Practices for Lesson 6

Practices Overview

In these practices, you create `if` and `if` / `else` constructs and also create a `switch` construct.

Practice 6-1: Writing a Class that Uses the `if/else` Statement

Overview

In this practice, you create classes that use `if` and `if/else` constructs. There are two sections in this practice. In the first section, you create the DateTwo class that uses `if / else` statements to display the day of the week based on the value of a variable. In the second section, you create the Clock class that uses `if/else` statements to display the part of the day, depending on the time of day.

Assumptions

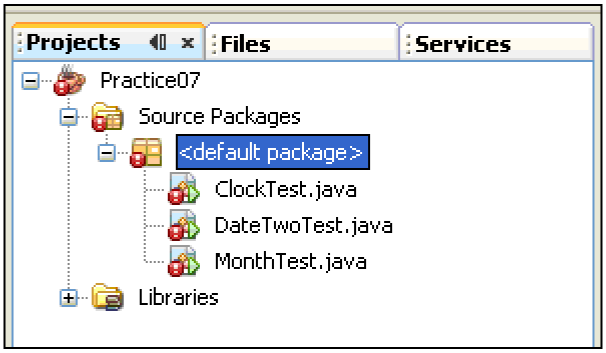
The following files appear in the practice folder for this lesson, `Lesson06`:

- `ClockTest.java`
- `DateTwoTest.java`

Writing a Class that Uses an `if/else` Statement

In this task, you create a DateTwo class that evaluates a numeric field in order to determine the day of the week that corresponds to that number. You use an `if/else` construct to do this.

1. Create a new project from existing sources called Practice07. Set the **Source Package Folder** to point to `Lesson06`. Remember to also change the Source/Binary Format property. If you need further details, refer to Practice 1-2, Steps 3 and 4.



2. Create a new Java class called “DateTwo”. Declare and initialize a member field for this class called `dayNumber`. The value assigned should be a number between 1 and 7 (inclusive) where the number 1 represents Monday (beginning of the week) and 7 represents Sunday (end of the week).

   Hint: Use the `int` data type.

3. Create a `displayDay` method in the DateTwo class. High level instructions for this task are provided in the table below. More detailed instructions can be found following the table.

Step	Window/Page Description	Choices or Values
a.	Use an `if/else` construct to inspect the value of `dayNumber`.	In each `if` block, display the corresponding day of the week
b.	Display an error message if an invalid number is found	This should be the last condition you check

   a. The following pseudo code will help you write the body of the `displayDay` method. Each `if` condition should check the value of `dayNumber`. Hint: Use the == sign. Within the `if` blocks, print out the day of the week (“Monday”, “Tuesday”, etc.)

```
if (condition1) {
     // print corresponding day
}else if (condition2) {
     // print corresponding day
}else if (condition3)
     …
}else {
     // if none of the conditions is true
}
```

   b. If `dayNumber` does not equal a number between 1 and 7 (inclusive), print out an error message. This will be in the final `else` block.

4. Save, compile, and execute your class by running the DateTwoTest class. Check the output in the Output window.
5. Repeat step 4 several times by assigning different values to the `DateTwo` member field.

Writing Another Class That Uses `if/else` Statements

In this task, you write a class called "Clock" that uses `if/else` statements to display the part of day depending upon the time of day. Use the following table as a guideline.

Time of Day	Part of Day
8:01 to 12:00	Morning
12:01 to 17:00	Afternoon
17:01 to 24:00	Evening
0:01 to 8:00	Early Morning

6. Create a new Java class called "Clock" that contains an `int` field called `currentTime`. Initialize this field to hold the hour of the day. (Example: 400 = 04:00, 1505 = 15:05).
7. In the Clock class, create a `displayPartOfDay` method. Display the part of the day associated with the value of the `currentTime` field. For example, if the `currentTime` field equals 2100, you would display "Evening". You need not check for values outside the range of 1 to 2400.

   Hint: Use a similar `if/else` construct to what you used in the previous task.

Solution:

```
public void displayPartOfDay() {
    if(currentTime >= 801 && currentTime <= 1200){
          System.out.println(“Morning”);
    }else if(currentTime >= 1201 && currentTime <= 1700){
          System.out.println(“Afternoon”);
    }else if(currentTime >= 1701 && currentTime <= 2400){
          System.out.println(“Evening”);
    }else {
          System.out.println(“Early Morning”);
    }
}
```

8. Save, compile, and execute your program by running the ClockTest class.
9. Repeat Step 8 several times by assigning different values to the `currentTime` member variable.

   Note: A leading zero indicates an octal value. Therefore, the program does not compile if you set currentTime to 0800. You need to specify currentTime as 800 for 8:00 AM to successfully compile the program.

Practice 6-2: Writing a Class that Uses the `Switch` Statement

Overview

In this practice, you create a class called “Month” that uses switch statements to display the name of the month based upon the numeric value of a field.

Assumptions

The MonthTest.java file appears in the practice folder for this lesson, `Lesson06`

Tasks

1. Create a new Java class called “Month”.
2. Declare an `int` field in the Month class called `monthNumber`. Assign a value to the field that is between 1 and 12 (inclusive), where 1 represents the month of January and 12 represents the month of December.
3. Create a new method in the Month class called `displayMonth`. This method uses a `switch` construct to inspect the value of the `monthNumber` field and display the corresponding name of the month. The `displayMonth` method should also display an error message if an invalid number is used.

   Hint: The syntax for a `switch` statement is:

   ```
   switch (<variable>){
       case <value1>:
           // do something
           break;
       case <value2>:
           // do something
           break;
       … // more cases
       default:
           // possibly error checking
           break;
   } // end of switch
   ```
4. Save, compile, and execute your program by running the MonthTest class.
5. Repeat step 4 several times assigning different values to the `monthName` field.
6. Close the Practice07 project in NetBeans.