한양대학교

# OBJECT-ORIENTED SYSTEMS DESIGN (Lab7)

*Heejin Park*

*Hanyang University*

# 7-1 (Display 6.18, 6.20)

Create classes *GradeBook* and *GradeBookDemo* defined as follows.

[*GradeBook*]

1. Create five instance variables as follows.

private int *numberOfStudents*

private int *numberOfQuizzes*

private int[][] *grade*

private double[] *studentAverage*

private double[] *quizAverage*

# 7-1 (Display 6.18)

2. Create a constructor **public GradeBook(int [][] a)** :

If **a.length** or **a[0].length** are 0,

print out “Empty grade records. Aborting.” and exit.

Store **a.length** into *numberOfStudents*.

Store **a[0].length** into *numberOfQuizzes*.

Invoke **fiillGrade(a)**.

Invoke **fiillStudentAverage()**.

Invoke **fiillQuizAverage()**.

# 7-1 (Display 6.18)

3. Create a constructor **public GradeBook(GradeBook book)** :

   Store **book.numberOfStudents** into *numberOfStudents*.

   Store **book.numberOfQuizzes** into *numberOfQuizzes*.

   Invoke **fiillGrade(book.grade)**.

   Invoke **fiillStudentAverage()**.

   Invoke **fiillQuizAverage()**.

# 7-1 (Display 6.18)

4. Create a constructor **public GradeBook()** :

Print the input and output below.

<input and output>

Enter number of students:
4
Enter number of quizzes:
3

Allocate to *grade* a new int [numberOfStudents][numberOfQuizzes].

# 7-1 (Display 6.18)

Print the input and output below and store input values into *grade* by using **for** statements.

<input and output>

```
Enter score for student number 1
on quiz number 1
10
Enter score for student number 1
on quiz number 2
10
Enter score for student number 1
on quiz number 3
10
Enter score for student number 2
on quiz number 1
2
⋮   ←—— This part is omitted in this page.
Enter score for student number 4
on quiz number 3
10
```

Invoke **fiillStudentAverage()**.

Invoke **fiillQuizAverage()**.

5. Create a method **private void fillGrade(int [][] a)**:

Allocate to *grade* a **new int [numberOfStudents][numberOfQuizzes]**.

Store the data of array *a* into array *grade*.

# 7-1 (Display 6.18)

6. Create a method **private void fillStudenetAverage()**:

   Allocate to *studentAverage* a **new double [numberOfStudents]**.

   Compute the averages of the students from the data in the array *grade* and store them into the array *studentAverage*.

7. Create a method **private void fillQuizAverage()**:

   Allocate to *quizAverage* a **new double [numberOfQuizzes]**.

   Compute the averages for the quizzes from the data in the array *grade* and store them into the array *quizAverage*.

# 7-1 (Display 6.18)

8. Create a method **public void display()**:

Print the output below by using **for** statements.

<output>

Student 1 Quizzes: 10 10 10 Ave = 10.0
Student 2 Quizzes: 2 0 1 Ave = 1.0
Student 3 Quizzes: 8 6 9 Ave = 7.66666666667
Student 4 Quizzes: 8 4 10 Ave = 7.33333333333
Quiz averages:
Quiz 1 Ave = 7.0 Quiz 2 Ave = 5.0 Quiz 3 Ave = 7.5

# 7-1

[*GradeBookDemo*]

Write a class *GradeBookDemo* that prints the input and output using the class *GradeBook*.

<input and output>

```
Enter number of students:
4
Enter number of quizzes:
3
Enter score for student number 1
on quiz number 1
10
Enter score for student number 1
on quiz number 2
10
 <The rest of the input dialogue is omitted to save space.>
Student 1 Quizzes: 10 10 10 Ave = 10.0
Student 2 Quizzes: 2 0 1 Ave = 1.0
Student 3 Quizzes: 8 6 9 Ave = 7.66666666667
Student 4 Quizzes: 8 4 10 Ave = 7.33333333333
Quiz averages:
Quiz 1 Ave = 7.0 Quiz 2 Ave = 5.0 Quiz 3 Ave = 7.5
```

# 7-2 (Display 7.2)

Create a class *Employee*.

Create a class *Date* by copying the class *Date* from chapter 4.

[*Employee*]

1. Create 2 private instance variables as follows.

   private String *name.*

   private Date *hireDate.*

2. Create a constructor **public Employee()** :

   Initialize *name* to “No name” and *hireDate* to (“January”, 1, 1000).

3. Create a constructor **public Employee(String theName, Date theDate)** :

If *theName* or *theDate* is null, print "Fatal Error creating employee." and exit. Otherwise, copy *theName* to *name* and deep copy *theDate* to *hireDate*.

4. Create a constructor **public Employee(Employee originalObject)** :

Copy *originalObject.name* to *name* and

deep copy *originalObject.hireDate* to *hireDate*.

5. Create a method **getName ()**: It returns **name**.
6. Create a method **getHireDate ()**: It returns **new Date(hireDate)**.

7. Create a method **setName (String newName)** :

   If *newName* is null, print “Fatal Error setting employee name.” and exit. Otherwise, copy *newName* to *name*.

8. Create a method **setHireDate (Date newDate)** :

   If *newDate* is null, print “Fatal Error setting employee hire date.” and exit. Otherwise, deep copy *newDate* to *hireDate*.

9. Create a method **toString ()**: It returns **name** + “ ” + **hireDate.toString**().

10. Create a method **equals (Employee otherEmployee)**:

    It returns **name.equals**(**otherEmployee.name**) && **hireDate.equals**(**otherEmployee.hireDate**).

# 7-3 (Display 7.3)

Create a class *HourlyEmployee* as a derived class of the class *Employee*.

[*HourlyEmployee*]

1. Create 2 private instance variables as follows.

   double *wageRate.*

   double *hours.*

2. Create a constructor **public HourlyEmployee ()** :

   Invoke **super()** and initialize *wageRate* and *hours* to 0.

3. Create a constructor **public HourlyEmployee(String theName, Date theDate, double theWageRate, double theHours)** :

Invoke **super(theName, theDate)**.

If *theWageRate* and *theHours* are greater than or equal to 0, store *theWageRate* to *wageRate* and *theHours* to *hours.* Otherwise, print "Fatal Error: creating an illegal hourly employee." and exit.

4. Create a constructor **public HourlyEmployee** (HourlyEmployee **originalObject)** :

Invoke **super(originalObject)**.

Copy *originalObject.wageRate* to *wageRage* and

*originalObject.hours* to *hours.*

5. Create a method **getRate ()**: It returns **wageRate**.
6. Create a method **getHours ()**: It returns **hours**.
7. Create a method **getPay ()**: It returns **wageRate*hours**.
8. Create a method **setHours (double hoursWorked)** :

   If *hoursWorked* is greater than or equal to 0, store *hoursWorked* to *hours*. Otherwise, print "Fatal Error: Negative hours worked." and exit.

9. Create a method **setRate (double newWageRate)** :

   If *newWageRate* is greater than or equal to 0, copy *newWageRate* to *wageRate*. Otherwise, print "Fatal Error: Negative wage rate." and exit.

10. Create a method **toString ()**:

    It returns **getName()** + “ ” + **getHireDate().toString()**

    + “₩n$” + *wageRate* + “ per hour for ” + *hours* + “ hours ”.

11. Create a method **equals (**HourlyEmployee **other)**

# 7-4 (Display 7.4)

Create a class *InheritanceDemo* that prints the output below using the class *HourlyEmployee* and *Date*.

The detailed description of the class is given on the next page.

<output>

joe's longer name is Joe Worker
Changing joe's name to Josephine.
joe's record is as follows:
Josephine January 1, 2015
$50.5 per hour for 160.0 hours

[*InheritanceDemo*]

1. Write a method **main**:

    1. Declare **HourlyEmployee type** *joe* by calling **HourlyEmployee** with 4 parameters “Joe Worker”, (“January”, 1, 2015) of **Date type,** 50.50, and 160.
    2. Print the output in the previous page using **getName()** and **setName()**.

# 7-5 (Display 7.5)

Create a class *SalariedEmployee* as a derived class of the class *Employee*.

[*SalariedEmployee*]

1. Create a private instance variable **double** *salary.*

2. Create a constructor **public SalariedEmployee ()** :

   Invoke **super()** and initialize *salary* to 0.

3. Create a constructor **public SalariedEmployee(**String **theName,** Date **theDate, double theSalary)** :

Invoke **super**(**theName, theDate**).

If *theSalary* is greater than or equal to 0, copy *theSalary* to *salary.*
Otherwise, print "Fatal Error: Negative salary." and exit.

4. Create a constructor **public SalariedEmployee** (SalariedEmployee **originalObject)** :

Invoke **super(originalObject**).

Copy *originalObject.salary* to *salary .*

5. Create a method **getSalary ()**: It returns **salary**.
6. Create a method **getPay ()**: It returns **salary/12**.

7. Create a method **setSalary** (double **newSalary**) :
If *newSalary* is greater than or equal to 0, copy *newSalary* to *salary*. Otherwise, print "Fatal Error: Negative salary." and exit.

8. Create a method **toString ()**:

It returns **getName()** + " " + **getHireDate().toString()**

+ "₩n$" + *salary* + " per year".

9. Create a method **equals (**SalariedEmployee **other)**:

# 7-6 (Display 7.6)

Create a class *IsADemo* that prints the output below using the class *SalariedEmployee*, *HourlyEmployee* and *Date*.

The detailed description of the class is given on the next page.

<output>

joe's longer name is Josephine
showEmployee(joe) invoked:
Josephine
January 1, 2015
showEmployee(sam) invoked:
Sam
February 1, 2016

[*IsADemo*]

1. Write a method **main**:

    1. Declare **SalariedEmployee type** *joe* by calling **SalariedEmployee** with 3 parameters “Josephine”, (“January”, 1, 2015) of **Date** type, and 100000.

    2. Declare **HourlyEmployee type** *sam* by calling **HourlyEmployee** with 4 parameters “Sam”, (“February”, 1, 2016) ) of **Date** type, 50.50, and 40.

    3. Print the output in the previous page using **getName()** and **showEmployee ().**

2. Create a method **showEmployee (**Employee **employeeObject)**:

    Print out **employeeObject.getName**() and **employeeObject.getHireDate**().