Java Chapter 3: Decisions

Contents

Time required: 90 minutes

DRY

Don't **R**epeat **Y**ourself

Do: Think Java, 2nd Ed. (Interactive Edition)

- Chapter 3 Input and Output
- Chapter 5 Conditionals and Logic

Do: Java Tutorials

- LearnJavaOnline.org Conditionals
- Java Type Casting
- Java Operators
- Java Strings
- Concatenation
- Numbers and Strings
- Special Characters
- Java Math
- Java Booleans
- Java If…Else
- Short Hand If…Else
- Java Switch

Visualize and Debug Programs

The website www.pythontutor.com helps you create visualizations for the code in all the listings and step through those programs one instruction at a time. As you do that, pay attention to the movements of the red and green arrows on the left, the diagram on the right, and the printed material at the bottom. This will help you to better understand the behavior of the programs we are working on.

This is a great way to debug your code. You can see the variables change as you step through the program.

https://pythontutor.com/java.html#mode=edit

If Selection Statement

In Java, the "**if**" statement is used for decision-making in programming. It evaluates a Boolean expression and executes a block of code if the expression is true. If the expression is false, the block of code is skipped.

```
public class If {
    public static void main(String[] args) {
        int age = 19;
        if (age >= 16) {
            System.out.println("You can drive!");
        }
    }// end main
}
```

If Else

In Java, the "**if-else**" statement extends the basic "**if**" statement by providing an alternative block of code to execute when the "**if**" condition evaluates to false. This allows for branching logic where one of two code paths is executed based on a Boolean condition.

```
public class IfElse {
    public static void main(String[] args) {
        int age = 19;
        if (age >= 16) {
            System.out.println("You can drive!");
        } else {
            System.out.println("You cannot drive yet!");
        } // end if-else
    }// end main
}
```

If Else If

In Java, the "**if else if else**" statement allows for multiple conditional branches. It evaluates each condition sequentially until one is found to be true, executing the corresponding block of code. This structure is useful for implementing complex decision-making logic with several possible outcomes.

```
import java.util.Scanner;
public class IfElseIf {
    public static void main(String[] args) {
        int age;
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Welcome to the Sunset Bar and Grille.");
        System.out.print("Please enter your age: ");
        age = keyboard.nextInt();
        if (age >= 21) {
            System.out.println("Here, have a beer.");
        } else if (age >= 16) {
            System.out.println("Here have a Coke!");
            System.out.println("At least you can drive!");
        } else {
            System.out.println("Here have a Coke!");
        }
        System.out.println("Thanks for coming to the Sunset Bar and
Grille!");
        keyboard.close();
    }
}
```

Compile and Run Java Programs at the Command Line

You may find yourself programming in an environment where you may not have a helpful IDE.

1. Start a command prompt in the folder that contains the Java program.

```
// Compile the code file to HelloWorld.class, the bytecode file
javac HelloWorld.java
// Run the bytecode file
java HelloWorld
```

Tutorial 3.1: IsItPositive

This program uses an **if** statement to determine if we have a positive number.

Create and test the following program called **IsItPositive.java**

```java
// Name: IsItPositive.java
// Written by:
// Written on:
// Purpose: Is the number greater than 0

public class IsItPositive {
    public static void main(String[] args) {

        int number = 10;

        // If number is greater than 0
        if (number > 0) {
            System.out.println(number + " is positive.");
        }

        System.out.println("Statement outside if block");
    }
}
```

Example program run:

```
10 is positive.
Statement outside if block
```

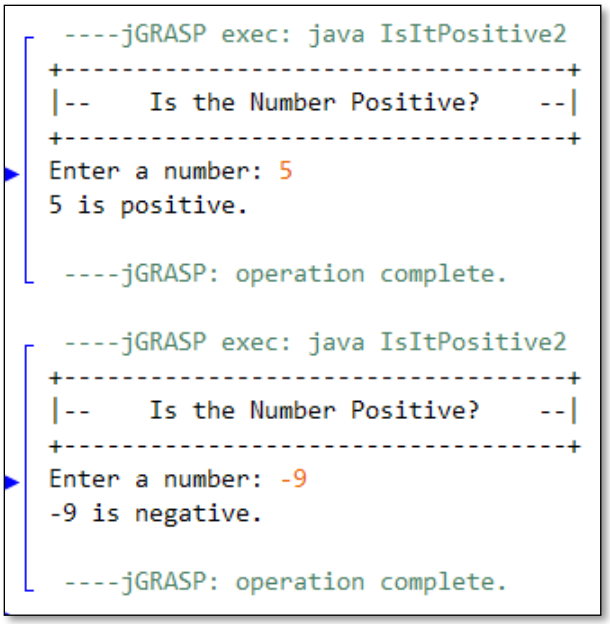
Tutorial 3.2: IsItPositive2

This program uses an **if else** statement to determine if we have a positive or negative number.

Create and test the following program called **IsItPositive2.java**

```java
// Name: IsItPositive2.java
// Written by:
// Written on:
// Purpose: Is the number positive or negative

// Import Scanner library for input
import java.util.Scanner;

public class IsItPositive2 {
    public static void main(String[] args) {
        // Declare Scanner object and initialize with
        // predefined standard input object, System.in
        Scanner keyboard = new Scanner(System.in);

        // Declare integer variable for user input
        int number = 0;

        // Print the heading and prompt
        System.out.println("+------------------------------------+");
        System.out.println("|--    Is the Number Positive?    --|");
        System.out.println("+------------------------------------+");
        System.out.print("Enter a number: ");

        // Get integer from the keyboard
        // Assign integer to variable
        number = keyboard.nextInt();

        // If number is greater than 0
        if (number > 0) {
            System.out.println(number + " is positive.");
        } else {
            System.out.println(number + " is negative.");
        }
        keyboard.close();
    }
}
```

Example run:



Tutorial 3.3: Grades

Create and save the following program as **Grades.java**

This code determines the grade based on the score using a process of elimination:

1. If the score is 90 or above: Assign the grade "A". All scores above 90 have been eliminated.
2. Otherwise, if the score is between 80 and 89: Assign the grade "B". Scores above 90 have already been eliminated.
3. Otherwise, if the score is between 70 and 79: Assign the grade "C".
4. Otherwise, if the score is between 60 and 69: Assign the grade "D".
5. If none of the above conditions are met (score below 60): Assign the grade "F".

This sequence ensures that each score range is evaluated in descending order of achievement (from A to F), and only one grade is assigned based on the first condition that is true.

```java
// Name: Grades.java
// Written by:
// Written on:
// Purpose: Is the number positive or negative

// Import Scanner library for input
import java.util.Scanner;

public class Grades {
    public static void main(String[] args) {
        // Declare Scanner object and initialize with
        // predefined standard input object, System.in
        Scanner keyboard = new Scanner(System.in);

        int score;
        String letterGrade;

        // Prompt the user for input
        System.out.println("+------------------------------------+");
        System.out.println("|--     Please enter a score      --|");
        System.out.println("+------------------------------------+");
        System.out.print("Enter a score: ");

        // Get integer from the keyboard
        // Assign integer to variable
        score = keyboard.nextInt();

        // Determine what the letter grade is
        if (score >= 90) {
            letterGrade = "A";
        } else if (score >= 80) {
            letterGrade = "B";
        } else if (score >= 70) {
            letterGrade = "C";
        } else if (score >= 60) {
            letterGrade = "D";
        } else {
            letterGrade = "F";
        }

        System.out.println("The score " + score + " is: " + letterGrade);
        keyboard.close();
    }
}
```

Example run:

```
+----------------------------------+
|--     Please enter a score      --|
+----------------------------------+
Enter a score: 90
The score 90 is: A

 ----jGRASP: operation complete.

 ----jGRASP exec: java Grades
+----------------------------------+
|--     Please enter a score      --|
+----------------------------------+
Enter a score: 59
The score 59 is: F
```

If we use separate **if** statements, each condition is checked regardless of whether it really needs to be. That is, if the score is a 95, the first program will print an A but then continue on and check to see if the score is a B, C, etc., which is a bit of a waste.

Using **else if**, as soon as we find where the score matches, we stop checking conditions and skip all the way to the end of the whole block of statements.

When using **else if**, the second part of the second if statement condition, **grade < 90**, becomes unnecessary because the corresponding **else if** does not have to worry about a score of 90 or above, as such a score would have already been caught by the first if statement.

Logical Operators

A logical operator is a symbol or word that connects two or more expressions so that the value of the produced expression created is solely determined by the value of the original expressions and the operator's meaning.

Following are the logical operators available in Java.

- and
- or
- not

Operator Symbol	Operator Name	Example	Description
&&	Logical And	Operand-A && Operand-B	It returns True, when both the operand-A and operand-B is True, otherwise it returns False.
||	Logical Or	Operand-A || Operand-B	It returns True, when either the operand-A or operand-B is True, otherwise it returns False.
!	Logical Not	! Operand-A	It returns the operand's logical state in reverse.

Tutorial 3.4: Logical Operators

AND Operator (&&): Checks if both conditions are true. In the example, if it's sunny (sunny is true) and warm (warm is true), the program suggests going to the beach.

OR Operator (||): Checks if at least one of the conditions is true. If it's either sunny or warm (or both), the program suggests having a picnic.

NOT Operator (!): Negates the boolean value. Here, if it's not raining (raining is false), the program suggests playing outdoor games.

```java
public class LogicalBeach {

    Run | Debug | Codiumate: Options | Test this method
    public static void main(String[] args) {
        // Let's define some variables to play with
        boolean sunny = true;
        boolean warm = true;
        boolean raining = false;

        // Example 1: AND Operator (&&)
        // It's a sunny day and warm weather, let's go to the beach!
        if (sunny && warm) {
            System.out.println("AND Operator Example:");
            System.out.println("Let's go to the beach!");
        } else {
            System.out.println("Hmm, maybe another day.");
        }

        // Example 2: OR Operator (||)
        // It's either sunny or warm, or both; let's have a picnic!
        if (sunny || warm) {
            System.out.println("\nOR Operator Example:");
            System.out.println("Let's have a picnic!");
        } else {
            System.out.println("No picnic today.");
        }

        // Example 3: NOT Operator (!)
        // It's not raining, let's play some outdoor games!
        if (!raining) {
            System.out.println("\nNOT Operator Example:");
            System.out.println("Let's play some outdoor games!");
        } else {
            System.out.println("Indoor activities it is.");
        }
    }
}
```

Example run:

```
AND Operator Example:
Let's go to the beach!

OR Operator Example:
Let's have a picnic!

NOT Operator Example:
Let's play some outdoor games!
```

Assignment 1: The Dating Dilemma App

For a single user-supplied age, tell them how old a person must be to have dating be acceptable according to the following folk rule (also called the "creepiness rule":

- You should only date someone who is at least seven years older than half your age.
- The converse is to subtract seven from your age and double it.

For example, an 18 year old needs to date somebody at least 15 year's old.
(18 / 2) + 7 = 15

Requirements

- The dateable age is to be calculated by using the above folk rule.
- Use **separate variables** for storing the user-supplied and dateable ages.
- The displayed dateable age is to be **integer**.

Example runs:

```
    ----    The Dating Dilemma    ----
Enter your age: 24
You can date someone older than: 19
You can date someone younger than: 34
```

```
    ----    The Dating Dilemma    ----
Enter your age: 68
You can date someone older than: 41
You can date someone younger than: 122
```

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show evidence of understanding.

3. Attach the program files.
4. **Command Line Compile:** Attach screenshots showing the successful command line compile and run of one program.
5. Submit in Blackboard.