CS1150
Principles of Computer Science
Boolean, Selection Statements

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Review

- What happens when we create a Scanner?
  - Scanner input = new Scanner(System.in);
  - int value = input.nextInt();
  - // close the Scanner

Overview

- Boolean data type
- If statement
- Switch statement

The boolean Type and Operators

Often a program needs to compare two values, such as whether i is greater than j.

Java provides six comparison operators (aka relational operators) to compare two values. See next slide.

Result of the comparison is a boolean value: true or false.

```
boolean b = (1 > 2);
```

Similar to int, float, double, etc., boolean is just another data type.

Relational Operators

Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
`<`	<	less than	`radius < 0`	`false`
`<=`	≤	less than or equal to	`radius <= 0`	`false`
`>`	>	greater than	`radius > 0`	`true`
`>=`	≥	greater than or equal to	`radius >= 0`	`true`
`==`	=	equal to	`radius == 0`	`false`
`!=`	≠	not equal to	`radius != 0`	`true`

```
boolean b = (1 > 2);
```

true and false are boolean literals

Logical Operators

Operator	Name	Description
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

Truth Table for Operator ! (not)

p	!p	Example (assume age = 24, weight = 140)
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(weight == 150) is true, because (weight == 150) is false.

Truth Table for Operator && (and)

p_1	p_2	p_1 && p_2	Example (assume age = 24, weight = 140)
false	false	false	(age <= 18) && (weight < 140) is false, because both conditions are false.
false	true	false	
true	false	false	(age > 18) && (weight > 140) is false, because (weight > 140) is false.
true	true	true	(age > 18) && (weight >= 140) is true, because both (age > 18) and (weight >= 140) are true.

Truth Table for Operator || (or)

p_1	p_2	p_1 || p_2	Example (assume age = 24, weihgt = 140)
false	false	false	
false	true	true	(age > 34) || (weight <= 140) is true, because (age > 34) is false, but (weight <= 140) is true.
true	false	true	(age > 14) || (weight >= 150) is true, because (age > 14) is true.
true	true	true	

Truth Table for Operator ^ (exclusive or)

p_1	p_2	p_1 ^ p_2	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) ^ (weight > 140) is false, because (age > 34) is false and (weight > 140) is false.
false	true	true	(age > 34) ^ (weight >= 140) is true, because (age > 34) is false but (weight >= 140) is true.
true	false	true	(age > 14) ^ (weight > 140) is true, because (age > 14) is true and (weight > 140) is false.
true	true	false	

Short Circuit Evaluation

- We stop evaluating a **boolean expression** as soon as its value can be determined
- This happens when have a compound expressions involving && and ||
  - **&&** - stop once an expression that evaluated to false is found
  - **||** - stops once an expression that evaluates to true is found
  - Example: Boolean1.java

Order of Operators (Section 3.15)

- Anything in parentheses
- expr++ expr-- (postfix)
- + - ++expr --expr (unary plus/minus, prefix)
- (type) (Casting)
- **!** **(not)**
- * / % (multiplication, division, remainder)
- + - (binary addition, subtraction)
- < <= > >= (relational operators)
- == != (equality)
- **^** **(exclusive or)**
- **&&** **(and)**
- **||** **(or)**
- = += -= *= /= %= (assignment, augmented assignment)

Let's see Boolean1.java as an example

Selection Statements

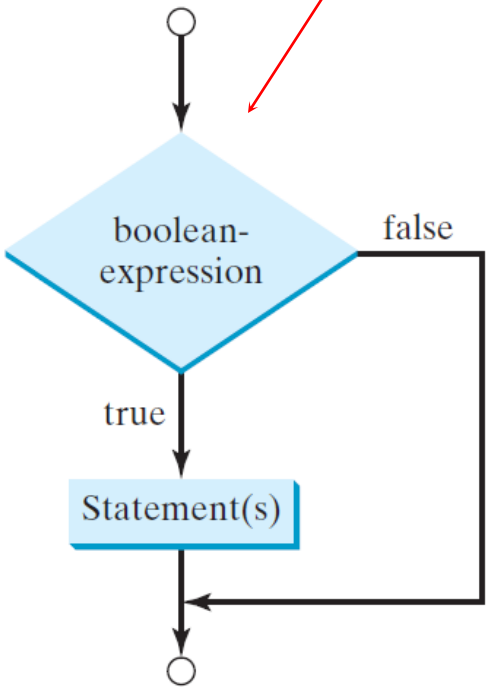
- Homework 1, Question 4 we assumed r>0
  - Calculate surface area of a sphere with radius r (user input)
  - Everything is fine as long as r is a positive number
- What if the user input is a negative number?
  - Write code that can handle user input that is not correct
- Types of selection statements
  - if statement
  - switch statement

If Statement

- Select a path of execution based on a condition
  - A condition is evaluated (i.e. boolean expression)
  - If the condition is true
    - The statements in the true branch are executed
  - If the condition is false
    - The statements in false branch are executed (if there is an "else")

One-way if Statements

```
if (boolean-expression) {
  statement(s);
}
```



One-way if Statements

```
if i > 0 {
   System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
   System.out.println("i is positive");
}
```

(b) Correct

```
if (i > 0) {
   System.out.println("i is positive");
}
```

(a)

Equivalent

```
if (i > 0)
   System.out.println("i is positive");
```

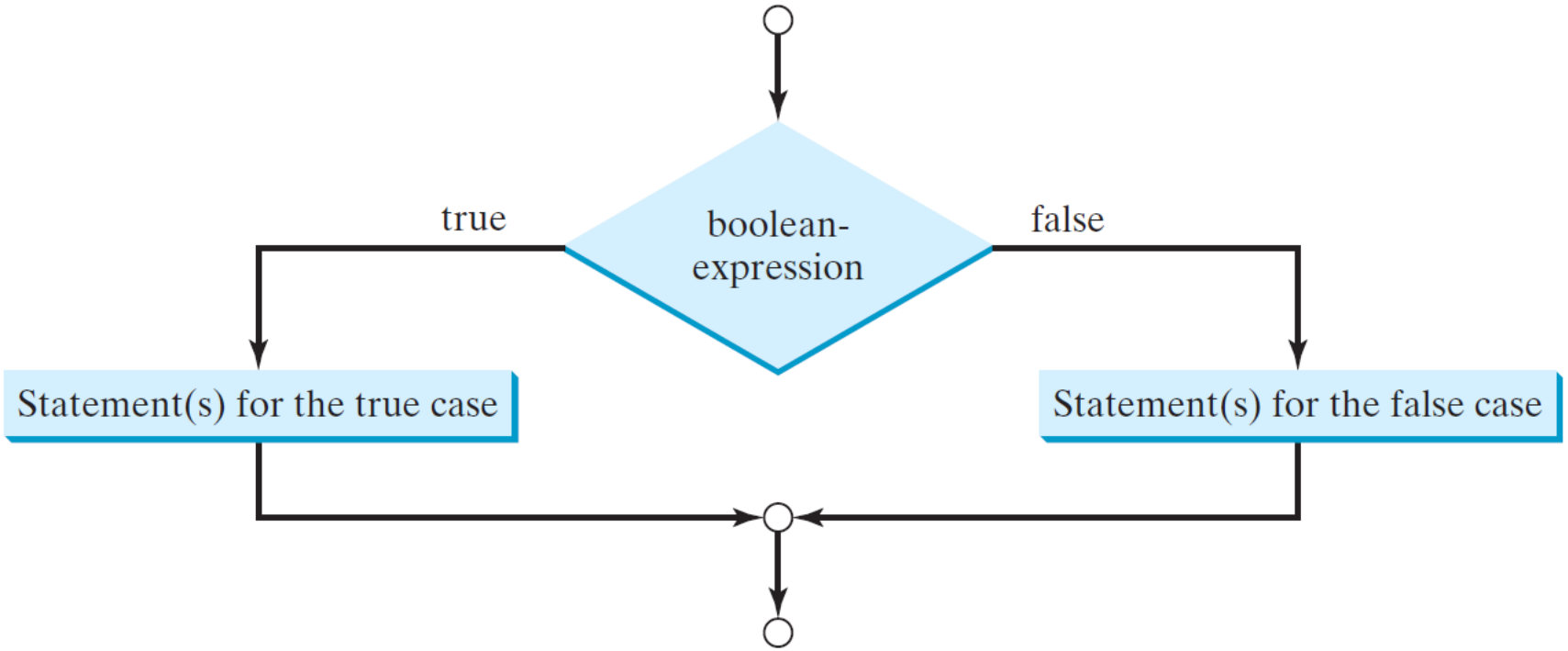
(b)

- But only one statement can omit {}
- Let's see code Boolean2.java, Even1.java

Two-way if Statement

```
if (boolean-expression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```

Example: Even2.java



Multiple Alternative if Statements

```
if (score >= 90.0)
  System.out.print("A");
else
  if (score >= 80.0)
    System.out.print("B");
  else
    if (score >= 70.0)
      System.out.print("C");
    else
      if (score >= 60.0)
        System.out.print("D");
      else
        System.out.print("F");
```

(a)

Equivalent

This is better

```
if (score >= 90.0)
  System.out.print("A");
else if (score >= 80.0)
  System.out.print("B");
else if (score >= 70.0)
  System.out.print("C");
else if (score >= 60.0)
  System.out.print("D");
else
  System.out.print("F");
```

(b)

The else clause matches the **most recent** if clause

Example: ComputeAndInterpretBMI.java

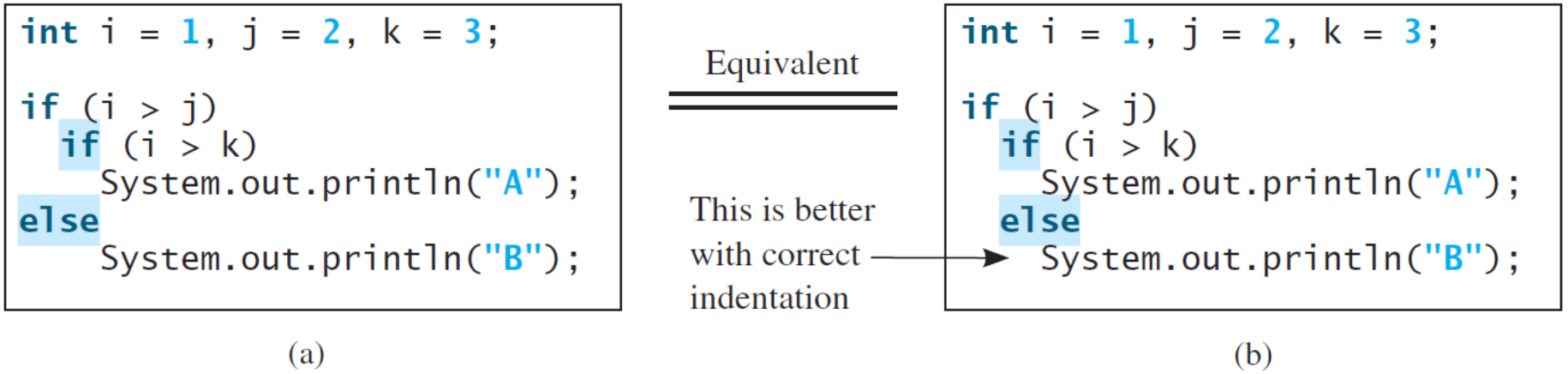
Problem: Body Mass Index

Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
BMI < 18.5	Underweight
18.5 <= BMI < 25.0	Normal
25.0 <= BMI < 30.0	Overweight
30.0 <= BMI	Obese

If statement notes

- The else clause matches the most recent if clause



- What's the output?

If statement notes

To force the else clause to match the first if clause, must add a pair of braces:

```
int i = 1, j = 2, k = 3;
if (i > j) {
  if (i > k)
    System.out.println("A");
}
else
  System.out.println("B");
```

If statement summary

- Block statements {} can be omitted if there is only one statement
- Recommendation
  - **Use curly braces: it helps make the code easier to modify and less error prone**
- Avoid deeply nested if statements
- An ”else" clause always belongs with the most recent if clause

Summary

- Numeric literals
- Data casting
- Boolean data type
- If statement