한양대학교

OBJECT-ORIENTED SYSTEMS DESIGN (Lab13-14)

Heejin Park

Hanyang University

13-1 (Display 13.1)

Create an interface *Ordered* defined as follows.

[*Ordered*]

1. Create two method headings.

public boolean **precedes**(Object **other**)

public boolean **follows**(Object **other**)

13-2 (Display 13.2)

Create a class *OrderedHourlyEmployee* that extends the class *HourlyEmployee* and implements the interface *Ordered* defined as follows.

Copy three classes *Employee, HourlyEmployee* and *Date* from chapter 7.

[*OrderedHourlyEmployee*]

1. Implement a method **precedes**(Object **other**) :

   If **other** is null, return false. If **other** is not an instance of *OrderedHourlyEmployee,* return false. Otherwise, create a variable OrderedHourlyEmployee *otherOrderedHourlyEmployee* and store in it the **other** converted to OrderedHourlyEmployee type, and then return whether or not **getPay()** is smaller than *otherOrderedHourlyEmployee*.**getPay().**

2. Implement a method **follows**(Object **other**):

   If **other** is null, return false. If **other** is not an instance of *OrderedHourlyEmployee,* return false. Otherwise, create a variable OrderedHourlyEmployee *otherOrderedHourlyEmployee* and store in it the **other** converted to OrderedHourlyEmployee type, and then return *otherOrderedHourlyEmployee*.**precedes(this).**

13-3 (Display 13.3)

Create an abstract class *MyAbstractClass* that implements the interface *Ordered* defined as follows.

[*MyAbstractClass*]

1. Create two instance variables as follows.

private int *number*

private char *grade*

13-3 (Display 13.3)

2. Create a method **public boolean precedes (Object other)**:

   If **other** is null, return false. If **other** is not an instance of *HourlyEmployee,* return false. Otherwise, create a variable MyAbstractClass *otherOfMyAbstractClass* and store in it the **other** converted to MyAbstractClass type, and then return whether or not *this.number* is smaller than *otherOfMyAbstractClass.number*.

3. Create a method heading **public abstract boolean follows**(Object **other).**

13-4 (Display 13.4)

Create an interface *ShowablyOrdered* that extends the interface *Ordered* defined as follows.

[*ShowablyOrdered*]

1. Create a method heading **public void showOneWhoPrecedes().**

13-5 (Display 13.5)

Create a class *GeneralizedSelectionSort* defined as follows.

[*GeneralizedSelectionSort*]

1. Create a method **public static void sort(Comparable [] a, int numberUsed):**

   Create two integer variables *index* and *indexOfNextSmallest.*

   Sort an array **a** in increasing order by using the methods **indexOfSmallest** and **interchange**.

2. Create a method **private static int indexOfSmallest(int startIndex, Comparable [] a, int numberUsed):**

   Create a variable Comparable *min* and store **a[startIndex]** in it.

   Create a variable *indexOfMin* and store **startIndex** in it.

   Create a variable *index*.

   find *min* among **a[startIndex]...a[numberUsed]** and set *indexOfMin* the index of *min.*

   Return *indexOfMin.*

3. Create a method **private static void interchange**(int **i**, int **j**, Comparable[] **a**):

   Create a variable **Comparable** *temp*.

   Swap **a[i]** and **a[j]**.

13-6 (Display 13.6)

Create a class *ComparableDemo* that prints the output below using the class *GeneralizedSelectionSort*.

<output>
Before sorting
10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0,
After sorting:
1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0,
Before sorting;
dog, cat, cornish game hen,
After sorting:
cat, cornish game hen, dog,

13-7 (Display 13.9)

Create a class *BankAccount* that prints the output below.

[*BankAccount*]

1. Create inner class **private** *Money* as follows.

    1. Create two instance variables as follows.

private long *dollars*

private int *cents*

2. Create a constructor **public Money(String stringAmount)** :

Invoke **abortOnNull(stringAmount)**

Create a variable **int** *length* and store **stringAmount.length()** in it.

Store **Long.parseLong(stringAmount.substring(0, length - 3))** in *dollars*.

Store **Integer.parseInt(stringAmount.substring(length - 2, length))** in *cents*.

3. Create a method **public String getAmount()** :

If *cents* is greater than 9, return (*dollars* + "." + *cents*).

Otherwise, return (*dollars* + ".0" + *cents*).

4. Create a method **public void addIn(Money secondAmount)** :

Invoke **abortOnNull(secondAmount)**

Create a variable **int** *newCents* and store (*cents*+**secondAmount**.*cents*)%100 in it.

Create a variable **int** *carry* and store (*cents*+**secondAmount**.*cents*)/100 in it.

Store *newCents* in *cents*.

Store (*dollars* + **secondAmount**.*dollars* +*carry)* in *dollars*.

5. Create a method **public void abortOnNull(Object o)** :

If **o** is null, print “Unexpected null argument.” and exit.

2. Create an instance variable **private Money balance;**

3. Create a constructor **public BankAccount()**:

Make ***balance*** points to a new object of the class **Money** that has an argument **"0.00"**.

4. Create a method **public String getBalance()**:

It returns **balance**.*getAmount()*

5. Create a method **public void makeDeposit** (String **depositAmount**):

   Invoke **balance**.*addIn* with an argument **new Money(depositAmount)**.

6. Create a method **public void closeAccount** ():

   Store 0 in **balance**.*dollars*.

   Store 0 in **balance**.*cents*.

13-8 (Display 13.10)

Create a class *BankAccountDemo* that prints the output below using the class *BankAccount*.

<output>

Creating a new account.
Account balance now = $0.00
Depositing $100.00
Account balance now = $100.00
Depositing $99.99
Account balance now = $199.99
Depositing $0.01
Account balance now = $200.00
Closing account.
Account balance now = $0.00

14-1 (Display 14.3)

Create a class *GolfScores* that prints the output below by using the class *ArrayList*.

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

<input and output>

```
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:
Enter a list of nonnegative numbers.
Mark the end of the list with a negative number.
69 74 68 -1
Average of the 3 scores = 70.3333
The scores are:
69.0 differs from average by -1.33333
74.0 differs from average by 3.66667
68.0 differs from average by -2.33333
```

[*GolfScores*]

1. Import 2 classes as follows.

    java.util.ArrayList

    java.util.Scanner

2. Write a method **main**:

Create an **Double**-typed **ArrayList** *score* and allocate to *score* a **new ArrayList<Double>()**.

Print the output below.

<output>
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:

Invoke **fillArrayList(score)**.

Invoke **showDifference(score)**.

3. Create a method **public static void fillArrayList(ArrayList<Double> a)**:
Print the output below.

<output>
Enter a list of nonnegative numbers.
Mark the end of the list with a negative number.

Create a variable **double** *next*.

Store the double-typed input into *next*.

while *next* is greater than or equal to 0,

store *next* into the array **a** and store the next input in *next*.

4. Create a method **public static double computeAverage(ArrayList<Double> a):**

Create a variable **double** *total* initialized to 0.

Compute the total of the numbers in **a** by using a **for-each loop**.

Create a variable **int** *numberOfScores* initialized to **a.size()**.

If *numberOfScores* is greater than 0, return the average of **a.**

Otherwise, print "ERROR: Trying to average 0 numbers." and "computeAverage returns 0." and return 0.

5. Create a method **public static void showDifference(**ArrayList<Double>**a):**

Create a variable **double** *average* initialized to **computeAverage(a).**

Print the below by using a **for-each loop**.

<output>
Average of the 3 scores = 70.3333
The scores are:
69.0 differs from average by -1.33333
74.0 differs from average by 3.66667
68.0 differs from average by -2.33333