Task: Control Structures - Do While Loop

www.hyperiondev.com

Introduction

Welcome to the Control Structures - Do While Loop Task
Overview:

You should currently be very comfortable with the understanding and implementation of different conditional statements - if not, be sure to go through your previous tasks content! You'll now be sequentially exposed to loop statements to understand how they can be utilised in reducing lengthy code, preventing coding errors, as well as paving the way towards code reusability. The next statement you'll be exposed to is the do while loop, which simply adds additional functionality to the while loop.

-The Hyperion Team

What is a do...while?
The '*do...while*'-loop is similar to a while-loop except that it places the loop test at the end of the loop body. A 'do...while'-loop executes at least one iteration and then continues execution so long as the test condition remains true.

Syntax:

```
do
{
        body
}while( logical condition );
```

Example:
When a user is required to make a choice from a program menu you want the menu to be displayed at least once even though the user might choose to terminate the program after seeing the menu.

```
do
{
        System.out.println( "Enter today's number from themenu:" );
        int choice = Integer.parseInt( JOptionPane.showInputDialog("1. Sunday\n"+
                                                                   "2. Monday\n"+
                                                                   "3. Tuesday\n"+
                                                                   "4. Wednesday\n"+
                                                                   "5. Thursday\n"+
                                                                   "6. Friday\n"+
                                                                   "7. Saturday"+
                                                                   "0. Quit program" ) );
        switch( choice ):
        {
                case 1: System.out.println( "Sunday! Last day to do the homework."); break;
                case 2: System.out.println( "Monday!  Be happy, 5 more days in the week."); break;
                case 3: System.out.println( "Tuesday!  Wednesday will be better."); break;
                case 4: System.out.println( "Wednesday!  Can you feel it in the air?"); break;
                case 5: System.out.println( "Thursday!  Get set!"); break;
                case 6: System.out.println( "Friday!  Time to party!"); break;
                case 7: System.out.println( "Saturday!  Put your legs up and relax."); break;
                default: System.out.println( "Please enter a number between 1 and 7, or 0 to quit."); break;
        }
}while(choice != 0);
```

The code given in the preceding example has been made available in **example.java**. Compile and execute this file with different input numbers for your amusement.

Switch it up
For the purposes of this task, we'll cover a very brief introduction of what you need to complete the task. The *switch* conditional statement is basically a shorthand version of writing many *if...else* statements. The syntax for *switch* statements is as follows:

```
switch (<variable>) {
  case <result>: <statements>; break;
  case <result>: <statements>; break;
  default: <statements>; break;
}
```

This means that if the variable included equals one of the case results, the statements following that case, until the word break, will run. The default case executes if none of the others are true.

An example of the switch statement can be seen here:

A switch.

```
1 int n = 2, x = 0;
2 switch (n) {
3   case 1: x = 2;
4     break;
5   case 2: x = 4;
6     break;
7   case 3: x = 6;
8     break;
9   case 4: x = 8;
10    break;
11 }
12 System.out.println(x);
```

In this example, since the integer variable n is equal to 2, case 2 will execute, make x equal to 4. Thus, 4 is returned by the method.

Instructions

First read example.java, 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 do_whilePassword.java.
- Imagine that the password to sign in to some computer is "John". Write a program that prompts the user to enter a password to login to this imaginary computer.
- If the user enters the correct password then notify the user and terminate the program.
- Should the user get the password incorrect three times in a row, then inform the user of the current password and allow the user to set a new password.
- Check the new password so that it satisfies these modified java identifier rules:
  - The password may have any number of characters between 1 and 20, inclusive.
  - The password may start with an underscore "_" or any letter of the alphabet
  - The password may be any combination of upper and lower case letters.
  - The password consists only of numbers, English alphabet letters, and the underscore character.
- Should a proposed password not match these rules then inform the user that they have entered an invalid password and prompt the user to try again until an acceptable password is found.
- Here is an example run on the assumption that the current password is "John":
  - prompt: Enter password:
  - input: peter
  - prompt: Incorrect password. Please enter password:
  - input: luke
  - prompt: Incorrect password. Please enter password:
  - input: tabo
  - prompt: Password incorrect on three attempts.
  - prompt: The password was John. Please set a new user password:
  - input: rivoningo23
  - prompt:password changed.
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Create a new file called do_whileMenue.java.
- Using a do...while loop, implement a program that presents a menu to the user. The options of the menu allow the user to use the programs you wrote in the previous activities.
- The menu of the program is as follows:
  - 1. task3.java
  - 2. errors.java
  - 3. while.java
  - 4. do_whilePassword.java
  - 5. quit
- The program executes the corresponding programs implemented in the previous tasks. Be sure to direct the user on how to use the menu.

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.]