# OOP with Java

## 7. Conditionals

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

# Outline

1. Introduction
2. If-Then-Else
3. Switch-Case
4. Summary

website

- Sometimes, we need to make a decision based on the data we have

# Conditionals

- Sometimes, we need to make a decision based on the data we have
- If a certain condition is met, we may do thing $A$ and otherwise thing $B$

# Conditionals

- Sometimes, we need to make a decision based on the data we have
- If a certain condition is met, we may do thing $A$ and otherwise thing $B$
- In other words, we want to structure the flow of our program to perform different actions based on the values of our variables.

- An *if-then* statement in Java has the form `if (cond){ code }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true.

# If-Then-Else

- An *if-then* statement in Java has the form `if (cond){ code }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true.
- An *if-then-else* statement in Java has the form `if (cond){ code } else { otherwise }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true, `otherwise` will be executed if `cond` evaluates to false.

# If-Then-Else

- An *if-then* statement in Java has the form `if (cond){ code }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true.
- An *if-then-else* statement in Java has the form `if (cond){ code } else { otherwise }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true, `otherwise` will be executed if `cond` evaluates to false.
- Inside the `code` and `otherwise` blocks, there can be arbitrarily many other commands

- An *if-then* statement in Java has the form `if (cond){ code }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true.
- An *if-then-else* statement in Java has the form `if (cond){ code } else { otherwise }`, where `cond` is a `boolean` expression. `code` is executed if `cond` evaluates to true, `otherwise` will be executed if `cond` evaluates to false.
- Inside the `code` and `otherwise` blocks, there can be arbitrarily many other commands, including more conditionals...

# If-Then-Else Example

## Listing: Examples for if-then and if-then-else.

```
import java.util.Scanner; // import the scanner class: ignore this for now

/** Examples for using if-then-else */
public class HelloIfThenElse {

  /** The main routine
   * @param args
   *           we ignore this parameter for now */
  public static final void main(String[] args) {
    Scanner scanner = new Scanner(System.in); // initiate reading from System.in, ignore for now

    System.err.println("Please␣enter␣your␣name:␣"); // <--- using System.err for status //$NON-NLS-1$
    String name = scanner.nextLine(); // read next line from input and store in variable "string"

    System.err.println("Please␣your␣gender␣[f=female,␣m=male]:␣"); // <--- using System.err for status
        //$NON-NLS-1$
    char gender = scanner.next().charAt(0); // read the next character from stdin

    System.out.print("Hello␣"); //$NON-NLS-1$
    if (gender == 'f') {
      System.out.print("Mrs.␣"); //$NON-NLS-1$
    } else {
      if (gender == 'm') {
        System.out.print("Mr.␣"); //$NON-NLS-1$
      }
    }
    System.out.println(name);
  }
}
```

# Switch-Case

- Nesting many *if-then-else* can be complicated
- For cases where we make decisions based on the values of `char`, integer type, or `String`, we can use the *switch-case* statement:

Listing: The structure of *switch-case*

```
switch(expression) { // expression must be char, integer, or String-valued
  case value1: {
    // what to do if expression == value1
    break; // exit switch-case statement
  }
  case value2: {
    // what to do if expression == value2
    // here I leave "break" away, i.e., we fall-through
  }
  case value3: {
    // what to do if expression == value3 OR expression == value2
    break; // exit switch-case statement
  }
  // ...
  default: { // optional
    // what to do if expression is different from all of the above
  }
}
```

# Switch-Case Example

## Listing: Examples for switch-case.

```java
import java.util.Scanner; // import the scanner class: ignore this for now

/** Examples for using if-then-else */
public class HelloSwitchCase {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    Scanner scanner = new Scanner(System.in); // initiate reading from System.in, ignore for now

    System.err.println("Please enter your name: "); // <--- using System.err for status //$NON-NLS-1$
    String name = scanner.nextLine(); // read next line from input and store in variable "string"

    System.err.println("Please your gender [f=female, m=male]: "); // <--- using System.err for status //$NON-NLS-1$

    String gender = scanner.next(); // read the gender from stdin
    switch (gender) { // choose what to do based on gender
      case "f": //$NON-NLS-1$
      case "F": {// we will get here if gender is either "f" or "F" //$NON-NLS-1$
        System.out.print("Mrs. "); //$NON-NLS-1$
        break;
      }

      case "m": //$NON-NLS-1$
      case "M": {// we will get here if gender is either "m" or "M" //$NON-NLS-1$
        System.out.print("Mr. "); //$NON-NLS-1$
        break;
      }

      default: { // we will get here if the gender is neither "f", "F", "m", "M"
        System.out.print(gender);
        System.out.print(' ');
        break;
      }
    }
    System.out.println(name);
  }
}
```

# Switch-Case: More Explanations

- We can write multiple `case` conditions and the `case` body is executed if any of them is met (like with logical `or`)

- We can write multiple `case` conditions and the `case` body is executed if any of them is met (like with logical `or`)
- The `default` body is executed if no `case` condition is met

- Did you notice the little `break` keyword at the end of each `case`-body?

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means "we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)"

## Switch-Case Fall-Through

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means “we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)”
- Why do you think it is there?

# Switch-Case Fall-Through

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means "we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)"
- Why do you think it is there?
- If a `case` condition was met and the corresponding body does not contain any `break`, then the control flow will simply continue by executing the body of the next `case` statement, until it hits a `break` or the end of the `switch` body

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means "we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)"
- Why do you think it is there?
- If a `case` condition was met and the corresponding body does not contain any `break`, then the control flow will simply continue by executing the body of the next `case` statement, until it hits a `break` or the end of the `switch` body
- This is called fall-though

# Switch-Case Fall-Through

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means "we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)"
- Why do you think it is there?
- If a `case` condition was met and the corresponding body does not contain any `break`, then the control flow will simply continue by executing the body of the next `case` statement, until it hits a `break` or the end of the `switch` body
- This is called fall-though
- Fall-throughs into the `default` case are allowed

# Switch-Case Fall-Through

- Did you notice the little `break` keyword at the end of each `case`-body?
- `break` here basically means "we are finished with this `switch`, jump to the end of it (and then execute the next instruction, if any)"
- Why do you think it is there?
- If a `case` condition was met and the corresponding body does not contain any `break`, then the control flow will simply continue by executing the body of the next `case` statement, until it hits a `break` or the end of the `switch` body
- This is called fall-though
- Fall-throughs into the `default` case are allowed
- This makes code very hard to read.

# Switch-Case Fall-Through Example

## Listing: Examples for switch-case fall-through.

```java
import java.util.Scanner; // import the scanner class: ignore this for now

/** Examples for using if-then-else */
public class HelloSwitchCaseFallThrough {

  /** The main routine
   * @param args
   *          we ignore this parameter for now */
  public static final void main(String[] args) {
    Scanner scanner = new Scanner(System.in); // initiate reading from System.in, ignore for now

    System.err.println("Please enter your name: "); // <--- using System.err for status //$NON-NLS-1$
    String name = scanner.nextLine(); // read next line from input and store in variable "string"

    System.err.println("Please your gender [f=female, m=male]: "); // <--- using System.err for status //$NON-NLS-1$

    String gender = scanner.next(); // read the gender from stdin
    switch (gender) { // choose what to do based on gender
      case "f": //$NON-NLS-1$
      case "F": {// we will get here if gender is either "f" or "F" //$NON-NLS-1$
        System.out.print("Mrs. "); //$NON-NLS-1$
        // break; <- We comment out the break, so if the user types in "f", we will fall-through to the next
        // body after printing Mrs., i.e., we will print "Mrs. Mr. " as greeting
      }

      //$FALL-THROUGH$
      case "m":   // we will get here if gender is either "m" or "M"       //$NON-NLS-1$
      case "M": { // or if the user entered "f" or "F" and we fell-through //$NON-NLS-1$
        System.out.print("Mr. "); //$NON-NLS-1$
        break;
      }

      default: { // we will get here if the gender is neither "f", "F", "m", "M"
        System.out.print(gender);
        System.out.print(' ');
        break;
      }
    }
    System.out.println(name);
  }
}
```

## Summary

- We have learned how to make decisions in a program.
- We can make binary decisions in the form of "if-this-then-do-that-otherwise-do-that-other-thing"
- We can make more complex choices by "switching" over the values of an expression

# 谢谢

# Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog