

Chapter 3

Flow of Control

Prof. Yongsu Park

Division of Computer Science and Engineering
Hanyang University

Flow of Control

- As in most programming languages, *flow of control* in Java refers to its *branching* and *looping* mechanisms
- Java has several branching mechanisms:
 - `if`, `else if`, `else`
 - `switch`
- Java has three types of loop statements:
 - `while`,
 - `do-while`
 - `for`
- Most branching and looping statements are controlled by Boolean expressions
 - A Boolean expression evaluates to either `true` or `false`
 - The primitive type `boolean` may only take the values `true` or `false`

Display 3.1 Tax Program

– an example of if, else if, else

```
import java.util.Scanner;

public class IncomeTax
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        double netIncome, tax, fivePercentTax,
        tenPercentTax;

        System.out.println("Enter net income.\n"
            + "Do not include a dollar sign or any
        commas.");
        netIncome = keyboard.nextDouble( );

        if (netIncome <= 15000)
            tax = 0;
        else if ((netIncome > 15000) && (netIncome <=
        30000))
            //tax = 5% of amount over $15,000
            tax = (0.05*(netIncome - 15000));
        else //netIncome > $30,000
        {
            //fivePercentTax = 5% of income from $15,000
            to $30,000.
            fivePercentTax = 0.05*15000;
            //tenPercentTax = 10% of income over
            $30,000.
            tenPercentTax = 0.10*(netIncome - 30000);
            tax = (fivePercentTax + tenPercentTax);
        }

        System.out.printf("Tax due = $%.2f", tax);
    }
}
```

Display 3.1 Tax Program

Enter net income.

Do not include a dollar sign or any commas.

40000

Tax due = \$1750.00

Display 3.2 A switch Statement

```
import java.util.Scanner;

public class SwitchDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard =
            new Scanner(System.in);

        System.out.println("Enter number of
                           ice cream flavors:");
        int numberOfFlavors = keyboard.nextInt( );

        switch (numberOfFlavors)
        {
            case 32:
                System.out.println("Nice selection.");
                break;
            case 1:
                System.out.println("I bet it's vanilla.");
                break;
            case 2:
            case 3:
            case 4:
                System.out.println(numberOfFlavors + " flavors");

                System.out.println("is acceptable.");
                break;
            default:
                System.out.println("I didn't plan for");
                System.out.println(numberOfFlavors + " flavors.");
                break;
        }
    }
}
```

Display 3.2 A switch Statement

Enter number of ice cream flavors:

1

I bet it's vanilla.

Enter number of ice cream flavors:

32

Nice selection.

Enter number of ice cream flavors:

3

3 flavors

is acceptable.

Enter number of ice cream flavors:

9

I didn't plan for 9 flavors.

Pitfall: Using `==` with Strings

- The equality comparison operator (`==`) can correctly test two values of a *primitive* type
- However, when applied to two *objects* such as objects of the `String` class, `==` tests to see if they are stored in the same memory location, not whether or not they have the same value
- In order to test two strings to see if they have equal values, use the method `equals`, or `equalsIgnoreCase`
`string1.equals(string2)`
`string1.equalsIgnoreCase(string2)`

Display 3.4 Comparing Strings

```
public class StringComparisonDemo
{
    public static void main(String[] args)
    {
        String s1 = "Java isn't just for breakfast.";
        String s2 = "JAVA isn't just for breakfast.";

        if (s1.equals(s2))
            System.out.println("The two lines are equal.");
        else
            System.out.println("The two lines are not equal.");

        if (s2.equals(s1))
            System.out.println("The two lines are equal.");
        else
            System.out.println("The two lines are not equal.");

        if (s1.equalsIgnoreCase(s2))
            System.out.println("But the lines are equal, ignoring case.");
        else
            System.out.println("Lines are not equal, even ignoring case.");
```

```
String s3 = "A cup of java is a joy forever.";
if (s3.compareToIgnoreCase(s1) < 0)
{
    System.out.println "\"" + s3 + "\"";
    System.out.println("precedes");
    System.out.println "\"" + s1 + "\"";
    System.out.println("in alphabetic
ordering");
}
else
    System.out.println("s3 does not
precede s1.");
}
```


Display 3.4 Comparing Strings

The two lines are not equal.
The two lines are not equal.
But the lines are equal, ignoring case.
“A cup of java is a joy forever.”
precedes
“Java isn’t just for breakfast.”
in alphabetical ordering

Loops

- *Loops* in Java are similar to those in other high-level languages
- Java has three types of loop statements: the **while**, the **do-while**, and the **for** statements
 - The code that is repeated in a loop is called the *body* of the loop
 - Each repetition of the loop body is called an *iteration* of the loop

Display 3.7 Demonstration of while Loops and do-while Loops

```
public class WhileDemo
{
    public static void main(String[] args)
    {
        int countDown;

        System.out.println("First while loop:");
        countDown = 3;
        while (countDown > 0)
        {
            System.out.println("Hello");
            countDown = countDown - 1;
        }

        System.out.println("Second while loop:");
        countDown = 0;
        while (countDown > 0)
        {
            System.out.println("Hello");
            countDown = countDown - 1;
        }
    }
}
```

```
        System.out.println("First do-while loop:");
        countDown = 3;
        do
        {
            System.out.println("Hello");
            countDown = countDown - 1;
        } while (countDown > 0);

        System.out.println("Second do-while loop:");
        countDown = 0;
        do
        {
            System.out.println("Hello");
            countDown = countDown - 1;
        } while (countDown > 0);
    }
}
```

Display 3.7 Demonstration of while Loops and do-while Loops

First while loop:

Hello

Hello

Hello

Second while loop:

First do-while loop:

Hello

Hello

Hello

Second do-while loop:

Hello