Hyperion Development

Task: Control Structures - Switch

www.hyperiondev.com

Introduction

Welcome to the Control Structures - Switch Task
Overview:

You should currently be very comfortable with the understanding and implementation of different conditional statements - if not, be sure to go through the content of your previous tasks! You'll now be exposed to the switch, which is essentially a different way of expressing a lengthy *if, elseif, else* statement. It's much more effective as your code use is shortened as well having code that is much more understandable if you're going to have a long list of conditions.

What is a switch statement?
The switch-statement is a special form of multiway selection that transfers control to one of several statements depending on the value of the expression. The Java syntax for the statement begins with the reserved keyword 'switch' and a selector expression enclosed in parentheses. The body of the statement is a block which contains one or more 'switch labels' consisting of the reserved word case, a constant, the delimiter colon, and a list of statements that are the action for the specific case.

Syntax:

```
switch( selector expression )
{
        case constant_1: Statement_for_constant_1;
                break;

        ........

        case constant_n: Statement_for_constant_n;
                break;
        default: Statement_if_no_case_matches_selector;
                break;
}
```

The selector expression must have a discrete (integer or character) value. The switch-statement executes by taking the selector value and comparing it with each case constant. If a match occurs then the corresponding statement sequence is executed.

If no match occurs then control passes to the 'default' label if it exists, or to the first statement following the switch block. When a statement sequence concludes there is no automatic transfer to the statement following the switch block. Instead, control continues to the next statement sequence. This situation is not normally desired and so Java provides a break statement which forces a branch out of the switch-statement.

Example: Describe types of coins for different values of the integer selector *coinValue*. This example allows two or more case options to be included with a single statement.

```
switch( coinValue )
{
        case 1:
        case 2:
        case 5:
        case 10: System.out.println( coinValue + " cents is a standard coin");
                break;
        case 50: System.out.println( coinValue + " cents is a special coin" );
                break;
        default: System.out.println( "No coin for " + coinValue + " cents" );
                break;
}
```

Compile and Run the program example.java in the Task 5 folder and make sure you observe the following output.

Run:

For coinvalue = 10, the output is "10 cents is a standard coin"

For coinvalue = 50, the output is "50 cents is a special coin"

For coinvalue = 70, the output is "No coin for 70 cents"

Missing break statement

If a break statement is not placed at the end of a case-option the runtime system will execute instructions in the next case-option as well. For instance, assume the example includes no break statements and *coinValue* is 10. A run would produce output that includes the case '50' option and the 'default' option.

Output:

10 cents is a standard coin
10 cents is a special coin
No coin for 10 cents

Feel free to add and/or remove the break statements in the different parts of exampleBreak. java.

Instructions

First read example.hava, open it using jGRASP (Right click the file and select 'Open with jGRASP").

- In this folder there is a file called example.java
- Open this folder using the JGRASP program. You can either do this by right clicking on the example.java file, going to the 'Open with' menu, and selecting JGrasp.exe. Alternatively, you can run the JGRASP program on your computer, go to the top left corner of the program, select File->Open and navigate to example.java on your hard drive and double click on it to open it.
- Once example.java is open in JGRASP please read all of its content very carefully. This will help you understand the basic structure of a Java program.
- There is a compulsory task at the end of the example.java document. The instructions of what to do to complete the task can be found here. You must complete this exercise to continue onto the next task of this course.

Compulsory Task 1

Follow these steps:

- Create a new file called switch.java.
- Using a switch statement: write a program that asks the user to enter a numeric digit from zero up to (and not including) 100.
- Once the user has entered the number the program should convert the number into words.
- For example: if the user enters '52' the program displays fifty two. Your program should use a switch statement to make the necessary numeric to word conversions for any integer number entered by the user.
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Enhance the compulsory task program so that the user may enter an amount of money which the program will read back to the user stating the correct magnitudes of thousands, hundreds, and cents.
- For example: Input: R123.99
  Output: One hundred and twenty three rands and ninety nine cents.
- Assume the input amount will always be in rands and that if the input amount is x, then
  10000 > x >= 0.
- Save this file as switchEnhanced.java.
- Compile, save and run your file.

Things to look out for

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **jGRASP or Java** may not be installed correctly.

2. If you are not using Windows, please ask your tutor for alternative instructions.

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 20/01/2016.
Authors: Riaz Moola and Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.