Ch11練習

鄭安翔

ansel_cheng@hotmail.com

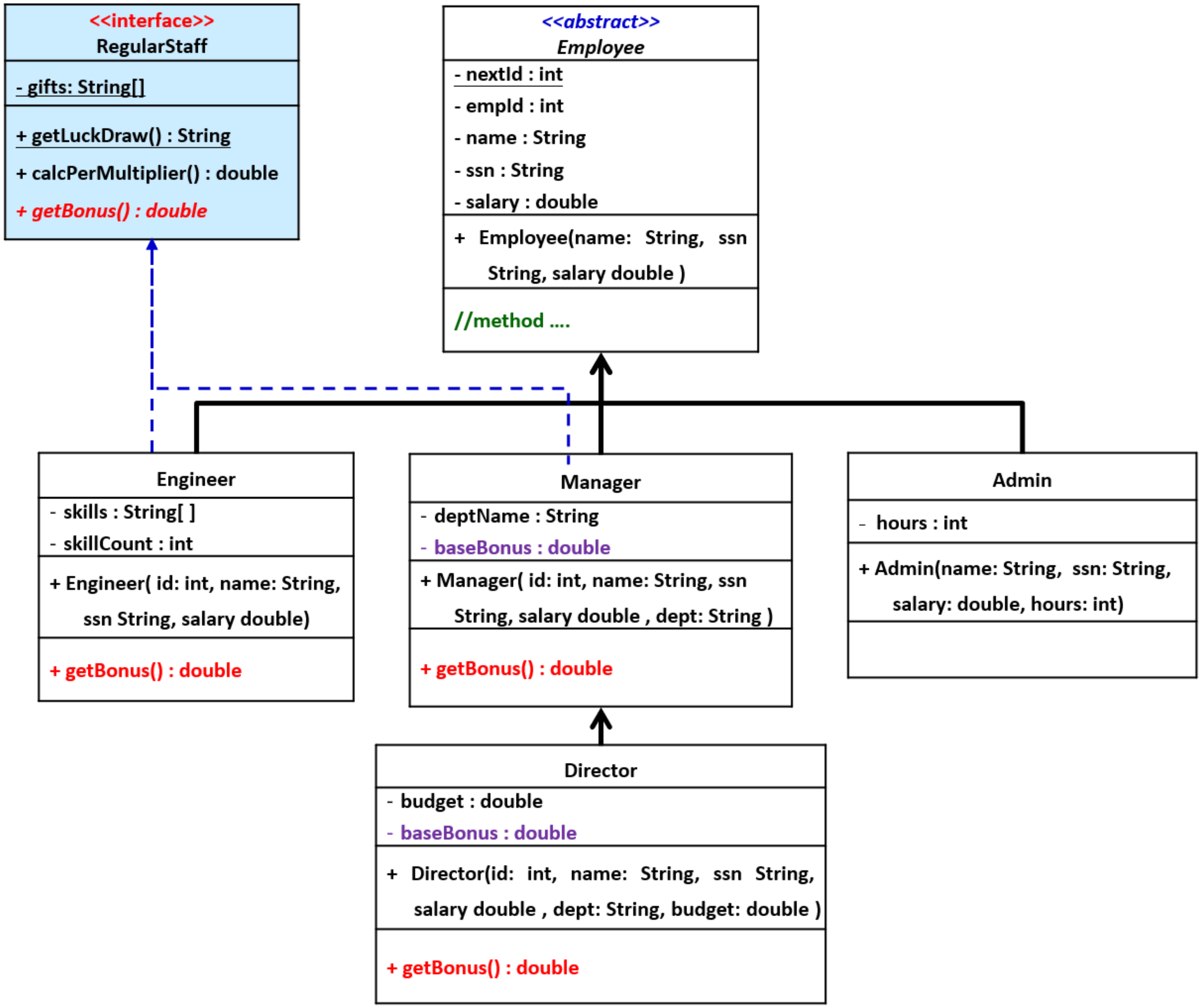
練習一

1. 修改 EmployeePractice 專案
2. 宣告 RegularStaff 介面 (正職員工)
   - 宣告 String[] 屬性 gifts
   - 宣告 getLuckDraw() : String 類別方法
   - 宣告 calcPerMultiplier() : double 預設方法
   - 宣告 getBonus() : double 抽象方法
3. 修改 Engineer 類別，實作 RegularStaff 介面
   - 實作 getBonus() 方法，傳回 salary * calcPerMultiplier()
4. 修改 Manager, 類別，實作RegularStaff 介面
   - 宣告 baseBonus 類別屬性，數值為100,000
   - 實作 getBonus() 方法，傳回 baseBonus * calcPerMultiplier()

練習一

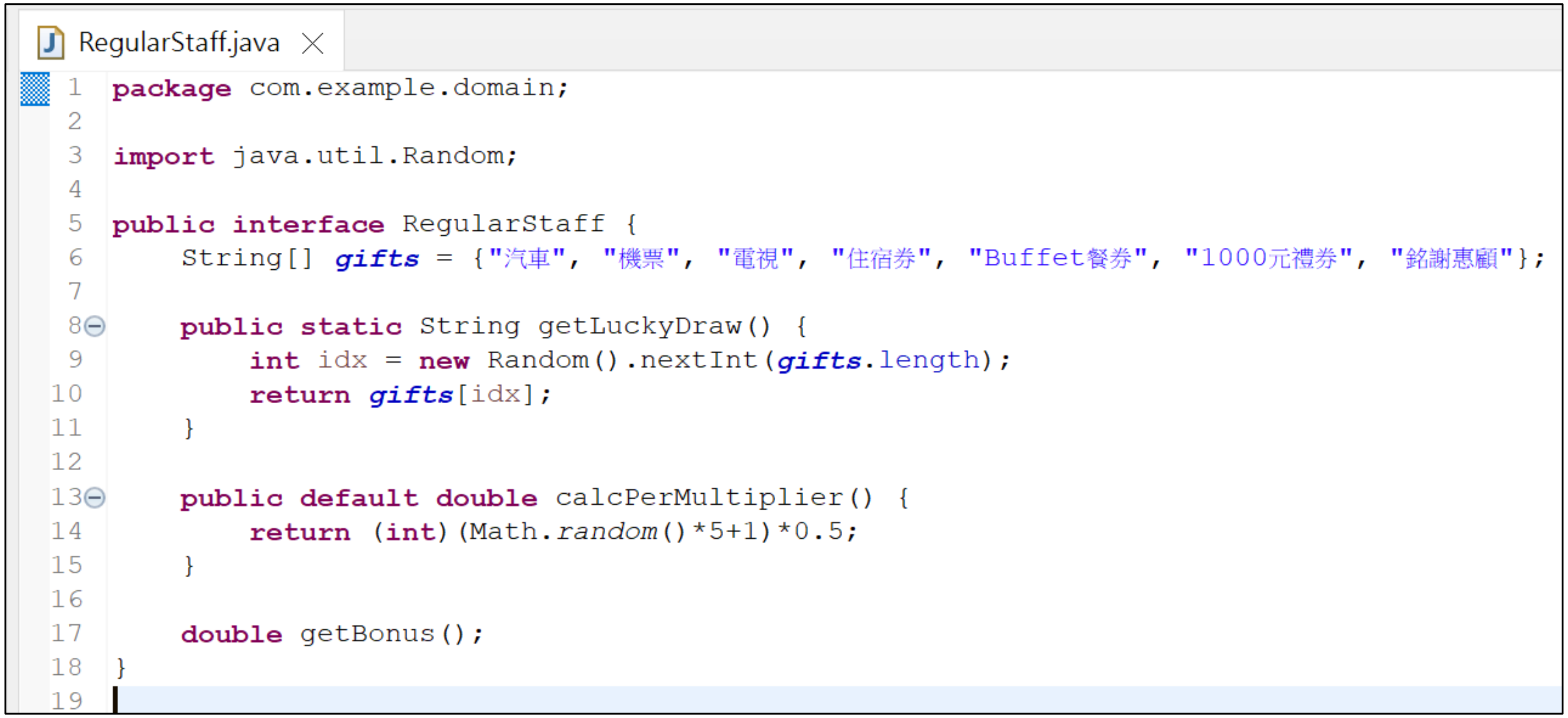
5. 修改 Director 類別
   - 宣告 baseBonus 類別屬性，數值為500,000
   - 實作 getBonus() 方法，傳回 baseBonus * calcPerMultiplier()
6. 修改 EmployeeTest 類別
   - 列印所有正職員工的 bonus 及尾牙摸彩結果
7. 測試執行

練習一



<<interface>>
RegularStaff
- gifts: String[]
+ getLuckDraw() : String
+ calcPerMultiplier() : double
+ getBonus() : double
<<abstract>>
Employee
- nextId : int
- empId : int
- name : String
- ssn : String
- salary : double
+ Employee(name: String, ssn String, salary double)
//method ....
Engineer
- skills : String[ ]
- skillCount : int
+ Engineer(id: int, name: String, ssn String, salary double)
+ getBonus() : double
Manager
- deptName : String
- baseBonus : double
+ Manager(id: int, name: String, ssn String, salary double , dept: String)
+ getBonus() : double
Admin
- hours : int
+ Admin(name: String, ssn: String, salary: double, hours: int)
Director
- budget : double
- baseBonus : double
+ Director(id: int, name: String, ssn String, salary double , dept: String, budget: double)
+ getBonus() : double

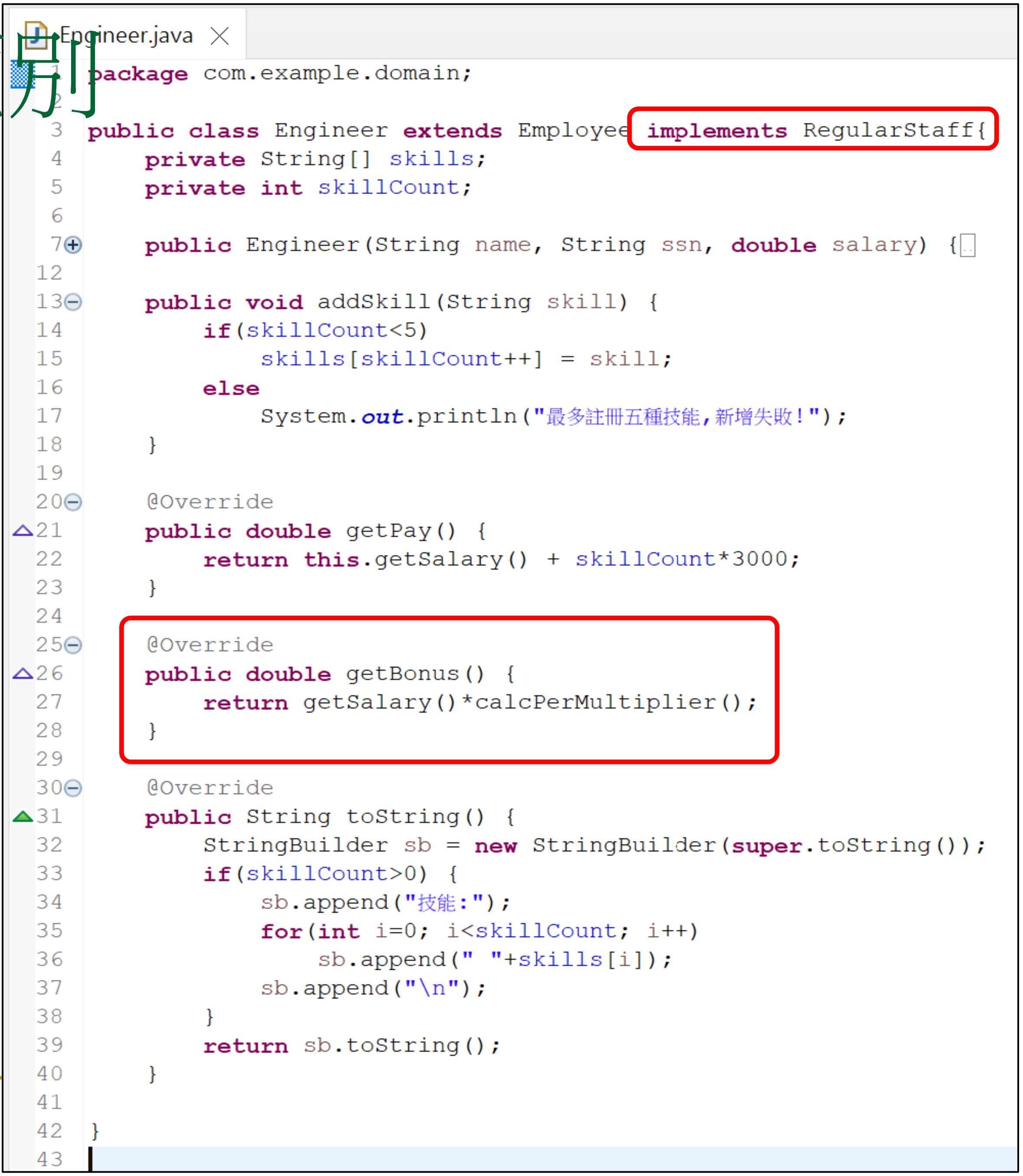
RegularStaff 介面



RegularStaff.java

```java
package com.example.domain;

import java.util.Random;

public interface RegularStaff {
    String[] gifts = {"汽車", "機票", "電視", "住宿券", "Buffet餐券", "1000元禮券", "銘謝惠顧"};

    public static String getLuckyDraw() {
        int idx = new Random().nextInt(gifts.length);
        return gifts[idx];
    }

    public default double calcPerMultiplier() {
        return (int)(Math.random()*5+1)*0.5;
    }

    double getBonus();
}
```

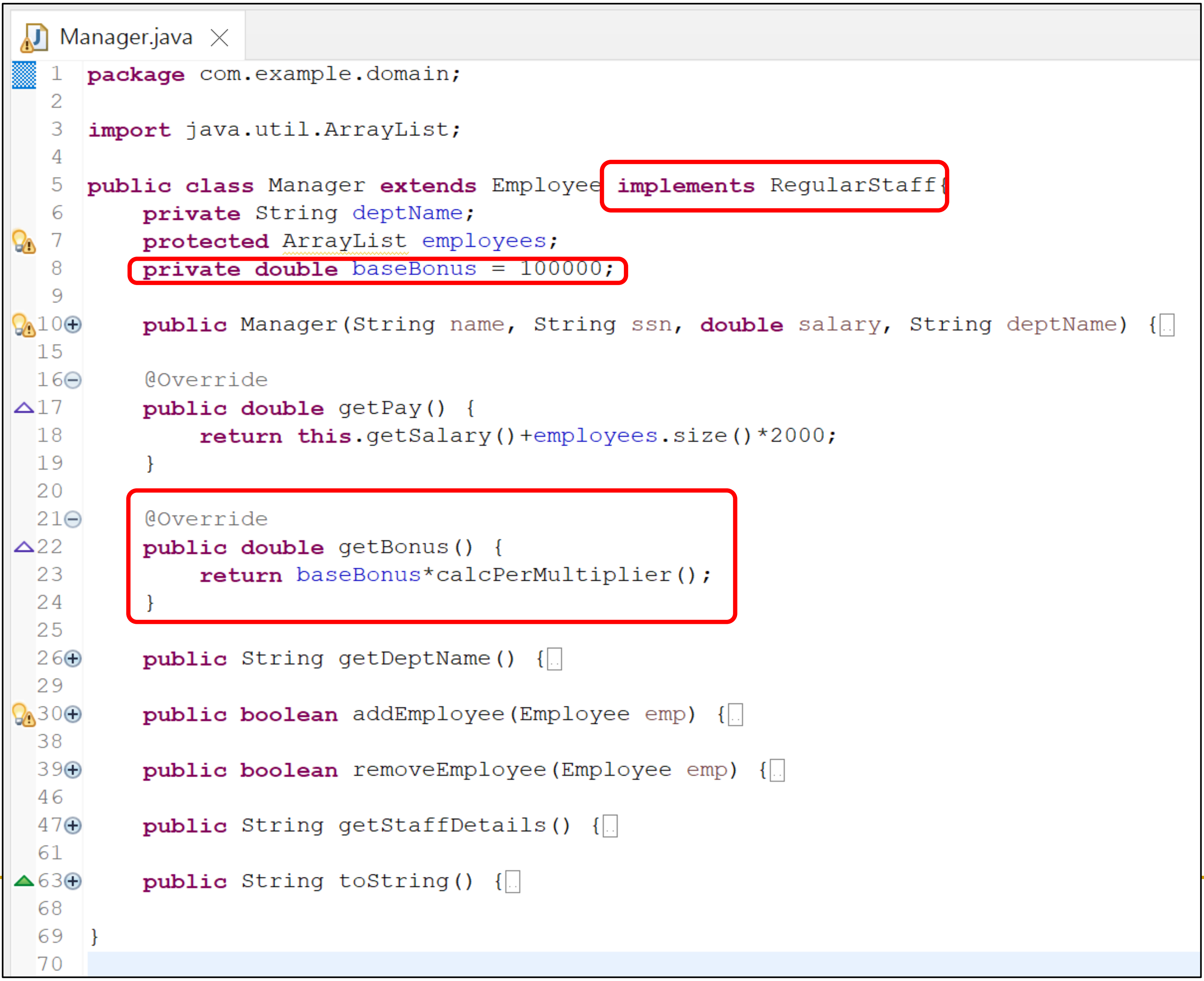
Engineer 類別



Engineer.java

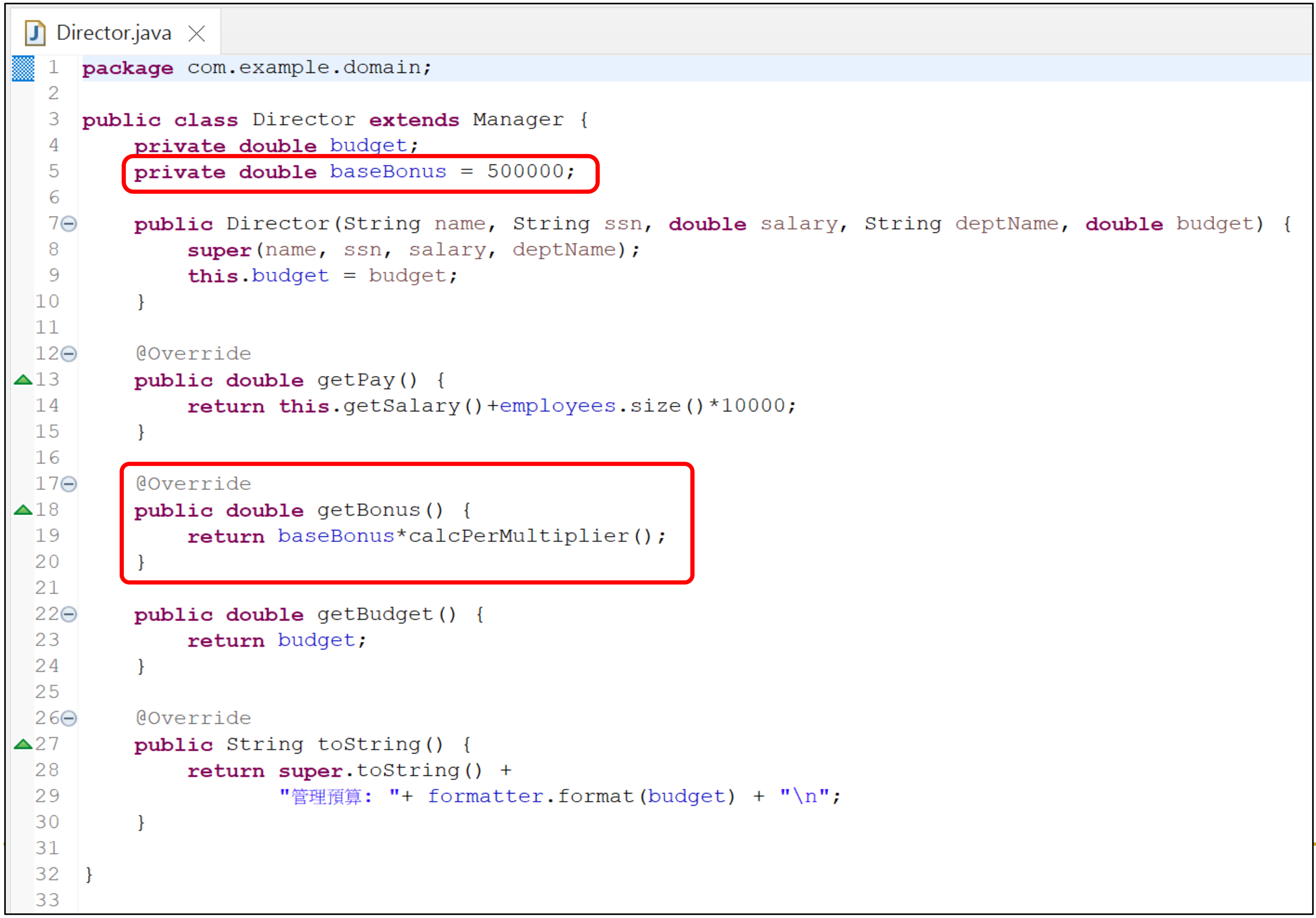
```java
package com.example.domain;

public class Engineer extends Employee implements RegularStaff{
    private String[] skills;
    private int skillCount;

    public Engineer(String name, String ssn, double salary) {

    public void addSkill(String skill) {
        if(skillCount<5)
            skills[skillCount++] = skill;
        else
            System.out.println("最多註冊五種技能,新增失敗!");
    }

    @Override
    public double getPay() {
        return this.getSalary() + skillCount*3000;
    }

    @Override
    public double getBonus() {
        return getSalary()*calcPerMultiplier();
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder(super.toString());
        if(skillCount>0) {
            sb.append("技能:");
            for(int i=0; i<skillCount; i++)
                sb.append(" "+skills[i]);
            sb.append("\n");
        }
        return sb.toString();
    }

}
```

Manager 類別



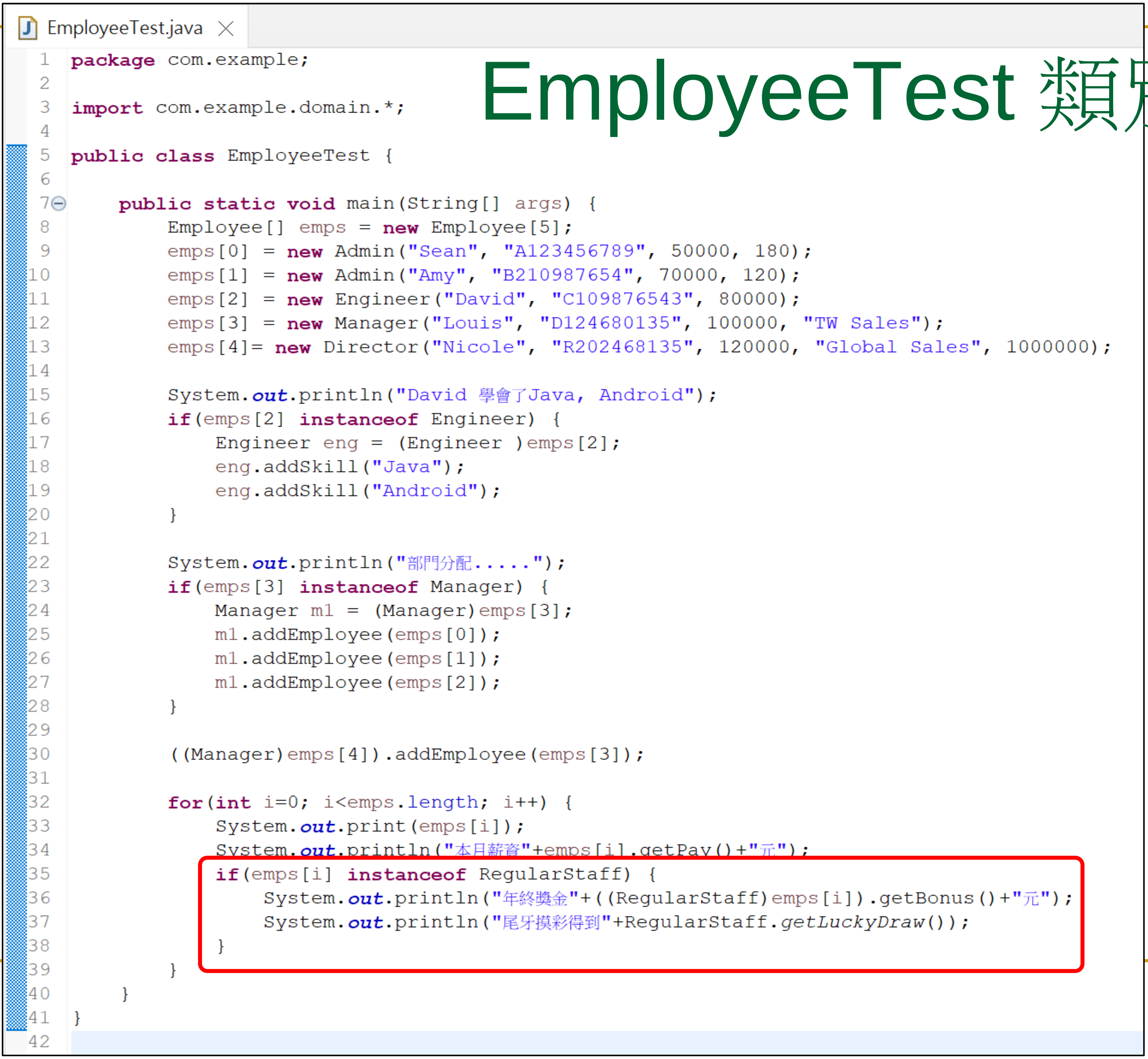
```
package com.example.domain;

import java.util.ArrayList;

public class Manager extends Employee implements RegularStaff{
    private String deptName;
    protected ArrayList employees;
    private double baseBonus = 100000;

    public Manager(String name, String ssn, double salary, String deptName) {

    @Override
    public double getPay() {
        return this.getSalary()+employees.size()*2000;
    }

    @Override
    public double getBonus() {
        return baseBonus*calcPerMultiplier();
    }

    public String getDeptName() {

    public boolean addEmployee(Employee emp) {

    public boolean removeEmployee(Employee emp) {

    public String getStaffDetails() {

    public String toString() {

}
```

Director 類別



```java
package com.example.domain;

public class Director extends Manager {
    private double budget;
    private double baseBonus = 500000;

    public Director(String name, String ssn, double salary, String deptName, double budget) {
        super(name, ssn, salary, deptName);
        this.budget = budget;
    }

    @Override
    public double getPay() {
        return this.getSalary()+employees.size()*10000;
    }

    @Override
    public double getBonus() {
        return baseBonus*calcPerMultiplier();
    }

    public double getBudget() {
        return budget;
    }

    @Override
    public String toString() {
        return super.toString() +
                "管理預算: "+ formatter.format(budget) + "\n";
    }

}
```

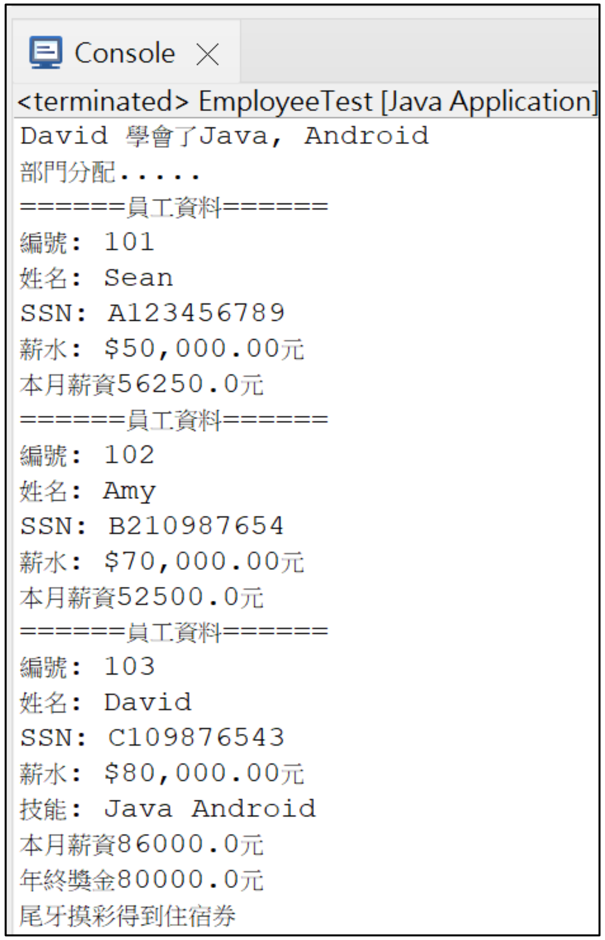
EmployeeTest 類別



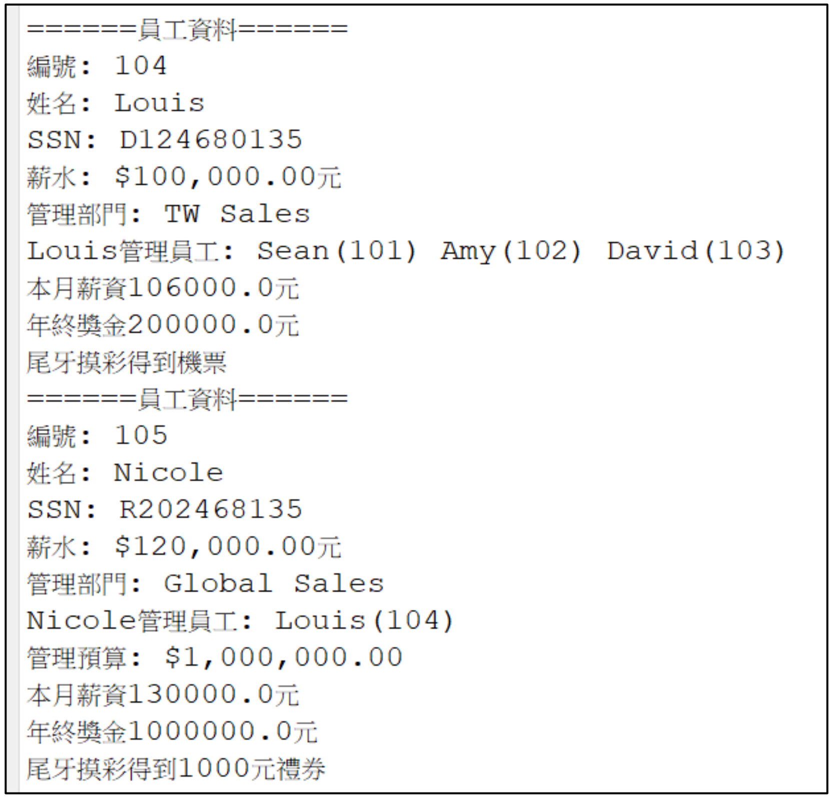
EmployeeTest.java

```java
package com.example;

import com.example.domain.*;

public class EmployeeTest {

    public static void main(String[] args) {
        Employee[] emps = new Employee[5];
        emps[0] = new Admin("Sean", "A123456789", 50000, 180);
        emps[1] = new Admin("Amy", "B210987654", 70000, 120);
        emps[2] = new Engineer("David", "C109876543", 80000);
        emps[3] = new Manager("Louis", "D124680135", 100000, "TW Sales");
        emps[4]= new Director("Nicole", "R202468135", 120000, "Global Sales", 1000000);

        System.out.println("David 學會了Java, Android");
        if(emps[2] instanceof Engineer) {
            Engineer eng = (Engineer )emps[2];
            eng.addSkill("Java");
            eng.addSkill("Android");
        }

        System.out.println("部門分配.....");
        if(emps[3] instanceof Manager) {
            Manager m1 = (Manager)emps[3];
            m1.addEmployee(emps[0]);
            m1.addEmployee(emps[1]);
            m1.addEmployee(emps[2]);
        }

        ((Manager)emps[4]).addEmployee(emps[3]);

        for(int i=0; i<emps.length; i++) {
            System.out.print(emps[i]);
            System.out.println("本月薪資"+emps[i].getPay()+"元");
            if(emps[i] instanceof RegularStaff) {
                System.out.println("年終獎金"+((RegularStaff)emps[i]).getBonus()+"元");
                System.out.println("尾牙摸彩得到"+RegularStaff.getLuckyDraw());
            }
        }
    }
}
```

測試



```
Console ✕
<terminated> EmployeeTest [Java Application]
David 學會了Java, Android
部門分配.....
======員工資料======
編號: 101
姓名: Sean
SSN: A123456789
薪水: $50,000.00元
本月薪資56250.0元
======員工資料======
編號: 102
姓名: Amy
SSN: B210987654
薪水: $70,000.00元
本月薪資52500.0元
======員工資料======
編號: 103
姓名: David
SSN: C109876543
薪水: $80,000.00元
技能: Java Android
本月薪資86000.0元
年終獎金80000.0元
尾牙摸彩得到住宿券
```



```
======員工資料======
編號: 104
姓名: Louis
SSN: D124680135
薪水: $100,000.00元
管理部門: TW Sales
Louis管理員工: Sean(101) Amy(102) David(103)
本月薪資106000.0元
年終獎金200000.0元
尾牙摸彩得到機票
======員工資料======
編號: 105
姓名: Nicole
SSN: R202468135
薪水: $120,000.00元
管理部門: Global Sales
Nicole管理員工: Louis(104)
管理預算: $1,000,000.00
本月薪資130000.0元
年終獎金1000000.0元
尾牙摸彩得到1000元禮券
```

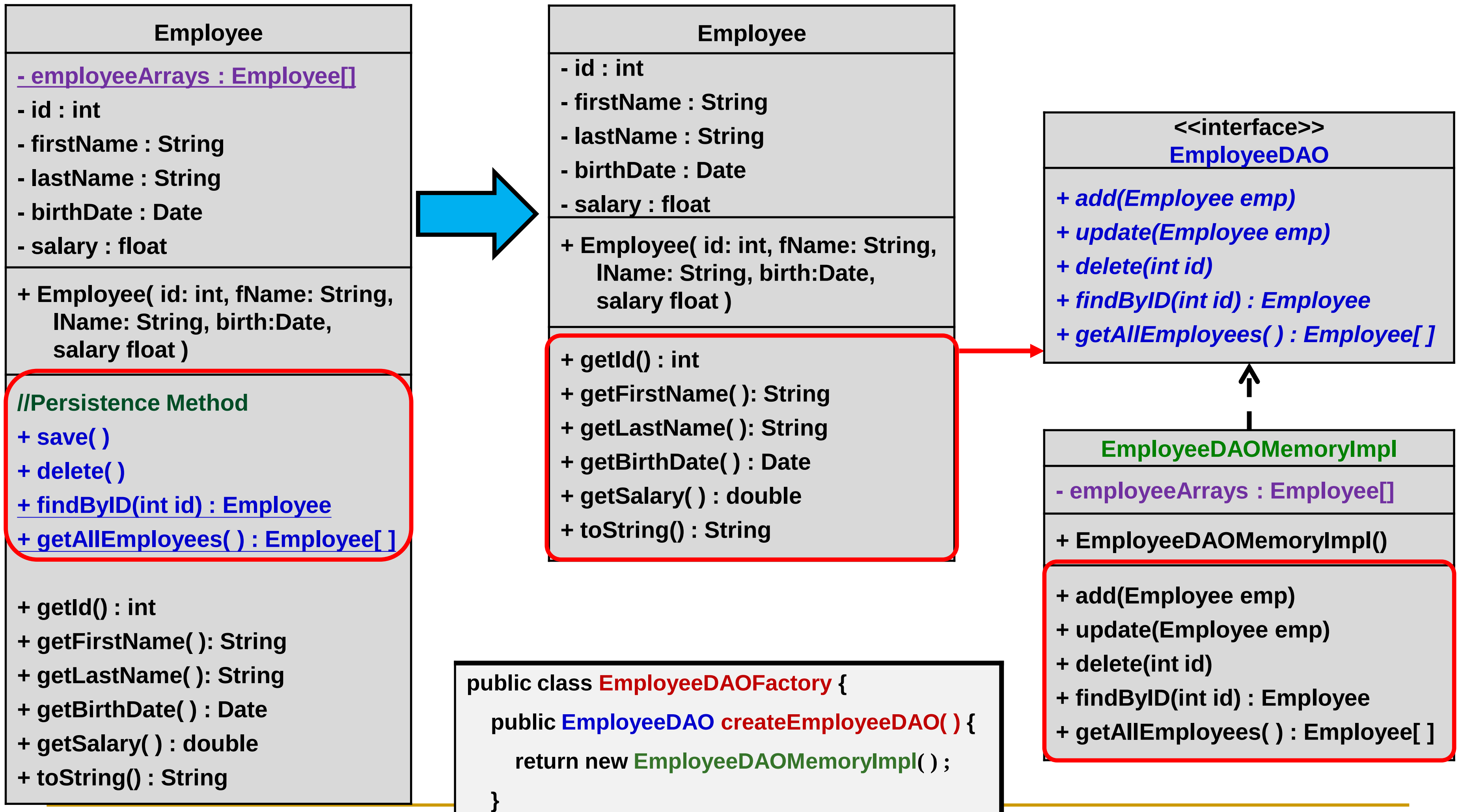
練習二 實作 DAO 模式

1. 開啟 EmployeeDAO 專案
2. 測試、執行
   - 日期格式: Jul 9, 2013
   - EmployeeID: 0~9
3. 檢視Employee.java原始碼
   - private static Employee[] employeeArray = new Employee[10];
   - save()/delete()/findById(int id)/getAllEmployees() 方法
4. 建立com.example.dao.EmployeeDAO介面
   - public void add(Employee emp);
   - public void update(Employee emp);
   - public void delete(int id);
   - public Employee findById(int id);
   - public Employee[] getAllEmployees();

練習二 實作 DAO 模式

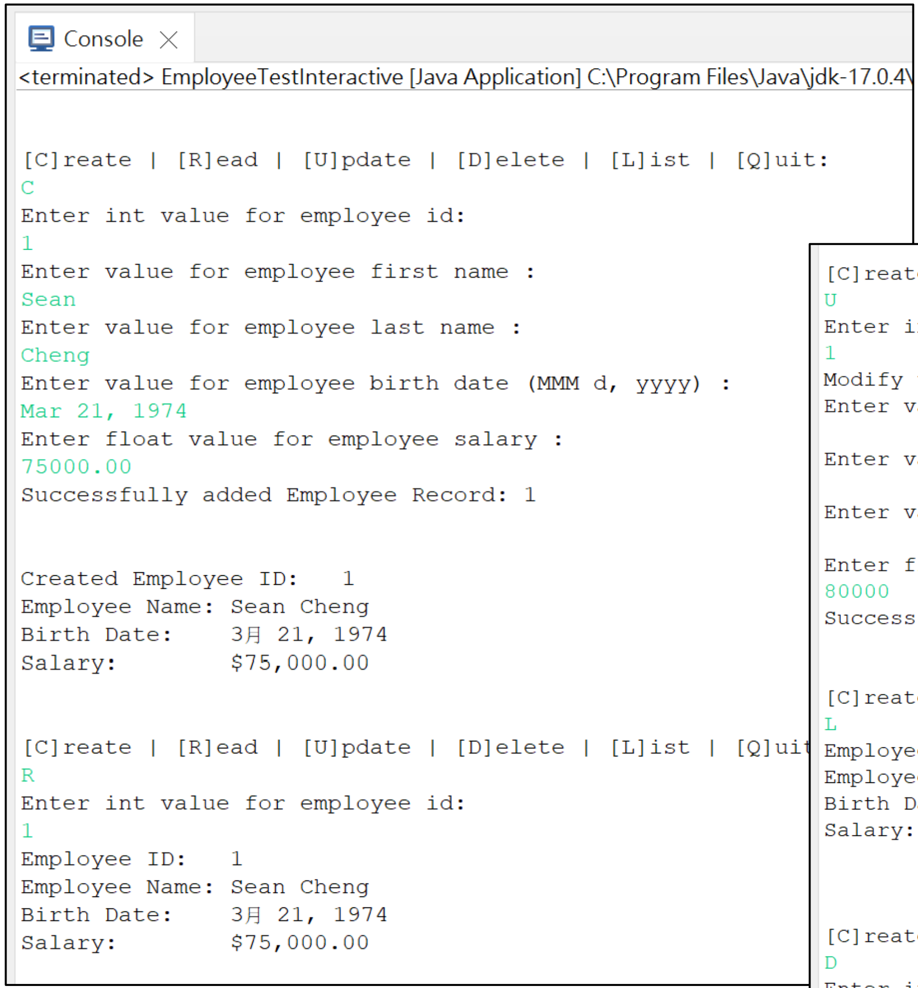
5. 建立com.example.dao.EmployeeDAOMemoryImpl類別
   - 實作EmployeeDAO介面
   - 將Employee類別中employeeArray陣列及操控陣列的方法改由EmployeeDAO介面的方法實作
   - 目前EmployeeDAOMemoryImpl中add()與update()邏輯相同
6. 建立com.example.dao.EmployeeDAOFactory類別
   - 提供 EmployeeDAO createEmployeeDAO() 工廠方法, 傳回EmployeeDAOMemoryImpl物件
7. 修改EmployeeTestInteractive類別
   - 建構EmployeeDAOFactory類別
   - 使用EmployeeDAOFactory類別取得EmployeeDAO物件
   - 修改executeMenu(BufferedReader in), 加入EmployeeDAO傳入參數, 成為executeMenu(BufferedReader in, EmployeeDAO dao)
   - 原emp物件呼叫save()/delete()/findById()/getAllEmployees()方法,以dao物件add()/update()/delete()/findById()/getAllEmployees()取代
8. 測試、執行

練習 實作 DAO 模式



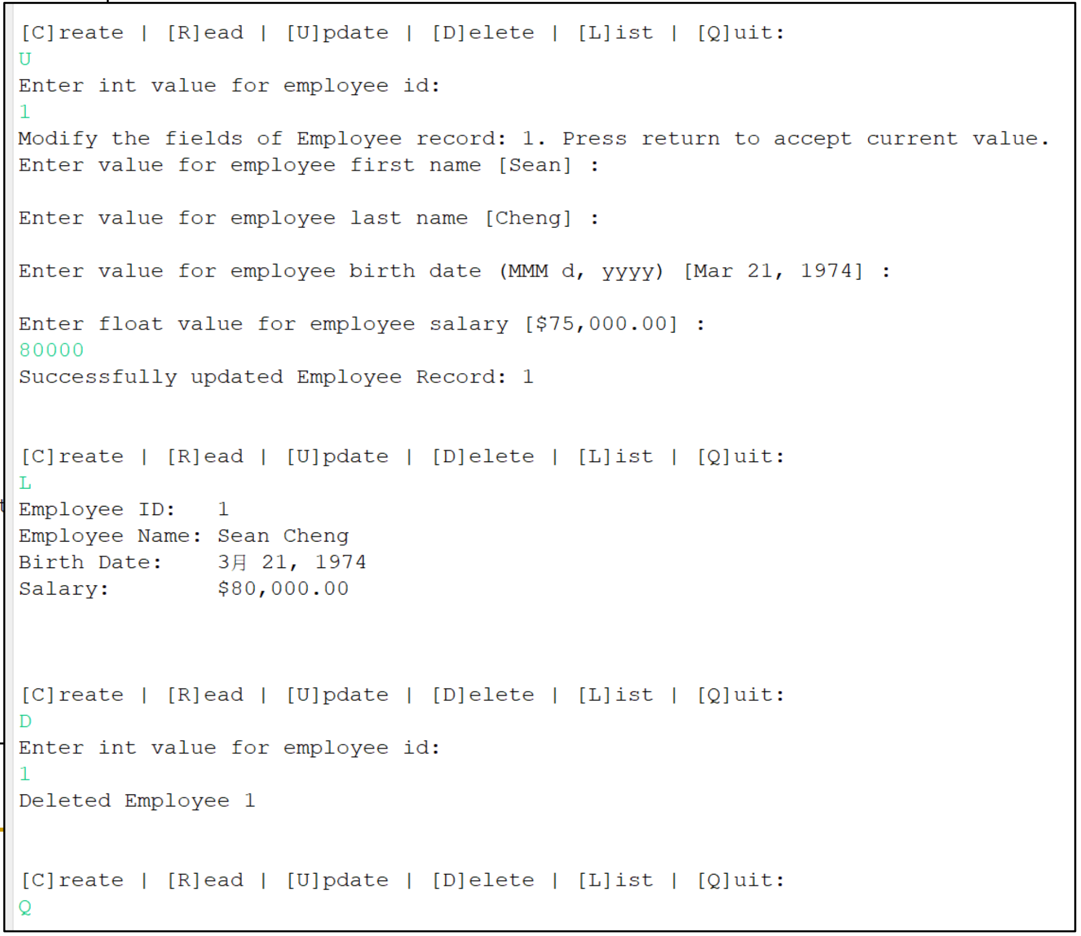
```
public class EmployeeDAOFactory {
    public EmployeeDAO createEmployeeDAO( ) {
        return new EmployeeDAOMemoryImpl( ) ;
    }
}
```

EmployeeMemoryDAO 測試、執行



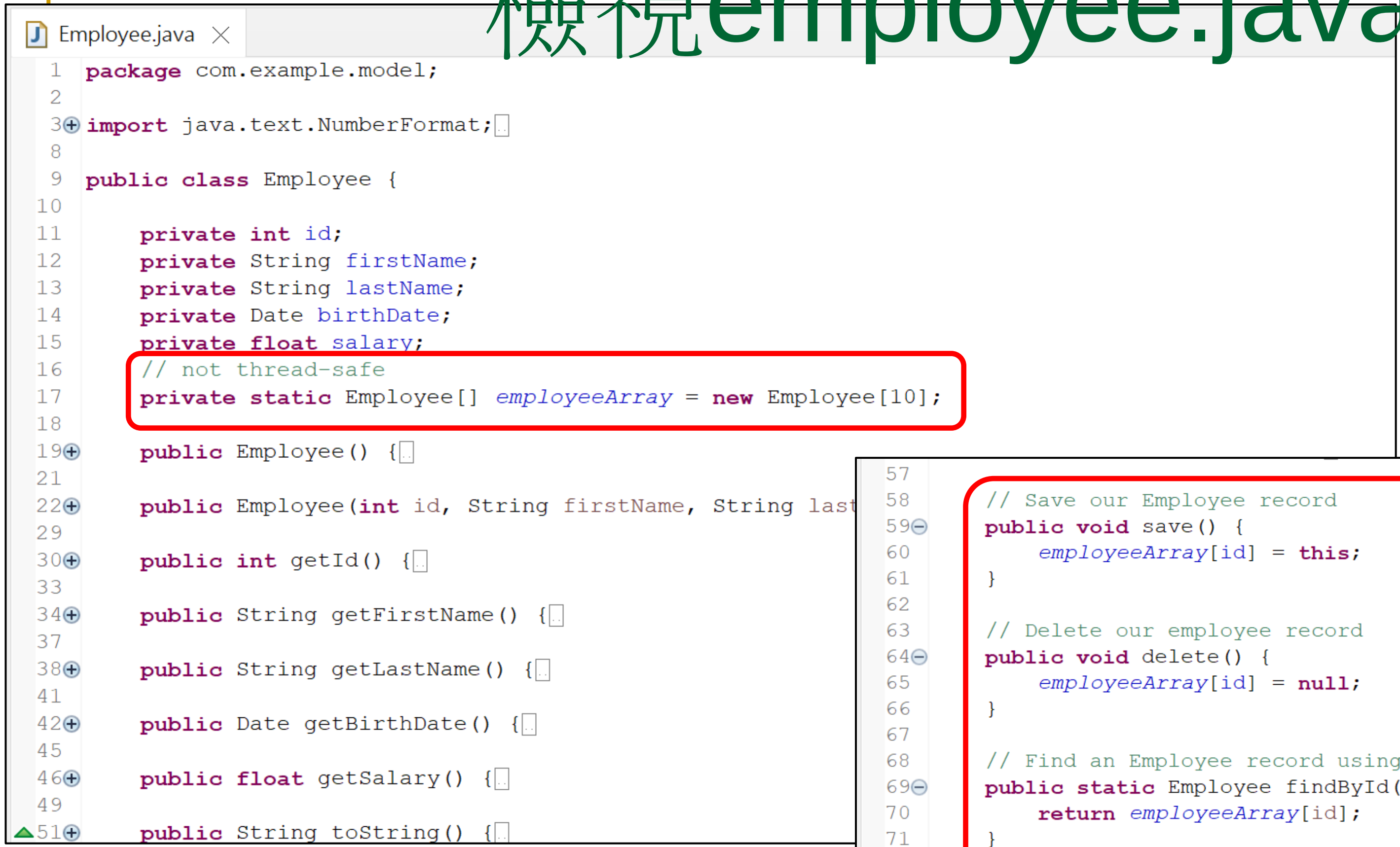
Console

```
<terminated> EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
C
Enter int value for employee id:
1
Enter value for employee first name :
Sean
Enter value for employee last name :
Cheng
Enter value for employee birth date (MMM d, yyyy) :
Mar 21, 1974
Enter float value for employee salary :
75000.00
Successfully added Employee Record: 1


Created Employee ID:   1
Employee Name: Sean Cheng
Birth Date:    3月 21, 1974
Salary:        $75,000.00


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit
R
Enter int value for employee id:
1
Employee ID:   1
Employee Name: Sean Cheng
Birth Date:    3月 21, 1974
Salary:        $75,000.00
```



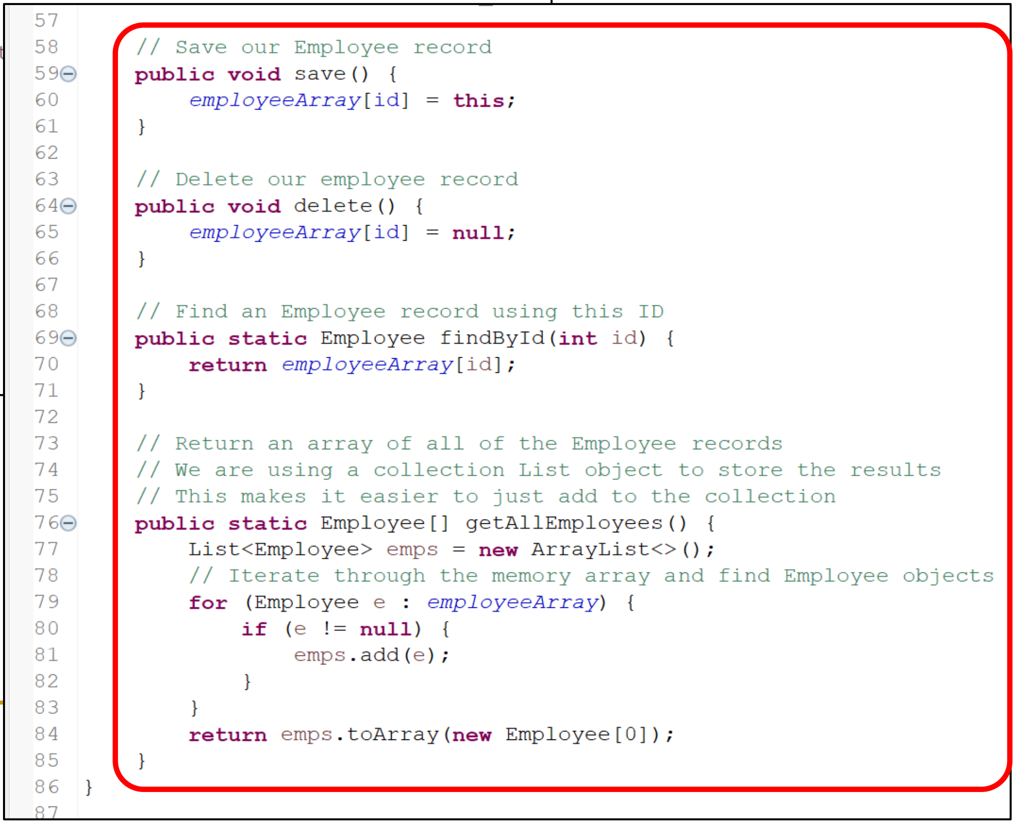
```
[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
U
Enter int value for employee id:
1
Modify the fields of Employee record: 1. Press return to accept current value.
Enter value for employee first name [Sean] :

Enter value for employee last name [Cheng] :

Enter value for employee birth date (MMM d, yyyy) [Mar 21, 1974] :

Enter float value for employee salary [$75,000.00] :
80000
Successfully updated Employee Record: 1


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
L
Employee ID:   1
Employee Name: Sean Cheng
Birth Date:    3月 21, 1974
Salary:        $80,000.00



[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
D
Enter int value for employee id:
1
Deleted Employee 1


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
Q
```

檢視employee.java原始碼



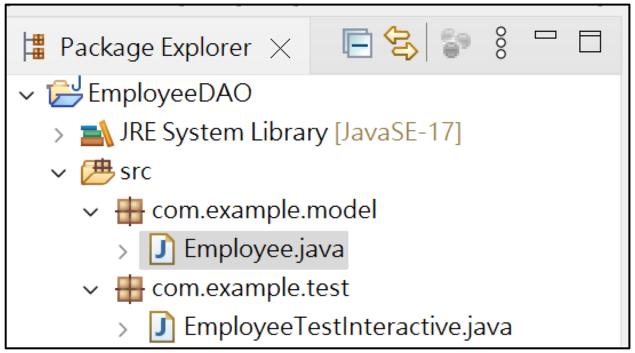
Employee.java

```
package com.example.model;

import java.text.NumberFormat;

public class Employee {

    private int id;
    private String firstName;
    private String lastName;
    private Date birthDate;
    private float salary;
    // not thread-safe
    private static Employee[] employeeArray = new Employee[10];

    public Employee() {

    public Employee(int id, String firstName, String last

    public int getId() {

    public String getFirstName() {

    public String getLastName() {

    public Date getBirthDate() {

    public float getSalary() {

    public String toString() {
```



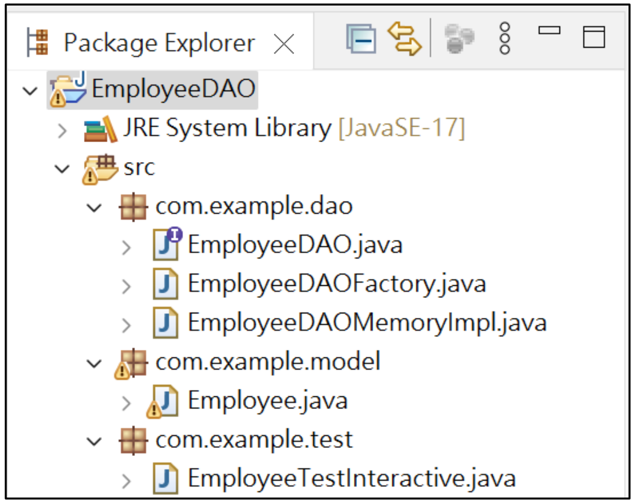
```

    // Save our Employee record
    public void save() {
        employeeArray[id] = this;
    }

    // Delete our employee record
    public void delete() {
        employeeArray[id] = null;
    }

    // Find an Employee record using this ID
    public static Employee findById(int id) {
        return employeeArray[id];
    }

    // Return an array of all of the Employee records
    // We are using a collection List object to store the results
    // This makes it easier to just add to the collection
    public static Employee[] getAllEmployees() {
        List<Employee> emps = new ArrayList<>();
        // Iterate through the memory array and find Employee objects
        for (Employee e : employeeArray) {
            if (e != null) {
                emps.add(e);
            }
        }
        return emps.toArray(new Employee[0]);
    }
}
```

新增com.example.dao.EmployeeDAO



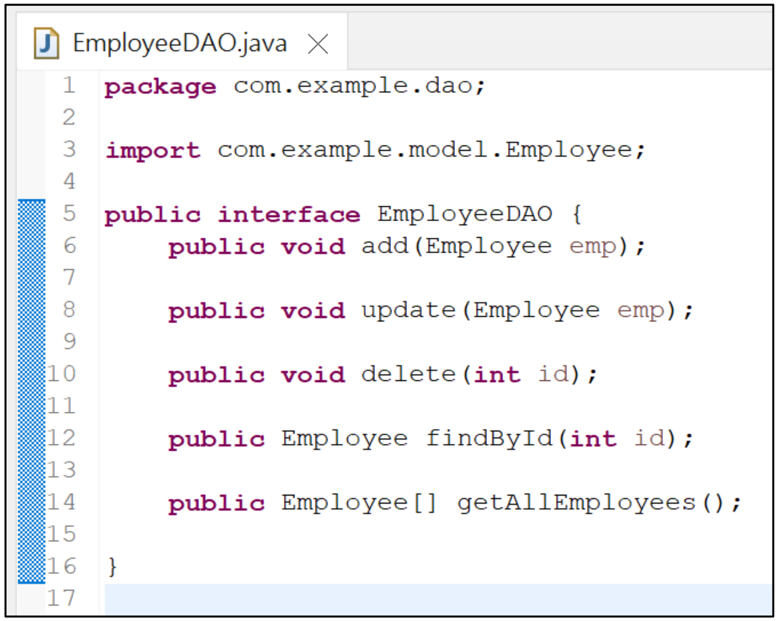
Package Explorer

```
EmployeeDAO
  JRE System Library [JavaSE-17]
  src
    com.example.model
      Employee.java
    com.example.test
      EmployeeTestInteractive.java
```



Package Explorer

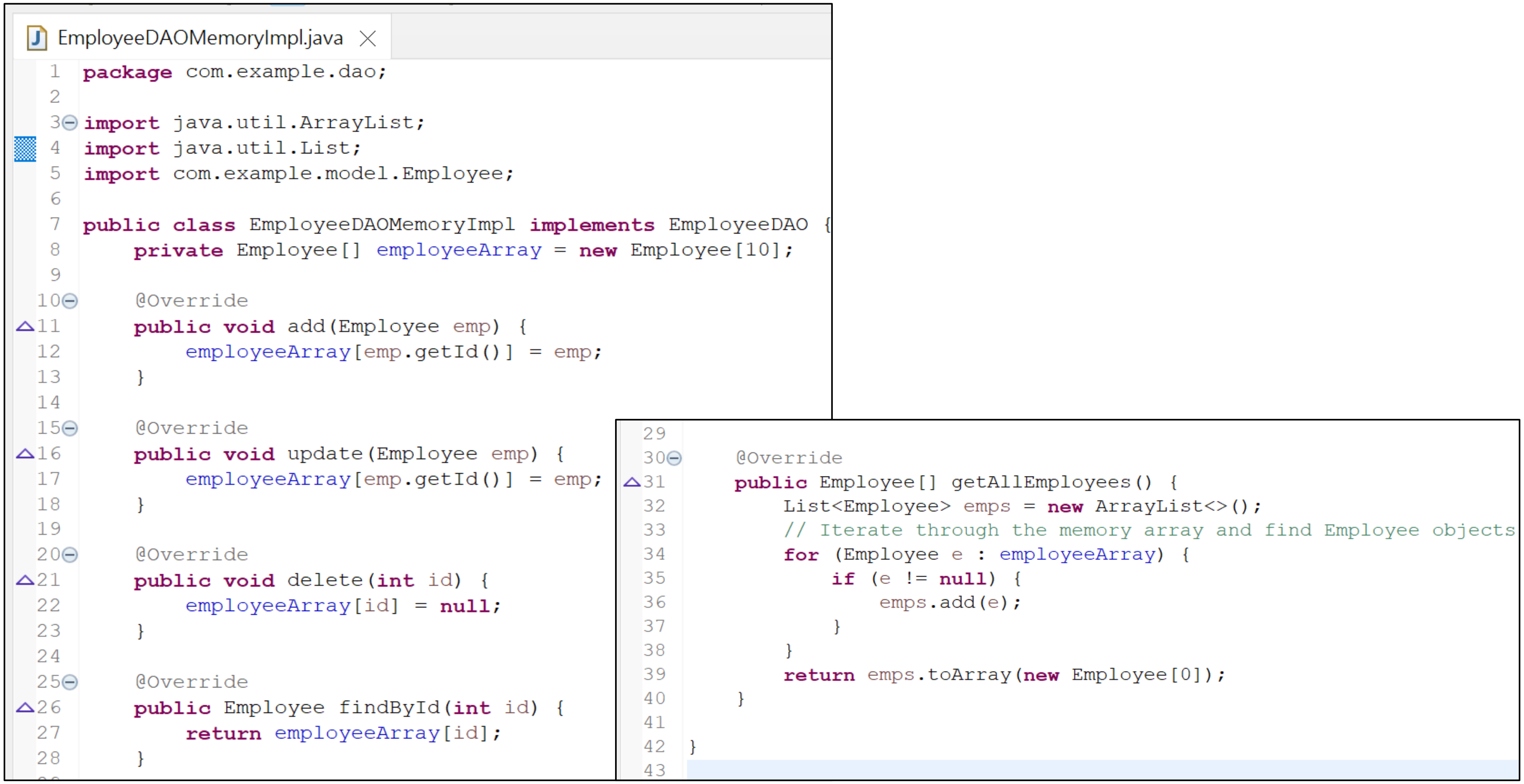
```
EmployeeDAO
  JRE System Library [JavaSE-17]
  src
    com.example.dao
      EmployeeDAO.java
      EmployeeDAOFactory.java
      EmployeeDAOMemoryImpl.java
    com.example.model
      Employee.java
    com.example.test
      EmployeeTestInteractive.java
```



EmployeeDAO.java

```java
package com.example.dao;

import com.example.model.Employee;

public interface EmployeeDAO {
    public void add(Employee emp);

    public void update(Employee emp);

    public void delete(int id);

    public Employee findById(int id);

    public Employee[] getAllEmployees();

}

```

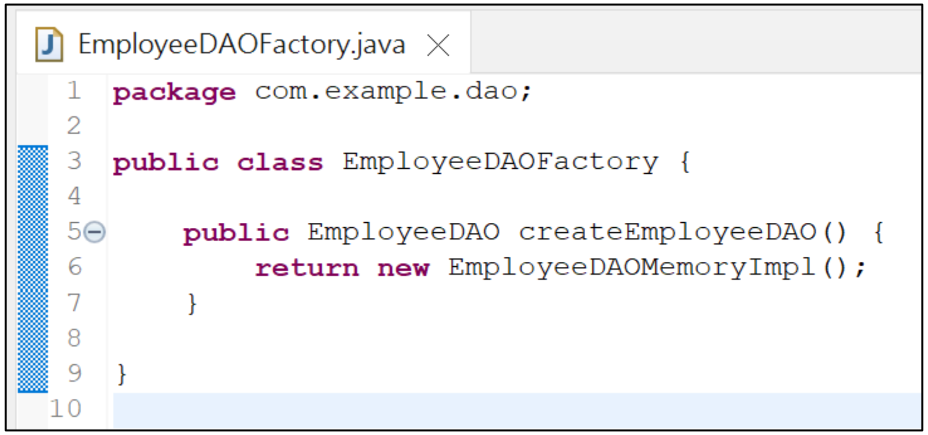
新增 EmployeeDAOMemoryImpl



EmployeeDAOMemoryImpl.java

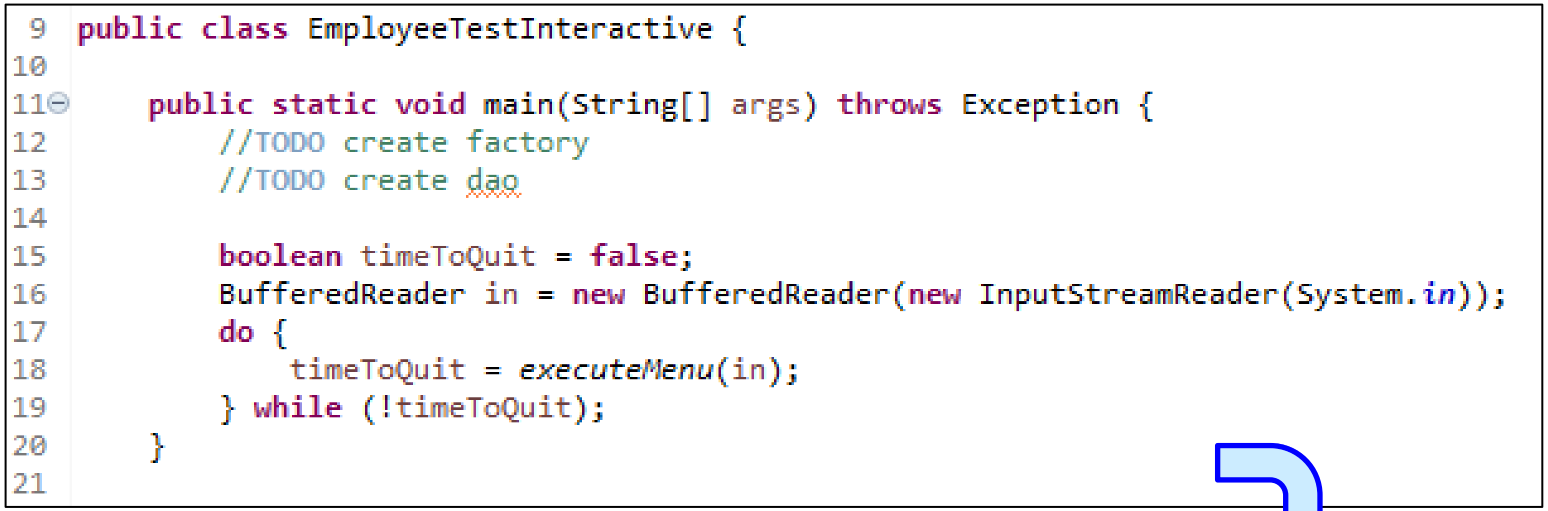
```java
package com.example.dao;

import java.util.ArrayList;
import java.util.List;
import com.example.model.Employee;

public class EmployeeDAOMemoryImpl implements EmployeeDAO {
    private Employee[] employeeArray = new Employee[10];

    @Override
    public void add(Employee emp) {
        employeeArray[emp.getId()] = emp;
    }

    @Override
    public void update(Employee emp) {
        employeeArray[emp.getId()] = emp;
    }

    @Override
    public void delete(int id) {
        employeeArray[id] = null;
    }

    @Override
    public Employee findById(int id) {
        return employeeArray[id];
    }

    @Override
    public Employee[] getAllEmployees() {
        List<Employee> emps = new ArrayList<>();
        // Iterate through the memory array and find Employee objects
        for (Employee e : employeeArray) {
            if (e != null) {
                emps.add(e);
            }
        }
        return emps.toArray(new Employee[0]);
    }

}
```

新增 EmployeeDAOFactory

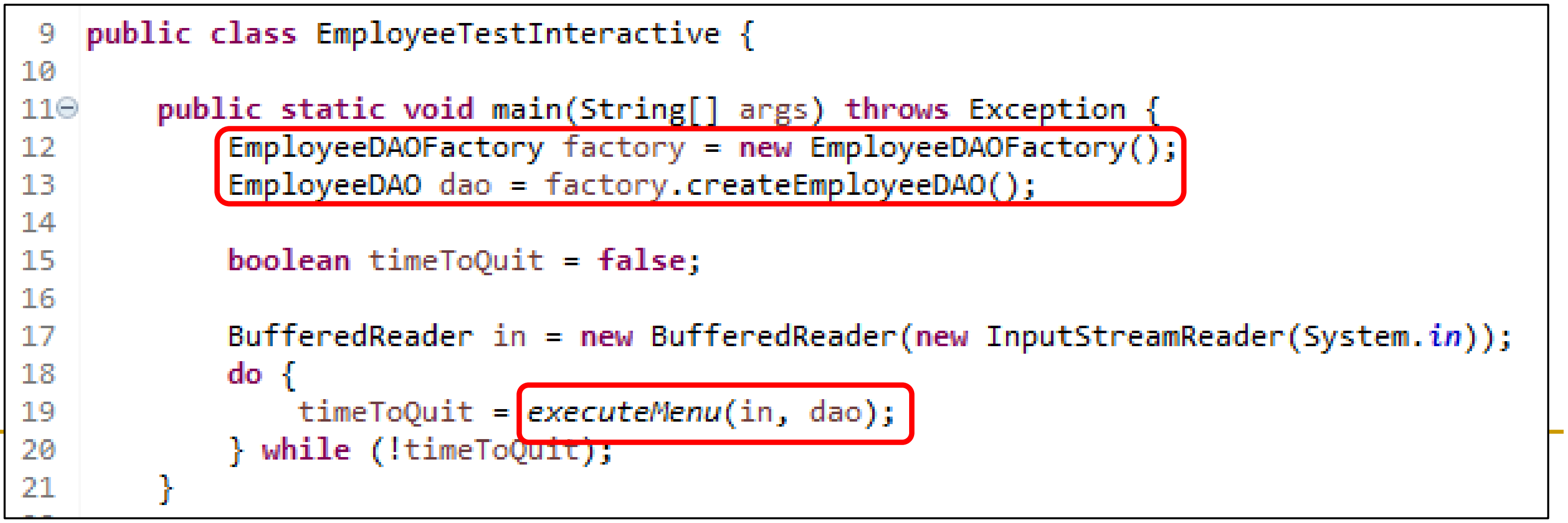


```java
package com.example.dao;

public class EmployeeDAOFactory {

    public EmployeeDAO createEmployeeDAO() {
        return new EmployeeDAOMemoryImpl();
    }

}
```

修改 EmployeeTestInteractive

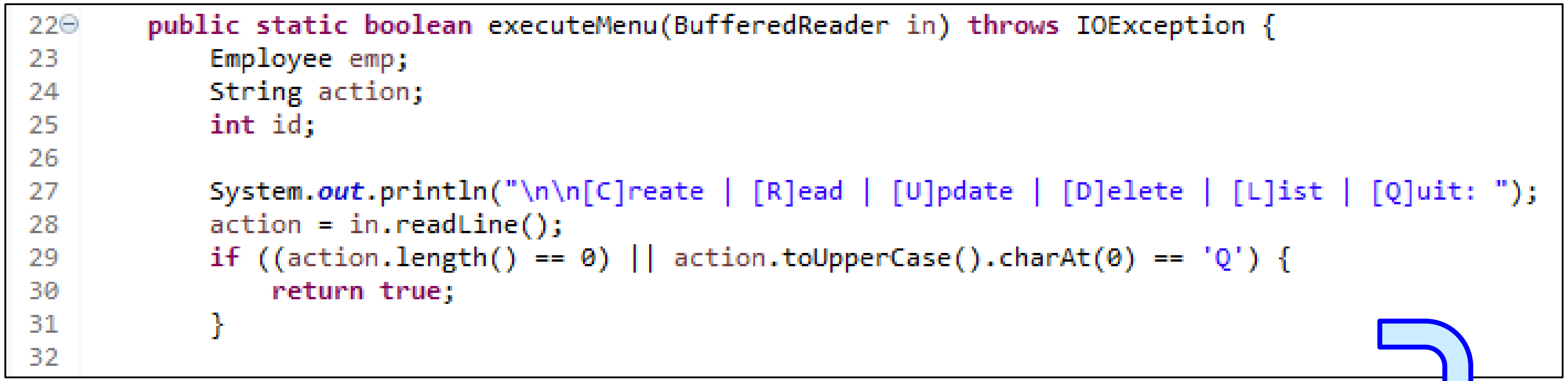


```
public class EmployeeTestInteractive {

    public static void main(String[] args) throws Exception {
        //TODO create factory
        //TODO create dao

        boolean timeToQuit = false;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        do {
            timeToQuit = executeMenu(in);
        } while (!timeToQuit);
    }

```

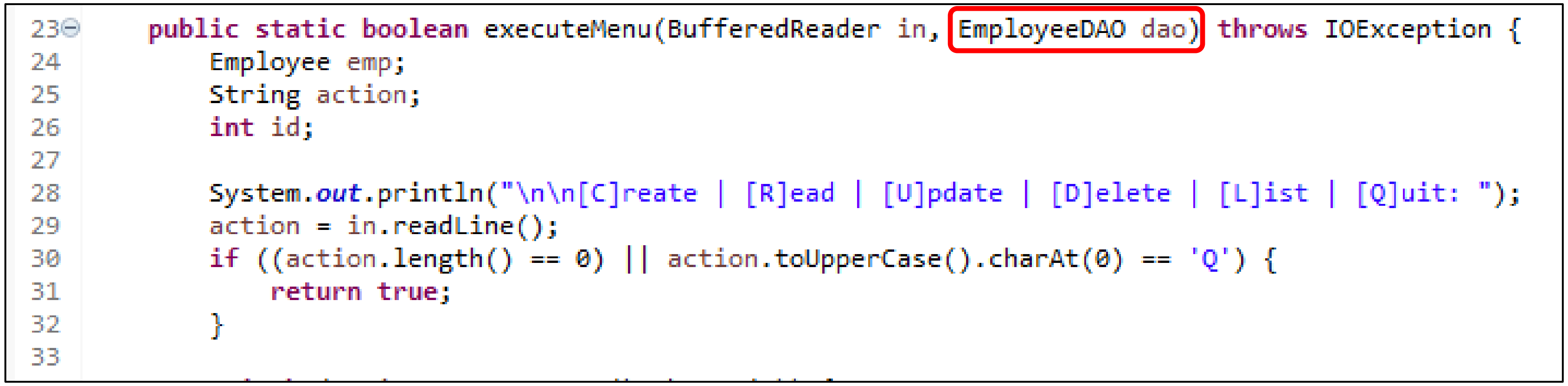


```
public class EmployeeTestInteractive {

    public static void main(String[] args) throws Exception {
        EmployeeDAOFactory factory = new EmployeeDAOFactory();
        EmployeeDAO dao = factory.createEmployeeDAO();

        boolean timeToQuit = false;

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        do {
            timeToQuit = executeMenu(in, dao);
        } while (!timeToQuit);
    }
```

修改 EmployeeTestInteractive

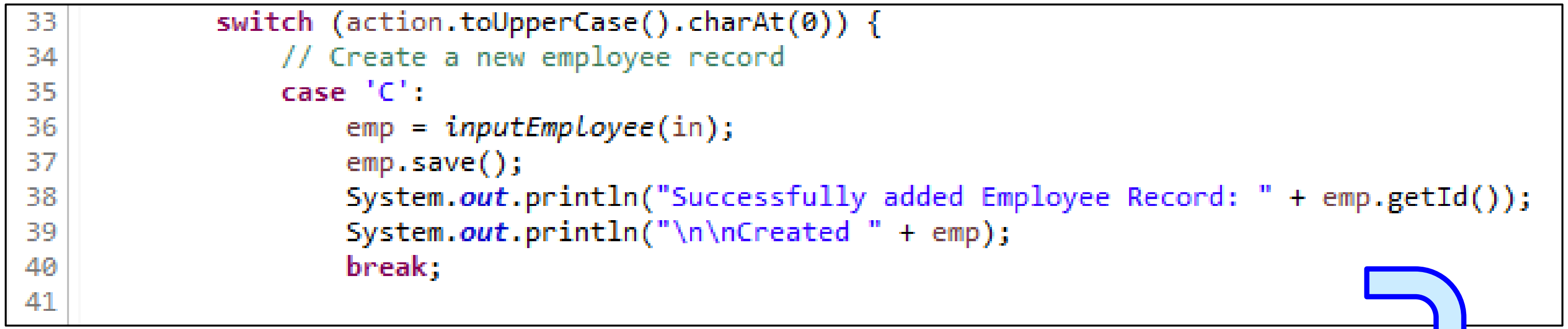


```
    public static boolean executeMenu(BufferedReader in) throws IOException {
        Employee emp;
        String action;
        int id;

        System.out.println("\n\n[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit: ");
        action = in.readLine();
        if ((action.length() == 0) || action.toUpperCase().charAt(0) == 'Q') {
            return true;
        }

```

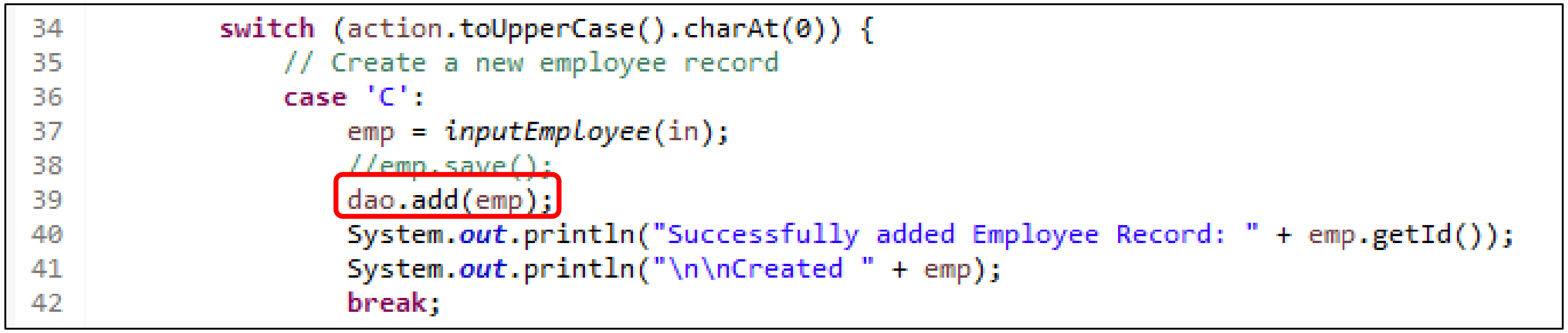


```
    public static boolean executeMenu(BufferedReader in, EmployeeDAO dao) throws IOException {
        Employee emp;
        String action;
        int id;

        System.out.println("\n\n[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit: ");
        action = in.readLine();
        if ((action.length() == 0) || action.toUpperCase().charAt(0) == 'Q') {
            return true;
        }

```

修改 EmployeeTestInteractive

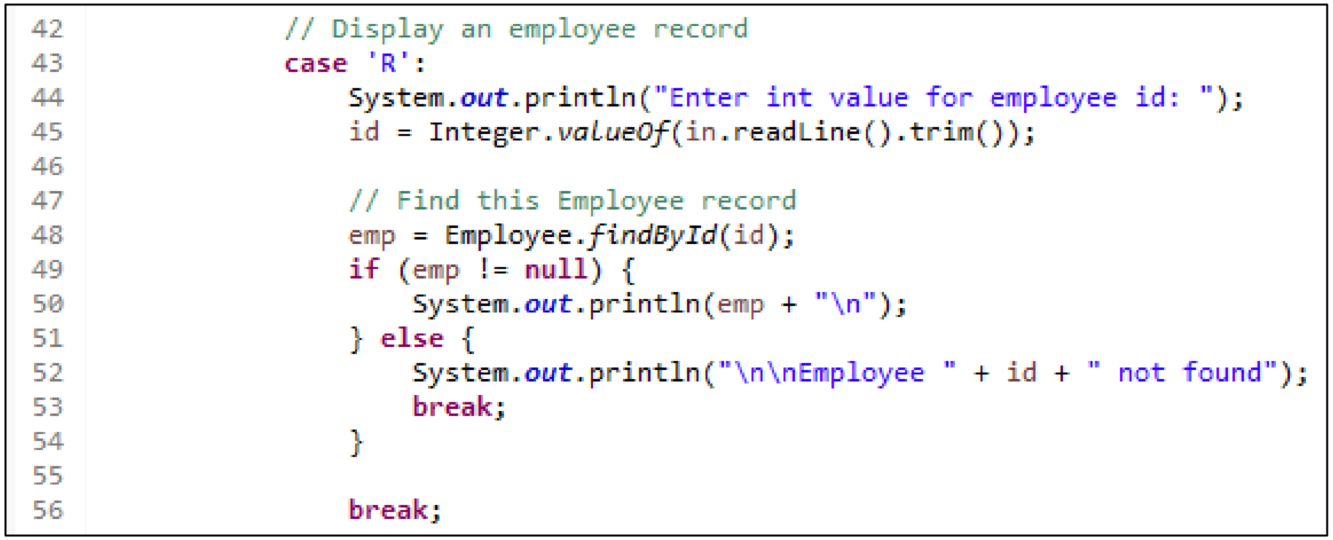


```java
        switch (action.toUpperCase().charAt(0)) {
            // Create a new employee record
            case 'C':
                emp = inputEmployee(in);
                emp.save();
                System.out.println("Successfully added Employee Record: " + emp.getId());
                System.out.println("\n\nCreated " + emp);
                break;

```

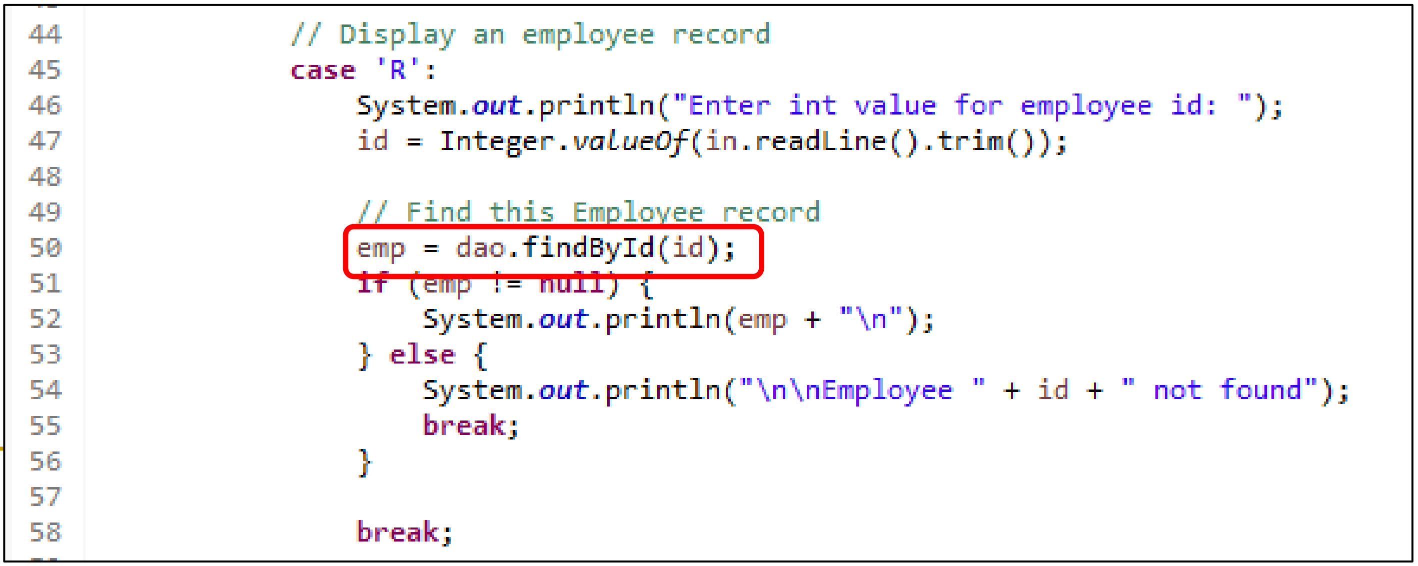


```java
        switch (action.toUpperCase().charAt(0)) {
            // Create a new employee record
            case 'C':
                emp = inputEmployee(in);
                //emp.save();
                dao.add(emp);
                System.out.println("Successfully added Employee Record: " + emp.getId());
                System.out.println("\n\nCreated " + emp);
                break;
```

修改 EmployeeTestInteractive

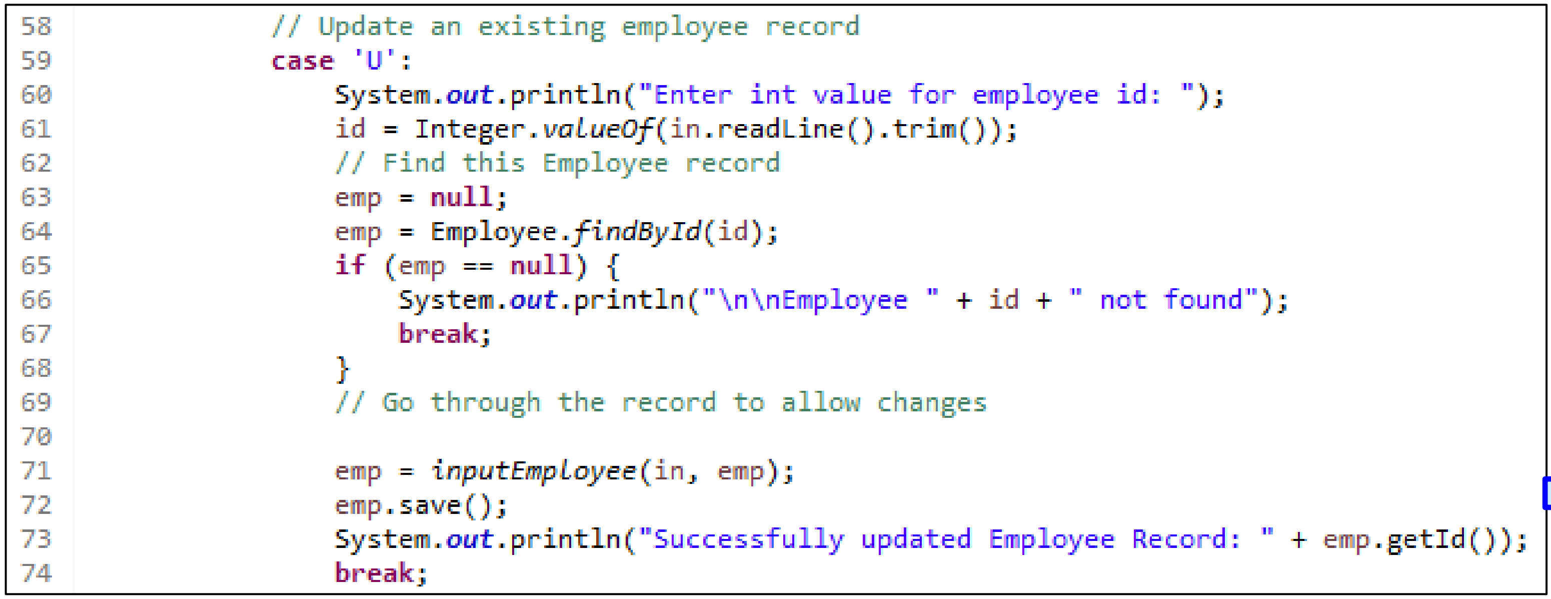


```
            // Display an employee record
            case 'R':
                System.out.println("Enter int value for employee id: ");
                id = Integer.valueOf(in.readLine().trim());

                // Find this Employee record
                emp = Employee.findById(id);
                if (emp != null) {
                    System.out.println(emp + "\n");
                } else {
                    System.out.println("\n\nEmployee " + id + " not found");
                    break;
                }

                break;
```

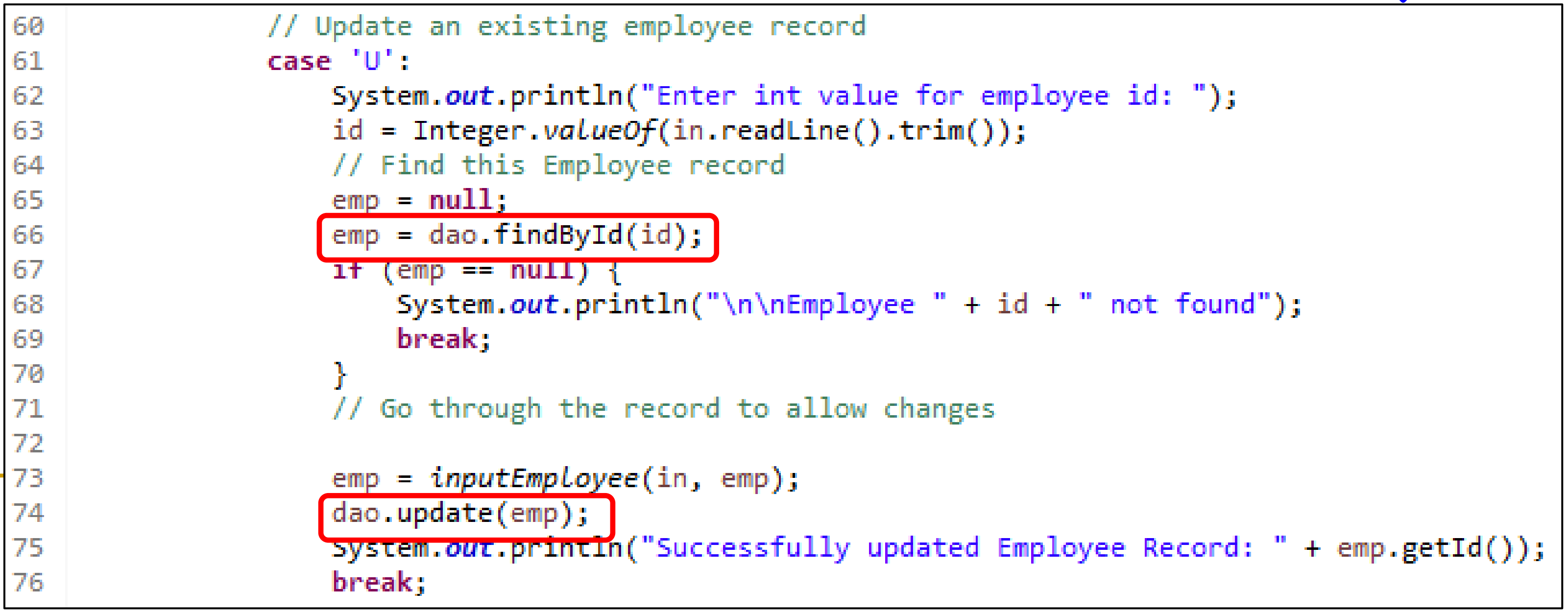


```
            // Display an employee record
            case 'R':
                System.out.println("Enter int value for employee id: ");
                id = Integer.valueOf(in.readLine().trim());

                // Find this Employee record
                emp = dao.findById(id);
                if (emp != null) {
                    System.out.println(emp + "\n");
                } else {
                    System.out.println("\n\nEmployee " + id + " not found");
                    break;
                }

                break;
```

修改 EmployeeTestInteractive

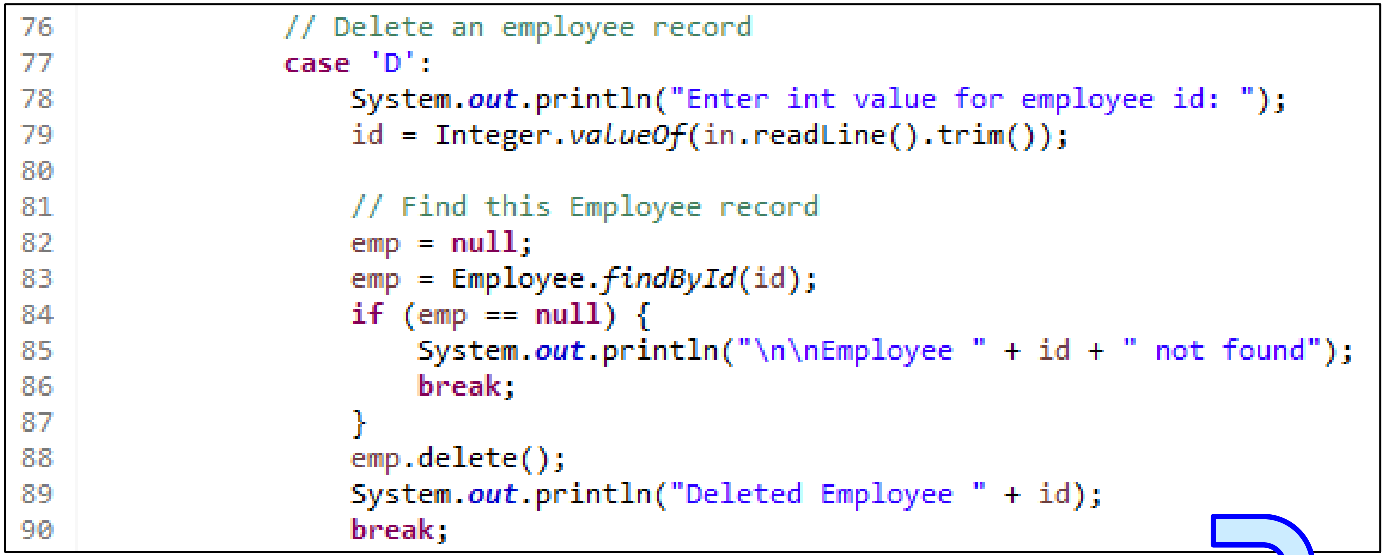


```java
// Update an existing employee record
case 'U':
    System.out.println("Enter int value for employee id: ");
    id = Integer.valueOf(in.readLine().trim());
    // Find this Employee record
    emp = null;
    emp = Employee.findById(id);
    if (emp == null) {
        System.out.println("\n\nEmployee " + id + " not found");
        break;
    }
    // Go through the record to allow changes

    emp = inputEmployee(in, emp);
    emp.save();
    System.out.println("Successfully updated Employee Record: " + emp.getId());
    break;
```

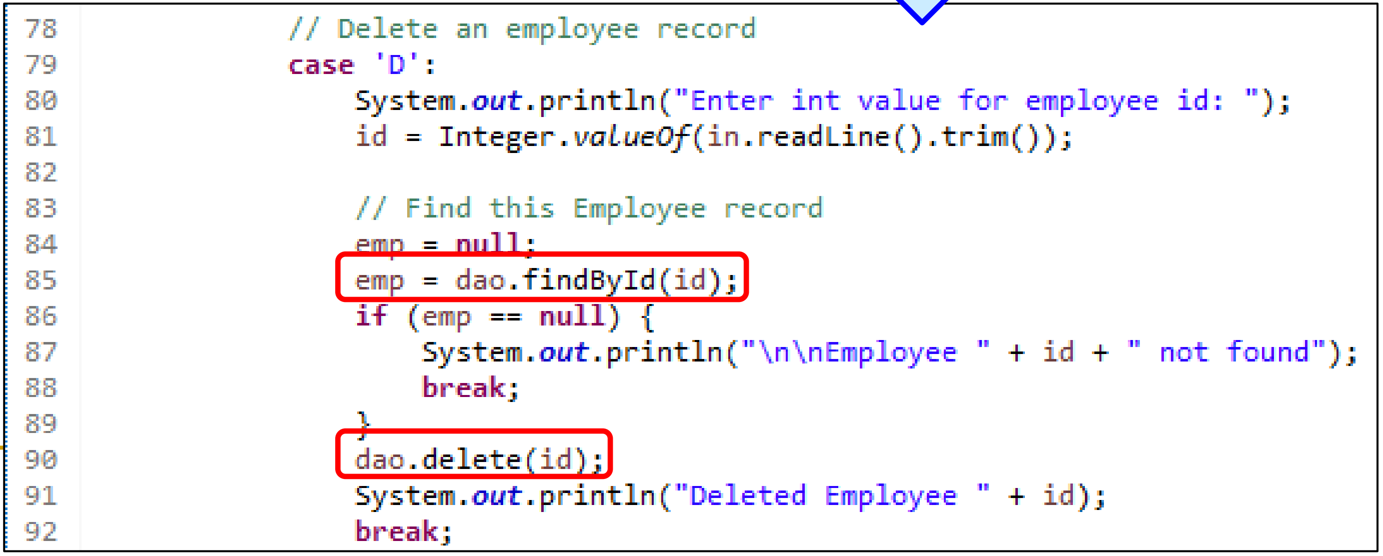


```java
// Update an existing employee record
case 'U':
    System.out.println("Enter int value for employee id: ");
    id = Integer.valueOf(in.readLine().trim());
    // Find this Employee record
    emp = null;
    emp = dao.findById(id);
    if (emp == null) {
        System.out.println("\n\nEmployee " + id + " not found");
        break;
    }
    // Go through the record to allow changes

    emp = inputEmployee(in, emp);
    dao.update(emp);
    System.out.println("Successfully updated Employee Record: " + emp.getId());
    break;
```

修改 EmployeeTestInteractive

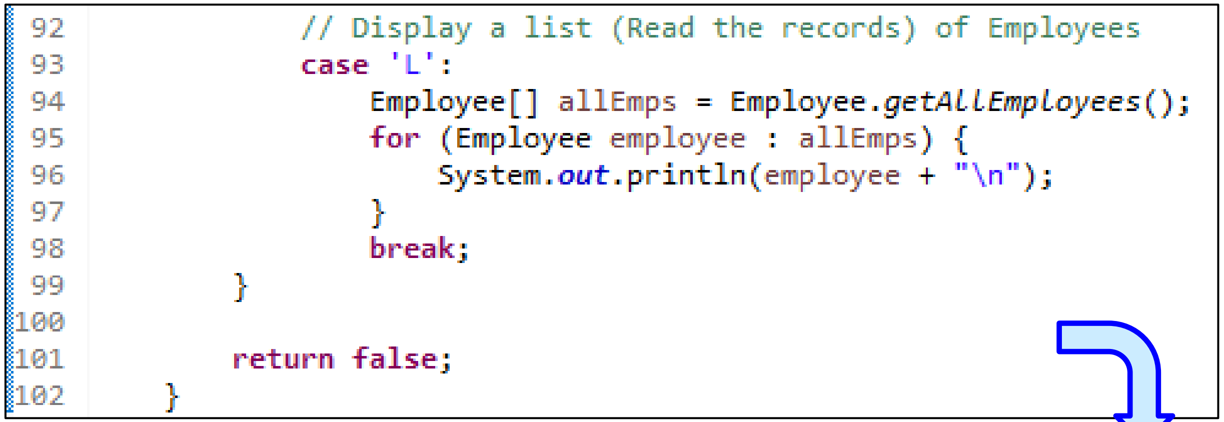


```
            // Delete an employee record
            case 'D':
                System.out.println("Enter int value for employee id: ");
                id = Integer.valueOf(in.readLine().trim());

                // Find this Employee record
                emp = null;
                emp = Employee.findById(id);
                if (emp == null) {
                    System.out.println("\n\nEmployee " + id + " not found");
                    break;
                }
                emp.delete();
                System.out.println("Deleted Employee " + id);
                break;
```

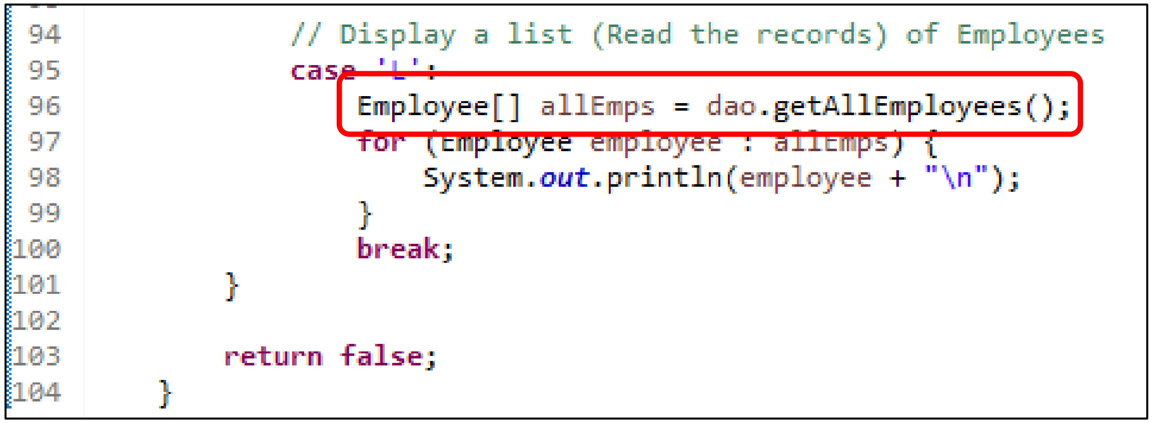


```
            // Delete an employee record
            case 'D':
                System.out.println("Enter int value for employee id: ");
                id = Integer.valueOf(in.readLine().trim());

                // Find this Employee record
                emp = null;
                emp = dao.findById(id);
                if (emp == null) {
                    System.out.println("\n\nEmployee " + id + " not found");
                    break;
                }
                dao.delete(id);
                System.out.println("Deleted Employee " + id);
                break;
```

修改 EmployeeTestInteractive

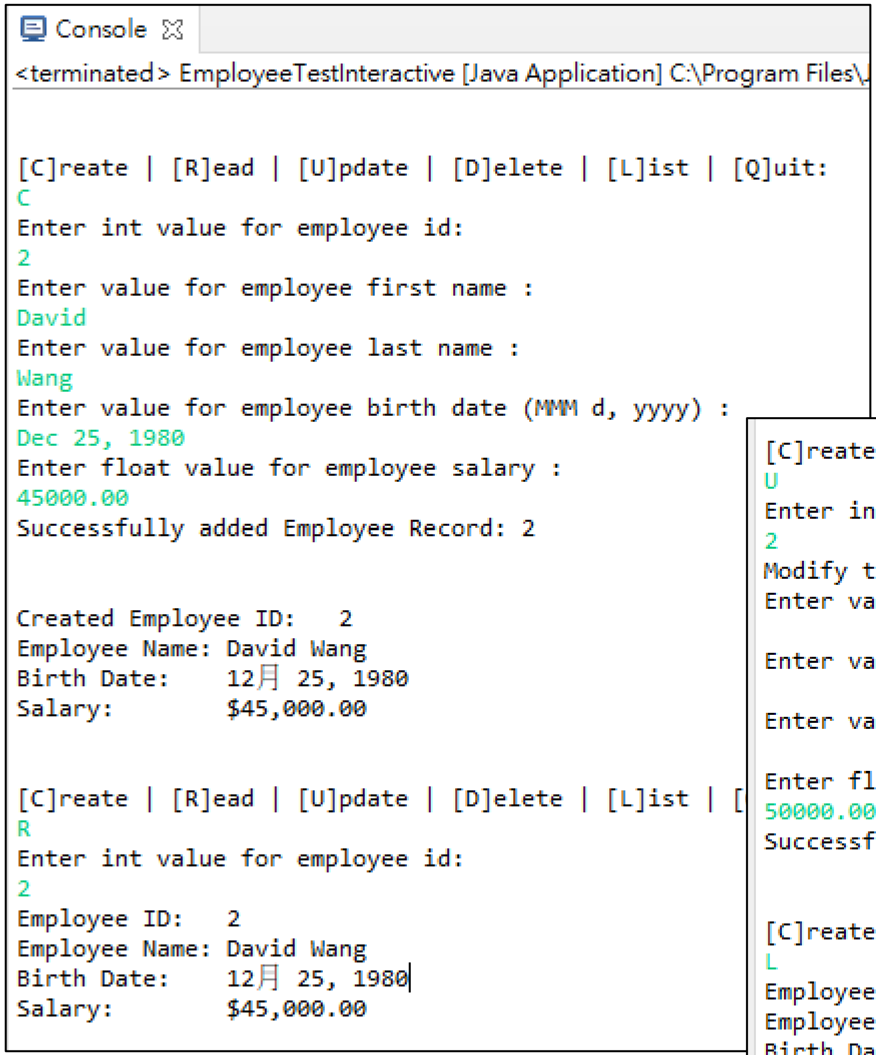


```
                // Display a list (Read the records) of Employees
                case 'L':
                    Employee[] allEmps = Employee.getAllEmployees();
                    for (Employee employee : allEmps) {
                        System.out.println(employee + "\n");
                    }
                    break;
            }

            return false;
        }
```

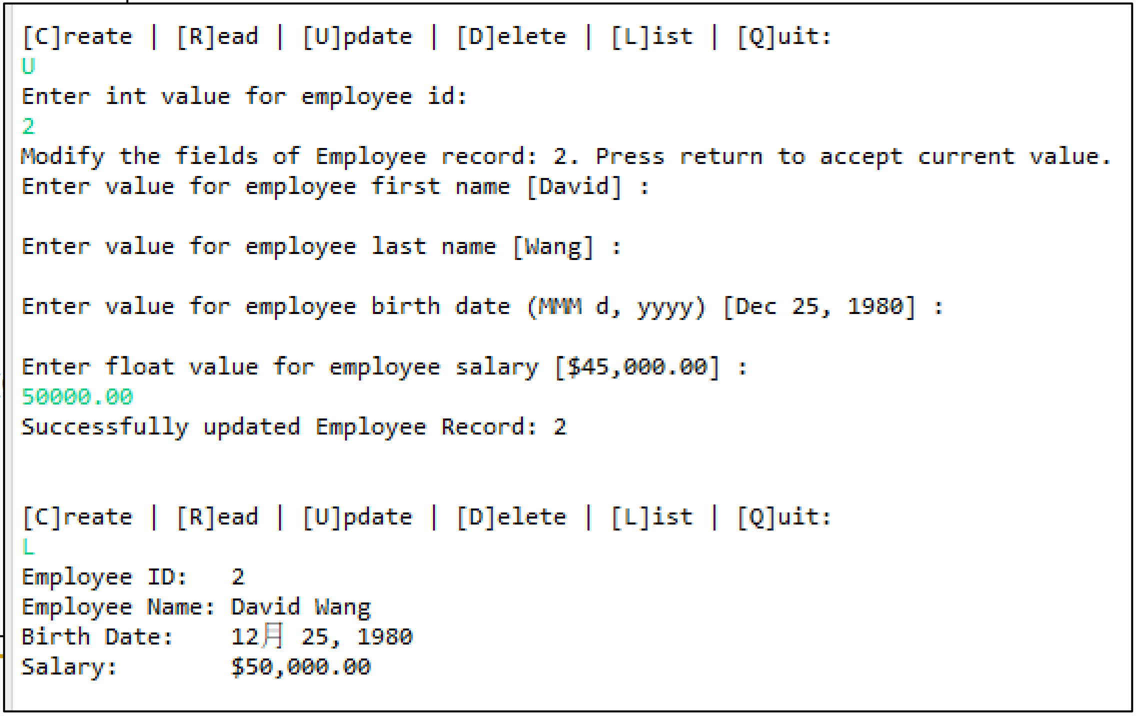


```
                // Display a list (Read the records) of Employees
                case 'L':
                    Employee[] allEmps = dao.getAllEmployees();
                    for (Employee employee : allEmps) {
                        System.out.println(employee + "\n");
                    }
                    break;
            }

            return false;
        }
```

測試、執行



```
Console

<terminated> EmployeeTestInteractive [Java Application] C:\Program Files\J


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
C
Enter int value for employee id:
2
Enter value for employee first name :
David
Enter value for employee last name :
Wang
Enter value for employee birth date (MMM d, yyyy) :
Dec 25, 1980
Enter float value for employee salary :
45000.00
Successfully added Employee Record: 2


Created Employee ID:   2
Employee Name: David Wang
Birth Date:    12月 25, 1980
Salary:        $45,000.00


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [
R
Enter int value for employee id:
2
Employee ID:   2
Employee Name: David Wang
Birth Date:    12月 25, 1980
Salary:        $45,000.00
```



```
[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
U
Enter int value for employee id:
2
Modify the fields of Employee record: 2. Press return to accept current value.
Enter value for employee first name [David] :

Enter value for employee last name [Wang] :

Enter value for employee birth date (MMM d, yyyy) [Dec 25, 1980] :

Enter float value for employee salary [$45,000.00] :
50000.00
Successfully updated Employee Record: 2


[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
L
Employee ID:   2
Employee Name: David Wang
Birth Date:    12月 25, 1980
Salary:        $50,000.00
```