Software Development (CS2500)

Lecture 8: Making Decisions

M. R. C. van Dongen

October 9, 2013

Decision Making

- One of the computer's most important tasks is decision making.
- E.g. consider a computer-aided, intelligent car.
- It is supposed to go from source to destination.
- When the car is at a junction, which direction should it go?
  - The car needs to make a decision.
  - Without the decision it'll probably never reach its destination.

Overview

- There are three constructs that affect the flow of control.
  - The `if` statement;
  - The `if-else` statement; and
  - The `switch` statement.
- The first two constructs depend on `boolean` expressions.
- For the moment we shall forget about the `switch` statement.

The `if` Statement

- The conditional or `if` statement is written as follows:

Java

```
if (⟨condition⟩)
  ⟨statement⟩
```

- ⟨`statement`⟩ is carried out iff[1] ⟨`condition`⟩ is `true`.
- Notice that ⟨`statement`⟩ is an expression.

[1] Pronounce if and only if.

The `if` Statement

- The conditional or `if` statement is written as follows:

Java

```
if (⟨condition⟩) {
  ⟨statement⟩
}
```

- ⟨`statement`⟩ is carried out iff[1] ⟨`condition`⟩ is `true`.
- Notice that ⟨`statement`⟩ is an expression.

[1] Pronounce if and only if.

First Examples

Java

```
if (true) {
    System.out.println( "This is printed." );
}
```

First Examples

Java

```
if (true) {
    System.out.println( "This is printed." );
}

if (false) {
    System.out.println( "This isn't printed." );
}
```

Example

Java

```
private static final double FREEZING_TEMPERATURE_OF_WATER = 0.0;
.
.
.

final Thermometer thermometer = new Thermometer( );
final double temperature = thermometer.read( );

if (temperature < FREEZING_TEMPERATURE_OF_WATER) {
    System.out.println( "It's freezing." );
}
```

Conditions

⟨fst⟩ == ⟨snd⟩: true iff ⟨fst⟩ is equal to ⟨snd⟩.
⟨fst⟩ != ⟨snd⟩: true iff ⟨fst⟩ is not equal to ⟨snd⟩.
⟨fst⟩ < ⟨snd⟩: true iff ⟨fst⟩ is less than ⟨snd⟩.
⟨fst⟩ <= ⟨snd⟩: true iff ⟨fst⟩ is less than or equal to ⟨snd⟩.
⟨fst⟩ > ⟨snd⟩: true iff ⟨fst⟩ is greater than ⟨snd⟩.
⟨fst⟩ >= ⟨snd⟩: true iff ⟨fst⟩ is greater than or equal to ⟨snd⟩.

Examples

Comparison	Result

Examples

Comparison	Result
0 == 0	

Examples

Comparison	Result
`0 == 0`	`true`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	`true`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	`true`
`3 == 9 / 3`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	`true`
`3 == 9 / 3`	`true`

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	`true`
`3 == 9 / 3`	`true`
`3 == 10 / 3`	

Examples

Comparison	Result
`0 == 0`	`true`
`0 == 1`	`false`
`0 < 2`	`true`
`2 < 0`	`false`
`0 < 0`	`false`
`1 <= 0`	`false`
`0 <= 2`	`true`
`3 == 9 / 3`	`true`
`3 == 10 / 3`	`true`

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`		

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`	`true`	

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`	`true`	trick question

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`	`true`	trick question
`x <= --x`		

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`	`true`	trick question
`x <= --x`	`false`	

Examples

Comparison	Result	
`0 == 0`	`true`	
`0 == 1`	`false`	
`0 < 2`	`true`	
`2 < 0`	`false`	
`0 < 0`	`false`	
`1 <= 0`	`false`	
`0 <= 2`	`true`	
`3 == 9 / 3`	`true`	
`3 == 10 / 3`	`true`	integer division
`x <= x--`	`true`	trick question
`x <= --x`	`false`	trick question

The `if-else` Statement

Java

```
if (⟨condition⟩)
  ⟨first statement⟩
 else
  ⟨second statement⟩
```

The `if-else` Statement

Java

```
if (⟨condition⟩) {
  ⟨first statement⟩
} else {
  ⟨second statement⟩
}
```

Example

Java

```
if (temperature < FREEZING_TEMPERATURE_OF_WATER) {
    System.out.println( "It's freezing." );
} else {
    System.out.println( "It's not freezing." );
}
```

An Invariant

Assuming `condition` is Side-Effect Free

Java

```
if (⟨condition⟩) {
    // ⟨condition⟩ is true
    ⟨statements⟩
} else {
    // ⟨condition⟩ is not true
    ⟨more statements⟩
}
```

Parsing `if` Statements

Java

```
if (condition1)
    if (condition2)
        ⟨stuff⟩
    else
        ⟨more stuff⟩
```

is equivalent to

Java

```
if (condition1) {
    if (condition2) {
        ⟨stuff⟩
    } else {
        ⟨more stuff⟩
    }
}
```

Dangling `else`

Java

```
if (condition1)
    if (condition2)
        ⟨stuff⟩
else
    ⟨more stuff⟩
```

Conditional Operator

- Consider the following scenario.
  - You have a variable, `var`;
  - You have to assign it a value;
  - The value depends on a condition:
    - If the condition holds, you want to assign `x`;
    - Otherwise, you want to assign `y`;
- This is such a common scenario that `Java` has an operator for it.
- The operator is weird because it's a ternary operator:

Java

```
final int var = condition ? x : y;
```

Conditional Operator

- Consider the following scenario.
  - You have a variable, `var`;
  - You have to assign it a value;
  - The value depends on a condition:
    - If the condition holds, you want to assign `x`;
    - Otherwise, you want to assign `y`;
- This is such a common scenario that `Java` has an operator for it.
- The operator is weird because it's a ternary operator:

Java

```
final int var = (condition) ? x : y;
```

Conditional Operator

- Consider the following scenario.
  - You have a variable, `var`;
  - You have to assign it a value;
  - The value depends on a condition:
    - If the condition holds, you want to assign `x`;
    - Otherwise, you want to assign `y`;
- This is such a common scenario that `Java` has an operator for it.
- The operator is weird because it's a ternary operator:

Java

```
final int var = condition ? x : y;
```

- In general `x` and `y` can be expressions.

Example

Java

```
final String message = temperature < 0.0 ? "It's freezing." : "It's not freezing";
System.out.println( message );
```

compareTo( )

- Many classes define an instance method `compareTo( )`.
- The method is used to compare class instances.
- You call it like this:
  - `this.compareTo( that )`.
- The return value is an `int`:

  negative `this` is less significant than `that`;
  zero `this` and `that` are the same (incomparable);
  positive `this` is more significant than `that`.

Example

Java

```
final String firstWordInDictonary = "a";
final String lastWordInDictonary = "Zyrian";

final int cmp = firstWordInDictionary.compareTo( lastWordInDictonary );
if (cmp < 0) {
    System.out.println( firstWordInDictonary
                        + " comes first." );
} else if (cmp > 0) {
    System.out.println( lastWordInDictonary
                        + " comes first." );
} else {
    System.out.println( "neither "
                        + firstWordInDictonary
                        + " nor "
                        + lastWordInDictonary
                        + " comes first." );
}
```

Shallow Equality

- You may also compare object reference values with == and `!=`.
- This works as expected:
  - `first == second` is `true` iff
    - `first` and `second` are both equal to `null`; or
    - `first` and `second` both reference the same object.
  - `first != second` is `true` iff `first == second` is `false`.
- A comparison of object reference values with == is called a *shallow comparison*.

Pitfalls Shallow Equality

Don't Try This at Home

```
final Scanner scanner = new Scanner( System.in );
final String firstWord = scanner.next( );
final String lastWord = scanner.next( );

if (firstWord == lastWord) {
    System.out.println( "The words are the same!" );
}
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." );
}

System.out.println( first );
System.out.println( second );
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );
System.out.println( second );
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );  // prints
System.out.println( second );
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );  // prints 1.0000000000000002
System.out.println( second );
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );  // prints 1.0000000000000002
System.out.println( second ); // prints
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );  // prints 1.0000000000000002
System.out.println( second ); // prints 1.0
```

Comparing FP Values

- We've already seen that FP operations may result in rounding.
- This may result in strange anomalies.

Java

```
final double SMALL = 1.0E-16;
final double BIG = 1.0;

final double first = SMALL + SMALL + BIG;
final double second = BIG + SMALL + SMALL;

if (first == second) {
    System.out.println( "They are equal." ); // not executed
}

System.out.println( first );  // prints 1.0000000000000002
System.out.println( second ); // prints 1.0
```

- Clearly, `first != second`, but they're almost the same.
- So, how do you compare FP values?

Comparing FP Values (Continued)

- Comparing FP values is like going to the English Market.
- When you and your friend buy 1 kg of apples, each:
  - You both pay the same price;
  - Yet the apples you got have different weights.
  - The weights are (supposed to be) roughly the same;
  - But there's a small difference, a threshold.
- When you compare FP values, you usually also need a threshold.
- The value of the threshold depends on the application:
  - Then you compare weights of ants, it should be small;
  - When you compare the distance of starts, it should be large.

Example

Java

```
private static final double KILOGRAM = 1.0;
private static final double EPSILON = 10.0E-3 * KILOGRAM; // 10 grams
:

/**
 * Compare the weights of two bags of apples bought on the market.
 *
 * @param firstWeight The weight of the first bag in kilograms
 * @param secondWeight The weight of the first bag in kilograms
 * @return true iff the weights are approximately the same.
 */
public static boolean compareWeights( final double firstWeight,
                                      final double secondWeight ) {
    final double difference = firstWeight - secondWeight;
    final boolean result;

    if (difference < -EPSILON) {
        result = false;
    } else if (EPSILON < difference) {
        result = false;
    } else {
        result = true;
    }
    return result;
}
```

Example (`Math.abs( )` to the Rescue

Java

```
private static final double KILOGRAM = 1.0;
private static final double EPSILON = 10.0E-3 * KILOGRAM; // 10 grams
.
.
.

/**
 * Compare the weights of two bags of apples bought on the market.
 *
 * @param firstWeight The weight of the first bag in kilograms
 * @param secondWeight The weight of the first bag in kilograms
 * @return true iff the weights are approximately the same.
 */
public static boolean compareWeights( final double firstWeight,
                                      final double secondWeight ) {
    return Math.abs( firstWeight - secondWeight ) <= EPSILON;
}
```

For Friday

- Study Chapter 3.
- Read Sections 4.1–4.2.

Acknowledgements

- This lecture corresponds to [*Big Java, Early Objects*, 3.1–3.2].

Bibliography

Horstfmann, Cay S. *Big Java, Early Objects.* International Student Version. Wiley. ISBN: 978-1-118-31877-5.

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.