enter the number of k: 5
0: 5
1: 4
2: 3
3: 21
4: 2
Unsorted order: 5 4 3 21 2
Sorted order: 2 3 4 5 21
```

```
Enter the number of k: 10
0: 234
1: 123
2: 324
3: 235
4: 2341
5: 123
6: 12342
7: 324
8: 3
9: 24
Unsorted order: 234 123 324 235 2341 123 12342 324 3 24
Sorted order: 3 24 123 123 234 235 324 324 2341 12342
```

(4) Create a class called **Quadrilateral** (四邊形) to represent a quadrilateral(四邊形). A quadrilateral in a two-dimensional plane is a polygon (多邊形) with four edges (四個邊) and four vertices(四個頂點). A quadrilateral is described by its four vertices. Each vertex is described by a coordinate (x, y) in the two dimensional space. Use an **inner class** called **Point** to represent the vertex coordinate. The **Point** class should have the following methods:

- A constructor of **Point** class to initialize the coordinate.
- A *set* and a *get* method of **Point** class for each instance variable in **Point** class.

For the **Quadrilateral** class,

- Use object of the **Point** class as the instance variable of **Quadrilateral** class.
- Provide a constructor that initializes the instance variables.
- Provide a *set* and a *get* method for each instance variable.
- Also, provide a method to print the coordinates of four vertices of the quadrilateral.

Write a test application to demonstrate the **nested** class capabilities. (25 分)

主要想法說明:

創建一個 class Quadrilateral，這裡比較偷懶沒有另外創一個 point 而是直接包在 Quadrilateral 裡面，point 有 x,y 座標 setter/getter，

```
3 public class Quadrilateral {
4     public class Point {
5         private int x;
6         private int y;
7
8         public Point() {
9         }
10
11        public Point(int x, int y)
12            this.x = x;
13            this.y = y;
14        }
15
16        public int getX() {
17            return x;
18        }
19
20        public void setX(int x) {
21            this.x = x;
22        }
23
24        public int getY() {
25            return y;
26        }
27
28        public void setY(int y) {
29            this.y = y;
30        }
31    }
```

class Quadrilateral 包含四個 point、constructurec 和輸出這個四邊形，

```
private Point point1;
private Point point2;
private Point point3;
private Point point4;

public Quadrilateral() {
}

public Quadrilateral(Point point1, Point point2, Point point3, Point point4) {
    this.point1 = point1;
    this.point2 = point2;
    this.point3 = point3;
    this.point4 = point4;
}

public void printCoordinates() {
    System.out.println("Coordinates of the four vertices:");
    System.out.println("Point 1: (" + point1.getX() + ", " + point1.getY() + ")");
    System.out.println("Point 2: (" + point2.getX() + ", " + point2.getY() + ")");
    System.out.println("Point 3: (" + point3.getX() + ", " + point3.getY() + ")");
    System.out.println("Point 4: (" + point4.getX() + ", " + point4.getY() + ")");
}
```

測試程式：

輸入四個座標並將他們加入到一個 array list 裡，方便使用遞迴，

```
Scanner scan = new Scanner(System.in);
System.out.println("Input quadrilateral:");
Quadrilateral.Point point1 = new Quadrilateral().new Point();
Quadrilateral.Point point2 = new Quadrilateral().new Point();
Quadrilateral.Point point3 = new Quadrilateral().new Point();
Quadrilateral.Point point4 = new Quadrilateral().new Point();
ArrayList <Quadrilateral.Point> points = new ArrayList<>();
points.add(point1);
points.add(point2);
points.add(point3);
points.add(point4);
```

輸入每個 point 的 x, y，

```
for (int i = 0; i< 4; i++) {
    System.out.print("Input "+i+"-X:");
    points.get(i).setX(scan.nextInt());
    System.out.print("Input "+i+"-Y:");
    points.get(i).setY(scan.nextInt());
}
scan.close();
```

創建一個新的四邊形並將其輸出。

```
 Quadrilateral quad = new Quadrilateral(
         points.get(0),
         points.get(1),
         points.get(2),
         points.get(3));


quad.printCoordinates();
```

輸出:

```
Input quadrilateral:
Input 0-X:0
Input 0-Y:0
Input 1-X:0
Input 1-Y:1
Input 2-X:1
Input 2-Y:0
Input 3-X:1
Input 3-Y:1
Coordinates of the four vertices:
Point 1: (0, 0)
Point 2: (0, 1)
Point 3: (1, 0)
Point 4: (1, 1)
```