Java Programming

Zheng-Liang Lu

Department of Computer Science & Information Engineering
National Taiwan University

Online Course

```
1 class Lecture3 {
2
3                     "Flow Controls: Selection"
4
5 }
6
7 // Keywords:
8 if, else, switch, case, default, break
```

Flow Controls

- We proceed to introduce the building blocks of algorithms as follows.
- First, most of statements are executed in order.
- A program can handle with multiple situations if the branching (selection) rules are known.
- Moreover, the program can repeat actions if necessary.
- For example, remember how to find the maximum in the input list?

Representation for Branching

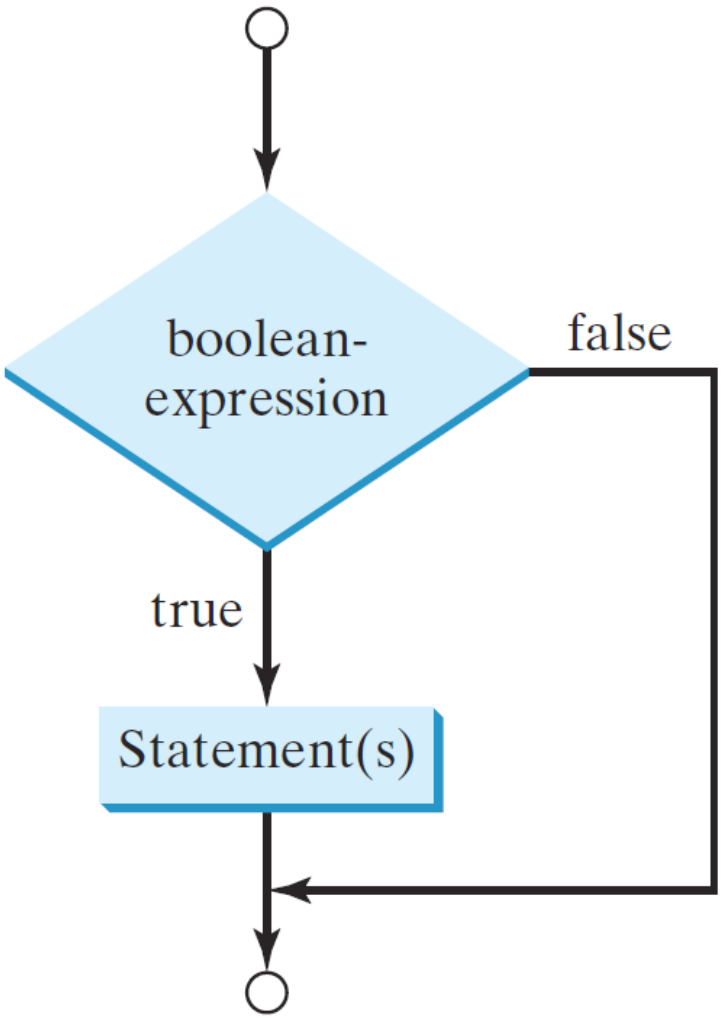
- Conditional statements by if-else.
- Conditional statements by switch-case-break-default.
- Conditional operators.

Branching Statements by if

- The syntax is simple, shown below.

```
1 ...
2         if (/* Condition: a boolean expression */) {
3             // Selection body: conditional statements.
4         }
5 ...
```

- If the condition is evaluated true, then the conditional statements will be executed once.
- If false, then the selection body will be ignored (or we say that the program jumps to Line 5).
- Note that the braces can be omitted if the body contains only single statement.


boolean-
expression
false
true
Statement(s)

Example: Circle Area (Revisited)

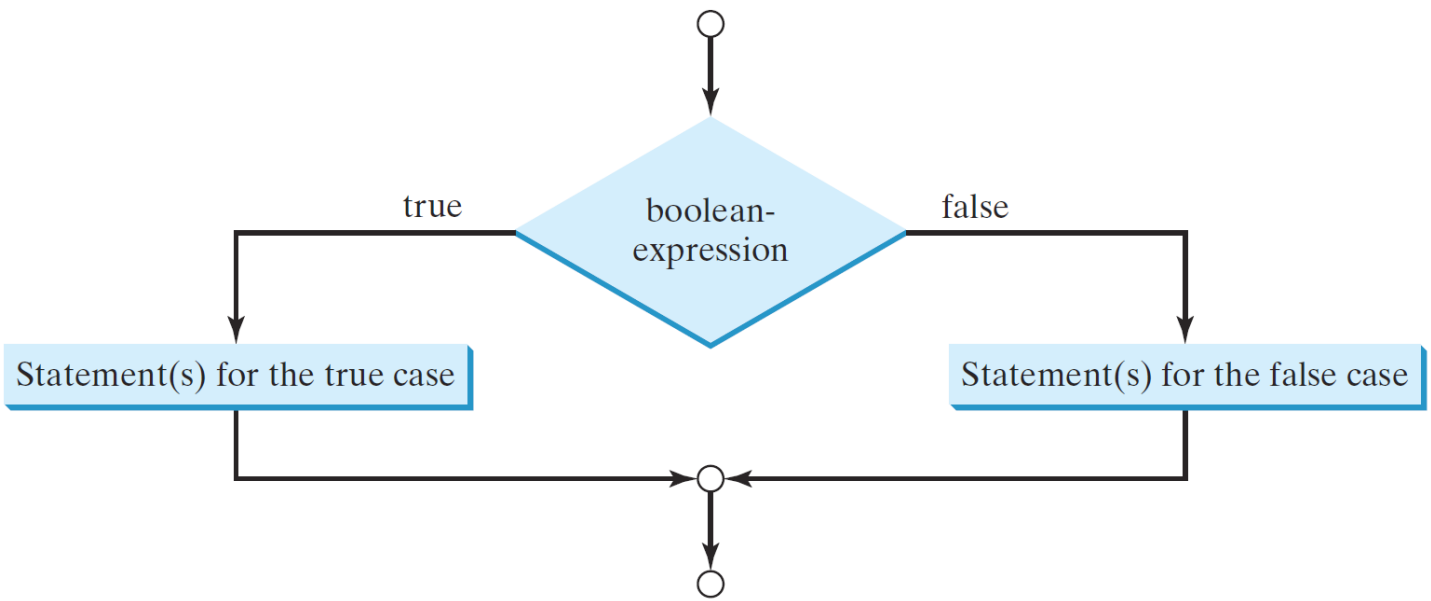
Write a program which receives a positive number as the circle radius and outputs the resulting area.

```
1 ...
2         if (r > 0) {
3             double A = r * r * 3.14;
4             System.out.println(A);
5         }
6 ...
```

- What if the false case?

The if-else Statements

```
1 ...
2         if (/* Condition: a boolean expression */) {
3             // Body for the true case.
4         } else {
5             // Body for the false case.
6         }
7 ...
```



Example: Circle Area (Revisited)

- Now add a conditional statement for the false case.

```
1 ...
2         if (r > 0) {
3             double A = r * r * 3.14;
4             System.out.println(A);
5         } else {
6             System.out.println("Not a circle.");
7         }
8 ...
```

Nested Conditional Statements by Example

- Write a program to convert percentage grades to letter grades.

```
...
        if (score >= 90)
            System.out.println("A");
        else {
            if (score >= 80)
                System.out.println("B");
            else {
                if (score >= 70)
                    System.out.println("C");
                else {
                    if (score >= 60)
                        System.out.println("D");
                    else
                        System.out.println("F");
                }
            }
        }
...
```

Multiple Branches

```
1  ...
2          if (score >= 90)
3              System.out.println("A");
4          else if (score >= 80)
5              System.out.println("B");
6          else if (score >= 70)
7              System.out.println("C");
8          else if (score >= 60)
9              System.out.println("D");
10         else
11             System.out.println("F");
12 ...
```

- Easier to read!
- We should avoid deep indentation.

- The alternative to the previous program looks like:

```
1 ...
2         if (score >= 90 && score <= 100)
3             System.out.println("A");
4         else if (score >= 80 && score < 90)
5             System.out.println("B");
6         else if (score >= 70 && score < 80)
7         ...
8 ...
```

- However, the order of conditions may be relevant. (Why?)
- The performance may degrade due to the order of conditions. (Why?)

Common Bugs

```
1 ...
2         if (r > 0);
3             double A = r * r * 3.14;
4             System.out.println(A);
5 ...
```

- Don't add a semicolon to the condition (in Line 2).
  - If you do so in Line 2, this statement is not effective (useless).
- Multiple conditional statements should be grouped by braces.

Example with Uncertainty

Write a program which first shows a math question, say sum of two random integers ranging from 0 to 9, and asks the user to type an answer.

- For example, the program shows $2 + 5 =$?
- If the user types 7, then the program reports "Correct."
- Otherwise, the program reports "Wrong answer. The correct answer is 7."
- You may use **Math**.random() for a random value between 0.0 and 1.0, excluding themselves.

Digression: Random Number Generation (RNG)[1]

- **Math**.random() produces numbers only between 0.0 and 1.0, exclusive.
- To generate any integer ranging 0 to 9, we could do

$$(\text{int})\ (\textbf{Math}.\text{random}() \times 10),$$

  because there are 10 possible states: 0, 1, 2,. . . , 9.
- In general, you could generate any integer between L and U by using

$$(\text{int})\ (\textbf{Math}.\text{random}() \times (\text{U} - \text{L} + 1)) + \text{L}.\ (\text{Why?})$$

[1] See
https://en.wikipedia.org/wiki/Pseudorandom_number_generator

```
...
        // (1) Generate two random integers.
        int x = (int) (Math.random() * 10);
        int y = (int) (Math.random() * 10);

        // (2) Display the math question.
        System.out.println(x + " + " + y + " =  ?");

        // (3) Ask the user to type his/her answer.
        Scanner input = new Scanner(System.in);
        int z = input.nextInt();
        input.close();

        // (4) Judge the input.
        if (z == x + y) {
            System.out.println("Correct.");
        } else {
            System.out.println("Wrong.");
            System.out.println("It is " + (x + y) + ".");
        }
...
```

- Can you extend this program for all arithmetic operators $(+ - \times \div)$?

"Exploring the unknown requires tolerating uncertainty."

– Brian Greene

"I can live with doubt, and uncertainty, and not knowing. I think it is much more interesting to live not knowing than have answers which might be wrong."

– Richard Feynman

Exercise

Write a program which outputs the maximum value in 3 random integers ranging from -50 to 50.

- Recall the first algorithm example in our class.

```
1  ...
2          int x = (int) (Math.random() * 101) - 50;
3          int y = (int) (Math.random() * 101) - 50;
4          int z = (int) (Math.random() * 101) - 50;
5
6          int max = x;
7          if (y > max) max = y;
8          if (z > max) max = z;
9          System.out.println("max = " + max);
10 ...
```

- This program is limited by the number of data.
- To develop a reusable solution, we need arrays and loops.

The switch-case-break-default Statements

```
1  ...
2          switch (target) {
3              case v1:
4                  // Conditional statements.
5                  break; // Leaving (jump to Line 16).
6              case v2:
7              .
8              .
9              .
10             case vk:
11                 // Conditional statements.
12                 break; // Leaving (jump to Line 16).
13             default:
14                 // Default statements.
15         }
16 ...
```

- The structure is convenient for finite and discrete conditions.
- The variable *target* must be a value of char, byte, short, int, or **String** type.
- The type of $v_1, \ldots,$ and v_k must be identical to *target*.
- A break statement may be needed to leave the construct.[2]
- The default case is used to perform default actions when none of cases matches *target*.
  - This acts like the else statements.

[2] If not, there will be a fall-through behavior.

Example

```
1  ...
2          // We define the traffic lights as follows:
3          // RED: 0
4          // YELLOW: 1
5          // GREEN: 2
6
7          int trafficLight = (int) (Math.random() * 3);
8
9          switch (trafficLight) {
10             case 0:
11                 System.out.println("Stop!!!");
12                 break;
13             case 1:
14                 System.out.println("Slow down!!");
15                 break;
16             case 2:
17                 System.out.println("Go!");
18         }
19 ...
```

Conditional Operators by Example

```
1 ...
2         if (num1 > num2)
3             max = num1;
4         else
5             max = num2;
6
7         // The above statement is equivalent to the following:
8         max = num1 > num2 ? num1 : num2;
9 ...
```

- If num1 $>$ num2, then do $\boxed{\text{max} = \text{num1}}$.
- Instead, do $\boxed{\text{max} = \text{num2}}$.

"We must all face the choice between what is right and what is easy."

– Prof. Albus Dumbledore,
Harry Potter and the Goblet of Fire, J.K. Rowling

"To be or not to be, that is the question."

– Prince Hamlet, *Hamlet*, William Shakespeare