Practical Exercise 6 – If and Switch statements

An “if” statement is a conditional statement that performs different actions depending on whether a specified boolean expression evaluates to true or false. It has a number of general forms…

if (hungry) eatSomething(); //if with single statement

if (darkOutside) // if-else with single statements

turnLightsOn();

else

turnLightsOff();

if (cookingCake) //if with multiple statements

{

mixIngredients();

bakeCake();

eatCake();

}

if (guess == correctNumber) // nested if-else

{

g.drawString(“Yay!”,20,20);

score = score + 1;

resetGame();

}

else if (guess < correctNumber)

{

g.drawString(“Too low. Guess again!”,20,20);

}

else

{

g.drawString(“Too high. Guess again!”,20,20);

}

Notes

* The value in *regular brackets* ( ) following the “if” is a boolean expression i.e. a statement that that is either true or false. This could simply be, for example, a boolean variable, but will more commonly be an expression that uses a comparison of some kind.
* When an “if” structure contains more than one statement in a single branch, these multiple statements must be enclosed in *curly brackets* { }. Otherwise only the first statement is controlled by the “if” structure.
* While the indentation of the curly brackets doesn’t matter so far as program execution is concerned, your code will be easier to read if you keep them lined up. This will lead to less mistakes for you, and stops Rob from getting cranky when he’s trying to make sense of your code.

A “switch” statement is often a much neater way of dealing with multiple possible execution paths than a nested if structure. For example…

int month = 2;

String monthString;

switch (month)

{

case 1: monthString = “January”;

break;

case 2: monthString = “February”;

break;

case 3: monthString = “March”;

break;

// …and so on until…

case 12: monthString = “December”;

break;

default: monthString = “Invalid month”;

break;

}

g.drawString(monthString,20,20);

Note the use of the break statement. Each break terminates the enclosing switch statement. If the break statement is left out, then program execution would continue through to the next case, and then next, until a break statement is reached.

THINK: When might it be useful to deliberately leave out the break statements?

## Comparison operators

== x == y x is equal to y

!= x != y x is not equal to y

> x > y x is greater than y

< x < y x is less than y

>= x >= y x is greater than or equal to y

<= x <= y x is less than or equal to y

## Logical operators

|| a || b a or b

&& a && b a and b

! !a not a

# Tasks

For each of these tasks you will be taking some user input, doing a calculation of some sort, and then displaying a result.

For each of these, start with a minimal program and get that running first. You could start with something from an earlier prac, but here's some sample code in case you need it.

Remember to change the name of the JPanel for each new program you create in the same project folder (e.g. change *Task0* to *Task1*, *Task2*, etc.), and please change the variable names to something that makes sense in your program (e.g. change *number* to *hoursWorked*, etc.)

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

class Task0 extends JPanel implements ActionListener

{

Label prompt;

TextField inputField;

Button enterButton;

int number = 42;

public void init()

{

prompt = new Label("Please enter a value: ");

inputField = new TextField(""+number,10);

enterButton = new Button("Enter");

add (prompt);

add (inputField);

add (enterButton);

inputField.setText("" + number);

enterButton.addActionListener(this);

}

public void paint(Graphics g)

{

paintComponent(g);

g.drawString("The number is: " + number, 20, 100);

}

public void actionPerformed(ActionEvent e)

{

number = Integer.parseInt(inputField.getText());

System.out.println("You entered: " + number);

repaint();

}

}

public class SimpleComponentDemo

{

public static void main(String[] a)

{

Task0 myCanvas = new Task0();

myCanvas.init();

JFrame window = new JFrame();

window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

window.setBounds(30, 30, 300, 300);

window.getContentPane().add(myCanvas);

window.setVisible(true);

}

}

## Task 1: Salaries

Create a program called *Salaries.java* to calculate the wages owing to a worker given the following rules.

* 1. If the total hours worked is less than 35, the hourly rate is $15 per hour.
  2. If the total hours worked is more than 35, the hours in excess of 35 earn a rate of $25 per hour.

Your program should have a text field for the user to enter the total number of hours worked.

## Task 2: Day of the Week - IF

1. Create a program called *DayOfWeek.java* to output the day of the week corresponding to the day number.   
     
   Note: For this exercise, Monday is the first day of the week and Sunday is the 7th day. That is, with dayNumber set to 3, the program should display “Wednesday”.

int dayNumber;

public void init()

{

dayNumber = 3;

}

public void paint (Graphics g)

{

<place if statement here>

}

An error message should be produced if the number input is outside the range 1 to 7.

1. Alter the program so it uses a nested if (if .. else) structure.
2. Add a text box so the user can enter a day number, and the correct day of the week is displayed automatically.

## Task 3: Day of the Week - SWITCH

1. Alter the program so it uses a switch statement instead of the if .. else.
2. Modify your switch statement to use the modulus operator (%) so that any number (including those outside of the range 1 to 7) will produce a meaningful result e.g. day 8 is Monday, day 0 is Sunday.

## Task 4: Time for Class (stretch goal – bonus marks)

1. Create a new program called *Time.java* that displays the current time, e.g. "The time is 9:12:05". The following code might be useful…

import java.time.LocalTime

...

LocalTime timeOfDay = LocalTime.ofSecondOfDay(seconds);

String time = timeOfDay.toString();

1. Calculate and display the number of seconds remaining in the day.
2. Calculate display the percentage of the day that has passed. You might have trouble when computing percentage with integers, so consider using a floating-point data type.
3. Write code to display an appropriate message depending on whether the time is before, during, or after class. For example,…
   1. If it's before class (i.e. 8:45am), display "I can't wait for Computer Science"
   2. If it's during class (i.e. between 8:45am and 12:10pm), display "It's time for Computer Science – yay!"
   3. If it's after class (i.e. after 12:10pm), display "Computer Science is over for the day… so sad!"

Note: This task was based on an exercise from an excellent (AND FREE) PDF book called "Think Java: How to Think Like a Computer Scientist" by Allen B. Downey and Chris Mayfield. If you haven't already checked it out, you should. Go on. Find it and download it now. Yes, now. Do it. Now.