Flow of Control: Branching
(Savitch, Chapter 3)

TOPICS

• Conditional Execution
• if and else Statement
• Boolean Data
• switch Statement

if Statement

• Ensures that a statement is executed only when some condition is true
• Conditions typically involve comparison of variables or quantities for equality or inequality
• Example:

```java
if (age >= 18)
    System.out.println("You are eligible to vote.");
```

The if Statement

• The if statement has the following syntax

```java
if (condition) statement;
```

- if is a Java reserved word
- The condition must be a boolean expression. It must evaluate to either true or false.
- If the condition is true, the statement is executed. If it is false, the statement is skipped.

Numeric Relational Operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Java</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>!=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
An if statement may have an optional else clause that will only be executed when the condition is false.

Example:
```
if (wages <= 57600)
tax = 0.124 * wages;
else
tax = 0.124 * 57600;
```

Give an example of when BOTH statements will execute?

Give an example of when NEITHER statement will execute?

To execute more than one statement conditionally, use {} to define a block (aka "compound statement") for the sequence of statements.

Example:
```
if (firstNumber <= secondNumber)
{
    quotient = secondNumber / firstNumber;
    remainder = secondNumber \ firstNumber;
} else
{
    quotient = firstNumber / secondNumber;
    remainder = firstNumber \ secondNumber;
}
```

Code written:
```
if (condition1)
    statement1;
else
    if (condition2)
        statement2;
    else
        statement3;
```

Which if does the else finish?

Be sure to use indentation properly. Otherwise too difficult to read!

```
else will match to the nearest unmatched if within the same block.
```
Fix dangling else using blocks

- Code written:
  
  ```java
  if (condition1) {
    if (condition2) statement1;
  } else statement2;
  ```

boolean Data Type

- boolean
- A primitive data type that can be set to:
- `true`
- `false`
- Example:
  
  ```java
  boolean correct = true;
  ```

boolean Operators

- Logical "and" (conjunction)
  - Java symbol `&&`
  - Math symbol \( \land \)
  - true only when both expressions are true
    
    ```java
    (MINIMUM_WAGE <= wages) && (wages <= MAXIMUM_WAGE)
    ```
- Logical inclusive "or" (disjunction)
  - Java symbol `||`
  - Math symbol \( \lor \)
  - true when either or both expressions are true
    
    ```java
    (wages < MINIMUM_WAGE) || (wages > MAXIMUM_WAGE)
    ```

Boolean Expressions

- Conditions are expressions that have a truth value.
- Arithmetic relational operators produce a truth value, e.g.,
  - 10 < 3
  - x > y
  - a >= (b + 12)
**boolean Operators (cont.)**

- **Logical “exclusive or”**
  - Java symbol `^`
  - Math symbol `⊕`
  - true when exactly one of the expressions is true
    
    \[(\text{MINIMUM\_WAGE} < \text{wages}) \lor (\text{MINIMUM\_WAGE} == \text{wages})\]

- **Logical “not” (negation)**
  - Java symbol `!`
  - Math symbol `¬`
  - `!(\text{MINIMUM\_WAGE} == \text{wages})`

---

**Java Logical and Arithmetic Operator Precedence Rules**

1. `!`  (unary)
2. `*`  `/`  `%`
3. `+`  `-`
4. `<`  `<=`  `>`  `>=`
5. `==`  `!=`
6. `^`  `&`  `|`
7. `&&`  `||`

---

**Complicated Boolean Expressions**

```java
boolean isLeapYear = ((year % 4) == 0) && ((year % 100) != 0) || ((year % 400) == 0);
```

**Interpretation:**

- Leap years are every four years (divisible by 4) except for centuries that are not divisible by 400.

---

**Combining Relational Operators**

- Unlike some other operators, relationals cannot be combined in Java.

- **Example:**
  
  ```java
  \{a <= b <= c\}
  ```
  
  - Does not mean `a <= b` and `b <= c`.
  - It produces a compile-time error -- cannot compare a boolean (return value of `<=` operator) with a number.
  - How should this be done?
**switch Statement**

- Used to accomplish multi-way branching based on the value of an integer selector variable
- Example:

```java
switch (numberOfPassengers) {
    case 0: System.out.println("The Harley");
    break;
    case 1: System.out.println("The Dune Buggy");
    break;
    default: System.out.println("The Humvee");
}
```

**Using break in switch statements**

- Consider the code fragment below

```java
int i = 1;
switch (i) {
    case 0: System.out.println("0");
    break;
    case 1: System.out.println("1");
    break;
    case 2: System.out.println("2");
    break;
    case 3: System.out.println("3");
    } System.out.println();
```

- Without breaks what is the output?

(note: it is legal to leave out the breaks and sometimes desired)

**Symbolic Constants in switch Statements**

```java
final int
SUNDAY = 1, MONDAY = 2, TUESDAY = 3,
WEDNESDAY = 4, THURSDAY = 5, FRIDAY = 6,
SATURDAY = 7;
int d;
... switch (d) {
    case SUNDAY: System.out.print("Sunday"); break;
    case MONDAY: System.out.print("Monday"); break;
    case TUESDAY: System.out.print("Tuesday"); break;
    case WEDNESDAY: System.out.print("Wednesday"); break;
    case THURSDAY: System.out.print("Thursday"); break;
    case FRIDAY: System.out.print("Friday"); break;
    case SATURDAY: System.out.print("Ski day"); break;
} 
```
Multiple case Labels

```java
switch (d) {
    case MONDAY:
    case WEDNESDAY:
    case FRIDAY:
        System.out.println("C.S. meets at 9:00 today");
        System.out.println("Math meets at 10:00 today");
        break;
    case TUESDAY:
    case THURSDAY:
        System.out.println("English meets at 9:00 today");
        System.out.println("Chemistry meets at 10:00 today");
        break;
    case SUNDAY:
    case SATURDAY:
        System.out.println("Enjoy the weekend");
    }
```

## switch example

Display the students’ grade based on entering their grade as an int between 0 and 100 (90+ = A, 80-89 = B, 70-79 = C)

```java
switch (grade / 10) {
    case 10:
    case 9:
        System.out.println("A");
        break;
    case 8:
        System.out.println("B");
        break;
    case 7:
        System.out.println("C");
        break;
    default:
        System.out.println("F");
}
```

### Comparing switch and if statements

#### switch statement

- `switch (expression)`
  
  ```java
  value = expression;
  if (value == value1) statement1;
  else if (value == value2) statement2;
  ...
  case valueX: statementX;
  break;
  default: statementY;
  ```

#### if equivalent

- ```java
  if (letter == 'A' || letter == 'a' || letter == 'E' || letter == 'e' ||
  letter == 'O' || letter == 'o' || letter == 'U' || letter == 'u')
  System.out.println("vowel");
  else System.out.println("consonant");
  ```

#### Comparing switch and if statements

Print out whether the char `ch` is a vowel or not

- `switch (letter)`
  
  ```java
  case 'A': case 'a':
  case 'E': case 'e':
  case 'I': case 'i':
  case 'O': case 'o':
  case 'U': case 'u':
  System.out.println("vowel");
  break;
  default: System.out.println("consonant");
  ```
Summary

- Flow of control
- if statements
- boolean expressions
- if-else statements
- Order of operations
- Relational operators
- Switch statement