Algoritma dan Struktur Data

Leon Andretti Abdillah

08

Control Flow Statements Selection by Using Switch and Case

Introduction

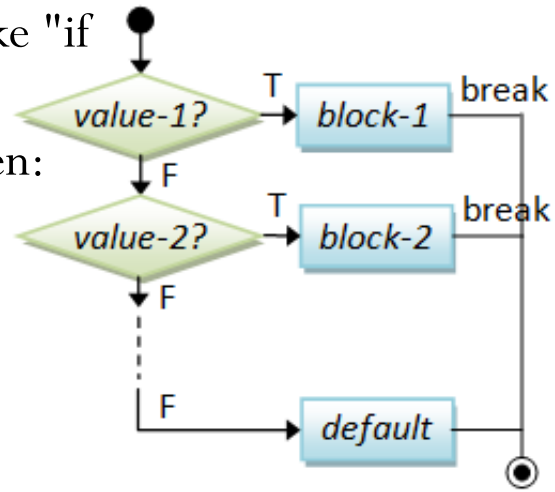
- The if statement allows you to select one of two sections of code to execute based on a boolean value (only two possible values). The switch statement allows you to choose from many statements.
- Unlike if-then and if-then-else statements, the switch statement can have a number of possible execution paths.
- A switch works with the byte, short, char, and int primitive data types.
- It also works with *enumerated types*, string.

Introduction

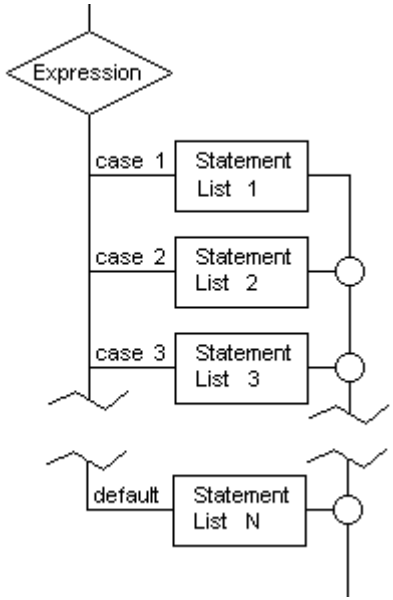
- An alternative to a series of else if is the switch statement.
- The switch statement allows you to choose a block of statements to run from a selection of code, based on the return value of an expression.
- The expression used in the switch statement must return an int or an enumerated value.

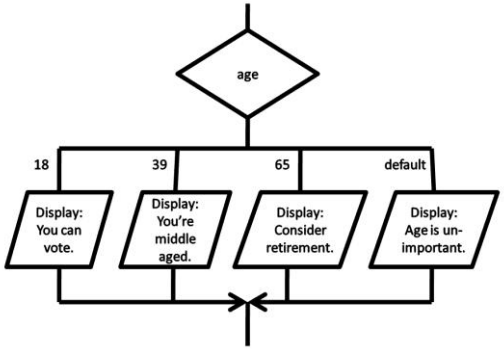
The syntax and flowchart

- In if statement, which reads like "if something is true, then do this before continuing." It is written:




```
if (<Boolean expression>){
      <then-statements>;
 }
 else if {
    <else-statements>;
}
 else  {
    <else-statements>;
}
```





If..else..

```
package Package04;

import java.util.Scanner;

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

        Scanner scDay = new Scanner(System.in);

        System.out.print("Input month no : ");
        int dayNo = scDay.nextInt();
        String dayStr = "";

        if (dayNo == 0) {
            dayStr = "Sunday";
        } else if (dayNo == 1) {
            dayStr = "Monday";
        } else if (dayNo == 2) {
            dayStr = "Tuesday";
        } else if (dayNo == 3) {
            dayStr = "Wednesday";
        } else if (dayNo == 4) {
            dayStr = "Thursday";
        } else if (dayNo == 5) {
            dayStr = "Friday";
        } else if (dayNo == 6) {
            dayStr = "Saturday";
        } else {
            dayStr = "Wrong Day No!";
        }
        System.out.println("Day = " + dayStr);
    }
}
```

```
<terminated> IfElseDemo
Input month no : 1
Day = Monday
```

The syntax of the switch statement

- ```
  switch (expression) {
  case value_1 :

       statement (s);
       break;
  case value_2 :
       statement (s);
       break;
    .
    .
    .
  case value_n :
       statement (s);
       break;
  default:
       statement (s);
  }
  ```

```
switch (expression) {
case cond1: code_block_1;
case cond2: code_block_2;
...
case condn: code_block_n;
default: code_block_default;
}
```
```

Switch

```java
package Package04;

import java.util.Scanner;

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

        Scanner scDay = new Scanner(System.in);
        String dayStr = "";

        System.out.print("Input day code [0-6]");
        int dayCode = scDay.nextInt();

        switch (dayCode) {
            case 0: dayStr = "Sunday";
                    break;
            case 1: dayStr = "Monday";
                    break;
            case 2: dayStr = "Tuesday";
                    break;
            case 3: dayStr = "Wednesday";
                    break;
            case 4: dayStr = "Thursday";
                    break;
            case 5: dayStr = "Frday";
                    break;
            case 6: dayStr = "Saturday";
                    break;
            default:dayStr = "Invalid day code!";
                    break;
        }
        System.out.printf("%nYou type the day code of %d, that is %s ", dayCode, dayStr);
    }
}
```

switch..case

- **`switch`**. The switch keyword is followed by a parenthesized integer expression, which is followed by the *cases*, all enclosed in braces.. The switch statement executes the case corresponding to the value of the expression. Normally the code in a case clause ends with a break statement, which exits the switch statement and continues with the statement following the switch. If there is no corresponding case value, the default clause is executed. If no case matched and there is no default clause, execution continues after the end of the switch statement.
- **`case`**. The case keyword is followed by an integer constant and a colon. This begins the statements that are executed when the switch expression has that case value.

default

- **`default`**. If no case value matches the switch expression value, execution continues at the default clause. This is the equivalent of the "else" for the switch statement. It is written after the last case be convention, and typically isn't followed by break because execution just continues out the bottom of switch if this is the last clause.

Break statement

- **`break`**. The break statement causes execution to exit to the statement after the end of the switch. If there is no break, execution flows thru into the next case. Flowing directly into the next case is almost always an error.
- Failure to add a **`break`** statement after a case will not generate a compile error but may have more serious consequences because the statements on the next case will be executed.

switch-string

- Java 7 allows string base conditional for switch statement

```java
package Package04;

import java.util.Scanner;

public class SwitchMonStr {

    public static void main(String[] args) {

        Scanner scMon = new Scanner(System.in);

        System.out.print("Input month name ");
        String month = scMon.next();
        int monthNumber = 0;

        switch (month.toLowerCase()) {
        case "january"  : monthNumber = 1; break;
        case "february" : monthNumber = 2; break;
        case "march"    : monthNumber = 3; break;
        case "april"    : monthNumber = 4; break;
        case "may"      : monthNumber = 5; break;
        case "june"     : monthNumber = 6; break;
        case "july"     : monthNumber = 7; break;
        case "august"   : monthNumber = 8; break;
        case "september": monthNumber = 9; break;
        case "october"  : monthNumber = 10;break;
        case "november" : monthNumber = 11;break;
        case "december" : monthNumber = 12;break;
        default         : monthNumber = 0; break;
        }
        System.out.printf("The code of month of %s is %d of 12 months",
                month, monthNumber);
    }
}
```

```
<terminated> SwitchMonStr [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (27/10/2012 9:59:05 AM)
Input month name september
The code of month of september is 9 of 12 months
```

switch with char value

- We can use **`charAt()`** method to get a character from any string that the user inputs, even if the user inputs a single letter.
- This method takes a parameter that is the position of the string. The **`charAt()`** method returns a character value for the character at the given position of the string (specified index in string).
- The string indexes start from zero. The string position start from 0.

charAt

- Syntax: **`stringName.charAt(parameter)`**
- Parameter is index (from 0).

- `charAt(i)`: return the **character** at **position** `i` in the string.

Example:

```
                01234567890    <---- Character position
String s = "Hello World";

char   x;

x = s.charAt(0);   // x = 'H';
x = s.charAt(1);   // x = 'e';
x = s.charAt(2);   // x = 'l';
```

```java
package Package04;

import java.util.Scanner;

public class SwitchCharGrade {

    public static void main(String[] args) {

        Scanner scGrade = new Scanner(System.in);

        System.out.print("Input grade [A-F] : ");
        //      String sG = scGrade.next();
        //      char grade = sG.charAt(0);

        char grade = scGrade.next().charAt(0);

        switch(grade) {
        case 'A' : case 'a': System.out.println("Excellent!"); break;
        case 'B' : case 'b': System.out.println("Good Job!"); break;
        case 'C' : case 'c': System.out.println("Well done!"); break;
        case 'D' : case 'd': System.out.println("You almost pass"); break;
        case 'E' : case 'e': System.out.println("You didn't make it"); break;
        case 'F' : case 'f': System.out.println("You are Fail!"); break;
        default  :           System.out.println("Invalid grade");
        }
        System.out.println("Your grade is " + grade);
    }
}
```

```
<terminated> SwitchCharGrade [Java Application]
Input grade [A-F] : a
Excellent!
Your grade is a
```

Enum Types

- Enum is a set range of values.
- Enums were introduced in java 5.0. Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums.
- An *enum type* is a type whose *fields* consist of a fixed set of constants. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week.
- Because they are constants, the names of an enum type's fields are in uppercase letters.
- In the Java programming language, you define an enum type by using the enum keyword. For example, you would specify a days-of-the-week enum type as:

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
    SATURDAY
}
```

```java
package Package04;

public class SwitchEnumDays {

    enum Day {
        SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
        THURSDAY, FRIDAY, SATURDAY
    }

    public static void main(String[] args) {

        Day dy = Day.FRIDAY;

        switch(dy) {
        case SUNDAY:
            System.out.println("Holiday");
            break;
        case MONDAY: case TUESDAY: case WEDNESDAY: case THURSDAY:
            System.out.println("Full working day");
            break;
        case FRIDAY:
            System.out.println("Praying day");
            break;
        case SATURDAY:
            System.out.println("Weekend");
            break;
        default:
            System.out.println("???.");
            break;
        }
    }
}
```

```
<terminated> SwitchEnumDays [Java Application]
Praying day
```

Notes

- Java's switch statement:
  - **No ranges**. It doesn't allow ranges, eg case 90-100:. Many other languages do.
  - **Integers only**. It requires integers and doesn't allow useful types like String (except Java 7 up). Many other languages do.
  - **Error-prone**. It is error-prone and a common source of bugs - forgetting break or default silently ignores errors. Some languages have eliminated these dangerous situations.

Exercise

- Hitunglah berapa jumlah hari pada suatu bulan di tahun tertentu dengan menggunakan switch..case dikombinasikan dengan if..else..
- Sebagai contoh Bulan April 2012 = 30 hari, October 2012 jumlah harinya = 31. Namun untuk bulan February jumlah harinya bisa 28/29. Feb 2013 = 28.

Exercise

- Use selection if and switch to calculate the price for travelling based on transportation vehicle (bus, train, plane)
- Use selection if and switch to calculate the room rent for hotel customers based on room types (standard, deluxe, president)
- Use selection if and switch to calculate the house price for sale based on it types (36, 45, 54)

References

Abdillah, L. A. (2005). Validasi data dengan menggunakan objek lookup pada borland delphi 7.0. *Jurnal Ilmiah MATRIK, 7(1), 1-16.*

Abdillah, L. A. (2009). *Pemrograman II (Delphi Dasar) Edisi 4. Palembang: Pusat Penerbitan dan Percetakan Universitas Bina Darma (PPP-UBD) Press.*

Abdillah, L. A. (2009). *Pemrograman III (Delphi Database) Edisi 4.* Palembang: Pusat Penerbitan dan Percetakan Universitas Bina Darma.

Abdillah, L. A. (2010). Introduction to Java Programming, from http://eprints.binadarma.ac.id/3124/1/Introduction%20to%20Java%20Programming%20-%20%2001%20Introduction.pdf

Abdillah, L. A. (2013). Algorithms & Programming. Retrieved from http://blog.binadarma.ac.id/mleonaa/teaching/programming/algorithm-and-programming-2/

Abdillah, L. A. (2014). Data Structures & Algorithms. Retrieved from http://blog.binadarma.ac.id/mleonaa/teaching/programming/data-structures/

Hilfinger, P. N. (2002). Data Structures (Into Java) Retrieved from http://www.cs.berkeley.edu/~hilfingr/cs61b/f2002/public_html/data-structures.pdf

Lafore, R., & Waite, M. (2003). *Data structures & algorithms in Java (Second Edition ed.). Indianapolis, Indiana, USA: Sams Publishing.*

Mehlhorn, K., & Sanders, P. (2007). Algorithms and Data Structures *The Basic Toolbox (pp. 295).* Retrieved from http://users.dcc.uchile.cl/~nbaloian/cc3001-02/Libros/Mehlhorn-Sanders-Toolbox.pdf

ORACLE. (2015). The Java™ Tutorials from http://docs.oracle.com/javase/tutorial/java/index.html

Wirth, N. (1985). *Algorithms and Data Structures. Zurich, Switzerland.*