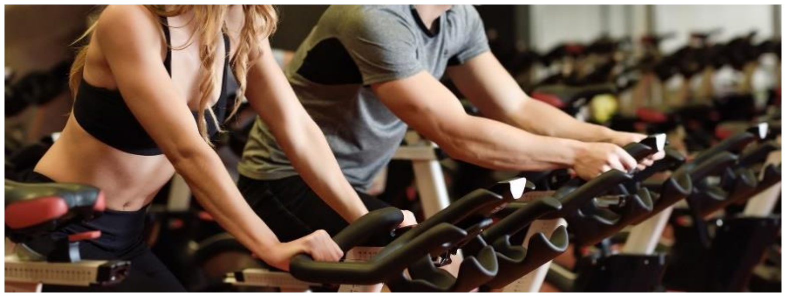
IECS273/274 [1111 3670/3671]
物件導向設計與實習

03#1 Java Fundamentals
Selection & Iterative Statement



01

Java Statement - Selection Statement (Conditional)

Java Statement

- In Java, flow of control (the program can decide which statements to be executed) refers to its branching and looping mechanisms.

⟨statement⟩ ::= ⟨assignment statement⟩ | ⟨sequential statement⟩ |
⟨selection statement⟩ | ⟨iterative statement⟩

➢ **branching mechanisms**
if-else, if, and switch statements

- Most branching and looping statements are controlled by Boolean expressions
  - ➢ A Boolean expression evaluates to either **true** or **false**
  - ➢ Primitive type of ***boolean*** only take the values **true** or **false**

Relational Java Operators

Java Operator	Means	Example (grade is 82)	Result
==	Equal to	grade == 80	False
>	Greater than	grade > 60	True
>=	Greater than or Equal to	grade >= 60	True
<	Less than	grade < 60	False
<=	Less thanor Equal to	grade <= 60	False
!=	Non Equal to	grade != 80	True

Truth Table of Expressions Operation

AND		
Exp1	Exp2	Exp1 && Exp2
True	True	True
True	False	False
False	True	False
False	False	False

NOT	
Exp1	! (Exp1)
True	False
False	True

OR		
Exp1	Exp2	Exp1 || Exp2
True	True	True
True	False	True
False	True	True
False	False	False

Selection_sample1.java

```java
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {

        // generate puzzle numbers
        int number1 = (int)(System.currentTimeMillis() % 10);
        int number2 = (int)(System.currentTimeMillis() / 10 % 10);

        // create a Scanner
        Scanner input = new Scanner(System.in);

        // show question
        System.out.print("What is " + number1 + " * " + number2 + " =? " );

        // get answer
        int answer = input.nextInt();

        // display result
        System.out.println(number1 + " * " + number2 + " = " +
        answer + " is " + (number1 * number2 == answer));
    }
}
```

Selection_sample1.class

```
What is 9 * 1 =? 9
9 * 1 = 9 is true

What is 9 * 6 =? 54
9 * 6 = 54 is true

What is 7 * 9 =? 62
7 * 9 = 62 is false
```

Selection Statement

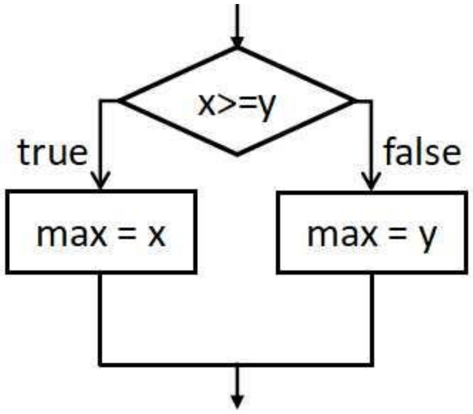
- **Selection/conditional statement**: According to the result of conditional judgment, select the relevant statement.
  - ✓ **if conditional statement,** syntax:

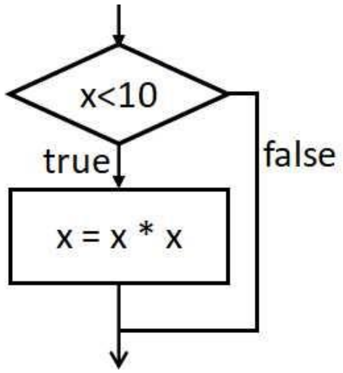
    ⟨if conditional statement⟩ ::= if (⟨condition⟩) {⟨statement⟩;}

    - If ⟨condition⟩ is evaluated to **true**, then execute ⟨statement⟩; otherwise, skip.
    - E.g., **if** (*x*<10) {*x* = *x* * *x*;}
  - ✓ **if-else conditional statement,** syntax:

    ⟨if-else conditional statement⟩ ::= if (⟨condition⟩) {⟨statement1⟩;} else {⟨statemen2⟩;}

    - If ⟨condition⟩ is evaluated to **true**, then execute ⟨statement1⟩; otherwise, skip execute ⟨statement2⟩;.
    - E.g., **if** (*x*>=*y*) {*max*=*x*;} **else {***max* = *y*;}



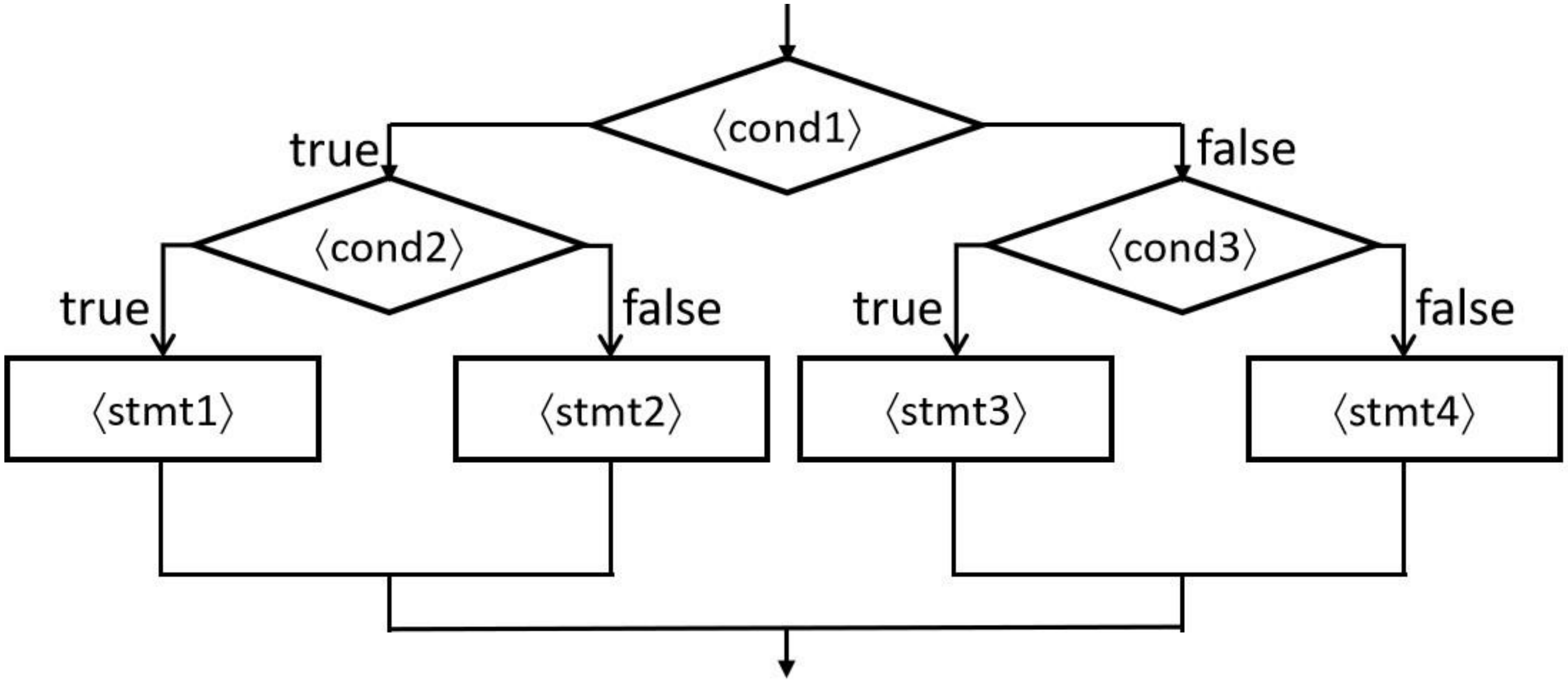


Selection Statement

✓ Complex conditional statement

- Ex. 1:

if (⟨cond1⟩) { **if** (⟨cond2⟩) {⟨stmt1⟩} **else** {⟨stmt2⟩} }
else { **if** (⟨cond3⟩) {⟨stmt3⟩} **else** {⟨stmt4⟩} }

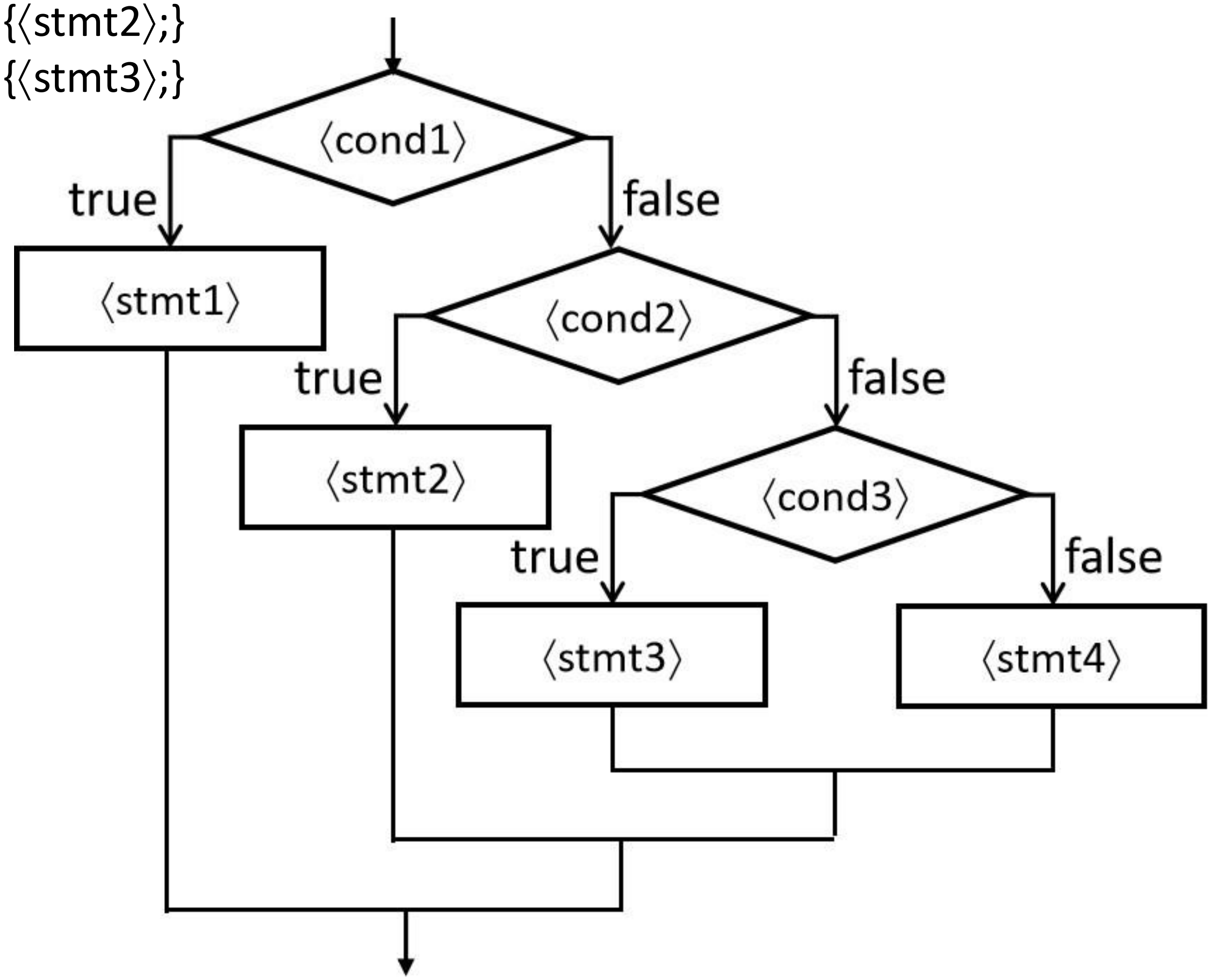


Selection Statement

✓ Complex conditional statement

- Ex. 2:

```
if (⟨cond1⟩)  { ⟨stmt1⟩;}
else if (⟨cond2⟩)  {⟨stmt2⟩;}
else if (⟨cond3⟩)  {⟨stmt3⟩;}
else {⟨stmt4⟩;}
```



Selection_sample2.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner keyboardInput = new Scanner(System.in);
        // Variable declare
        double radius, area;
        // Prompt the user to enter a radius
        System.out.print ("Enter a number for radius: ");
        radius = keyboardInput.nextDouble();

        if (radius < 0) {
                System.out.println(“Input Error");
        }
        else {
                // Compute area
                area = radius * radius * 3.14159;
                // Display results
                System.out.println("The area for the circle of
                                 radius " + radius + " is " + area);
        }
    }
}
```

Selection_sample2.class

```
Enter a number for radius: 4
The area for the circle of radius 4.0 is 50.26544

Enter a number for radius: -3
Input Error

Enter a number for radius: 16
The area for the circle of radius 16.0 is 804.24704
```

Selection Statement

- **Conditional Operator** (also called **arithmetic if**): The conditional operator is a notational variant on certain forms of the if-else statement.

```
if (n1 > n2)    max = n1;
else    max = n2;

max = (n1 > n2) ? n1 : n2;
```

- the expression to the right of the assignment operator is a conditional operator expression
- if the Boolean expression is true, then the expression evaluates to the value of the first expression

Selection_sample3.java

```java
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner keyboardInput = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print ("Enter two grade number for testing: ");
        int grade1 = keyboardInput.nextInt();
        int grade2 = keyboardInput.nextInt();

        System.out.println("Grade1 " +
                                (grade1 >= 60 ? "Passed." : "Failed." ));
        System.out.println("Grade2 " +
                                (grade2 >= 60 ? "Passed." : "Failed." ));
        }
}
```

Selection_sample3.class

```
Enter two grade number for testing: 30 80
Grade1 Failed.
Grade2 Passed.

Enter two grade number for testing: 70 50
Grade1 Passed.
Grade2 Failed.
```

Selection Statement

➢ **switch-case conditional statement,** syntax:

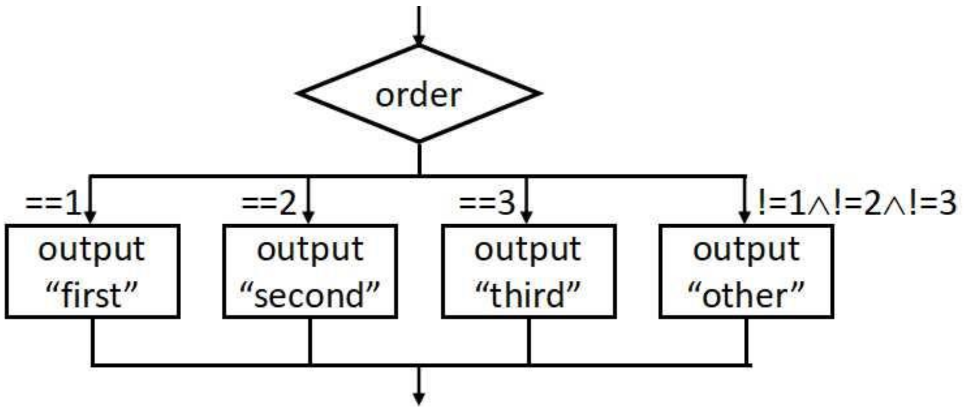
⟨switch-case conditional statement⟩ ::=

switch (⟨expression⟩) {case ⟨constant value⟩: ⟨statement⟩;} [default: ⟨statement⟩;]

✓ If ⟨expression⟩ is evaluated to a ⟨constant value⟩, execute the corresponding ⟨statement⟩; otherwise, execute the ⟨statement⟩ in **default** clause.

✓
```
switch (order) {
  case 1: { system.out.println("first");
            break; }
  case 2: { system.out.println("second");
            break; }
  case 3: { system.out.println("third");
            break; }
  default: { system.out.println("other"); }
}
```

The **break** statement stops the execution of the ⟨statement⟩ in the case; Without the **break** statement, program execution continues to the next ⟨statement⟩ after the ⟨switch-case conditional statement⟩ .



Selection_sample4.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner keyboardInput = new Scanner(System.in);
        // Prompt the user to enter a radius
        System.out.print ("Enter a number for switch statement test: ");
        int number = keyboardInput.nextInt();
        switch (number)
        {
        case 1:
                System.out.println("A");
                break;
        case 2:
                System.out.println("B");
                break;
        default:
                System.out.println("others");
                break;
        }
    }
}
```

Selection_sample4.class

```
Enter a number for switch statement test: 1
A

Enter a number for switch statement test: 2
B

Enter a number for switch statement test: 4
others
```

02

Java Statement - Iterative Statement

Java Statement

- In Java, flow of control (the program can decide which statements to be executed) refers to its branching and looping mechanisms.

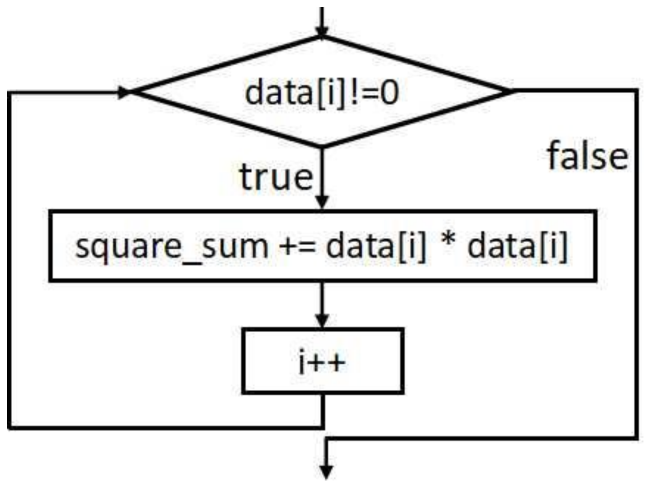
⟨statement⟩ ::= ⟨assignment statement⟩ | ⟨sequential statement⟩ |
⟨selection statement⟩ | ⟨iterative statement⟩

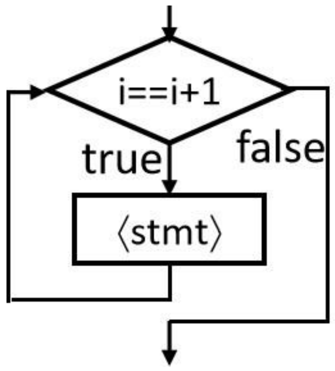
➢ **loop statements**
while, do while, and for statements

- A loop can be used to tell a program to execute statements repeatedly. It is similar to those in other high level programming languages.
  - ➢ he code that is repeated in a loop is called the body of the loop
  - ➢ each repetition of the loop body is called an iteration of the loop

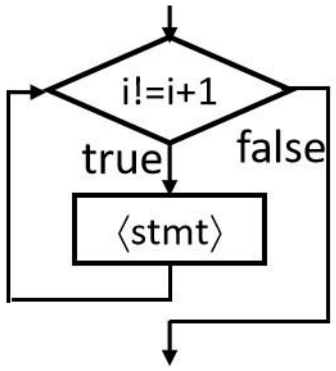
Iterative Statement

- Iterative/loop statement: According to the result of the loop condition, decide whether to repeat the statement in the loop.
  - ✓ **while loop statement,** syntax: **⟨while loop statement⟩ ::= while (⟨condition⟩) {⟨statement⟩;}**
    - If ⟨condition⟩ is evaluated to **true**, execute ⟨statement⟩, after completion of ⟨statement⟩, evaluate ⟨condition⟩ again; otherwise, the loop terminates.
    - The basic pattern of iterative/loop structure statement, all loop statements can be represented by **while** loop statement.
    - **while** (*data*[*i*]!=0) {*square_sum* += *data*[*i*] * *data*[*i*]; *i*++;}





empty loop



infinite loop

Iterative_sample1.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {

        // generate puzzle numbers
        int number1 = (int)(System.currentTimeMillis() % 10);
        int number2 = (int)(System.currentTimeMillis() / 10 % 10);

        // create a Scanner
        Scanner input = new Scanner(System.in);

        // show question
        System.out.print("What is " + number1 + " * " + number2 + " =? " );

        // get answer
        int answer = input.nextInt();

        while (number1 * number2 != answer) {
                System.out.print("Wrong answer. Try again. What is "
                + number1 + " * " + number2 + "? ");
                answer = input.nextInt();
        }
        System.out.println("You got it!");     }
}
```

Iterative_sample1.class

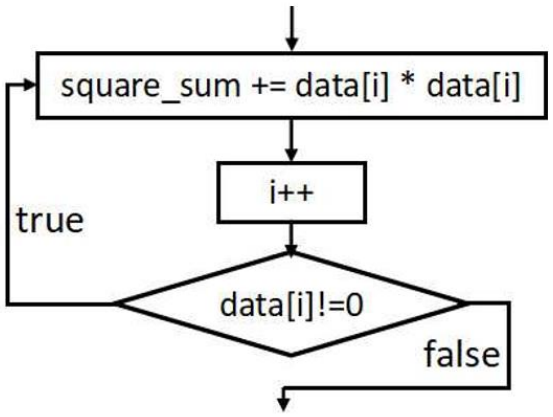
```
What is 5 * 7 =? 32
Wrong answer. Try again. What is 5 * 7? 36
Wrong answer. Try again. What is 5 * 7? 35
You got it!

What is 9 * 0 =? 0
You got it!

What is 9 * 8 =? 81
Wrong answer. Try again. What is 9 * 8? 72
You got it!
```

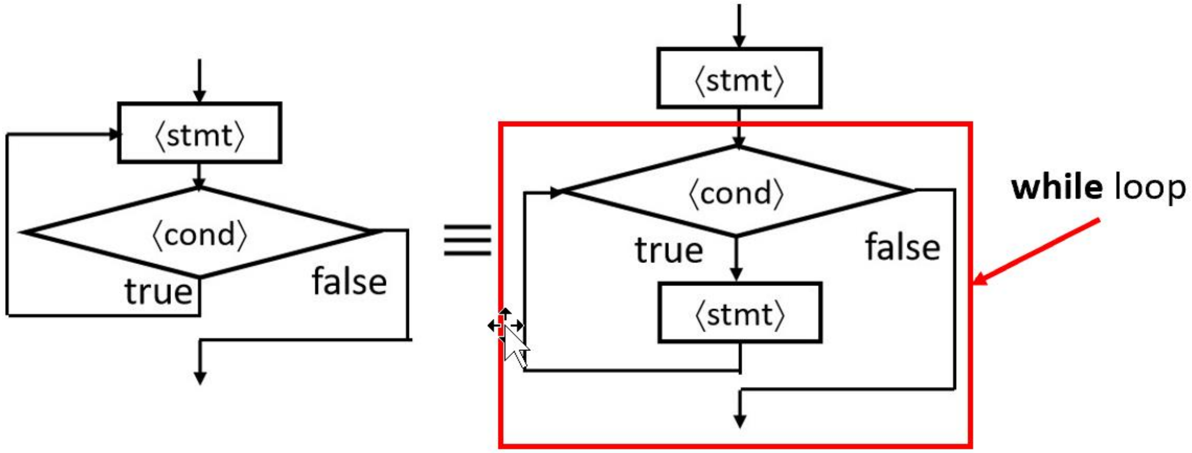
Iterative Statement

- **do-while loop statement,** syntax:

  ⟨do-while loop statement⟩ ::= do {⟨statement⟩;} while (⟨condition⟩);

  - ✓ Execute ⟨statement⟩ once. Then, if ⟨condition⟩ is evaluated to **true**,execute ⟨statement⟩, after completion of ⟨statement⟩, evaluate ⟨condition⟩ again; otherwise, the loop terminates.
  - ✓ **do-while** can be said is a **while** loop statment with execution of ⟨statement⟩ at least once.
  - ✓ **do** {*square_sum* += *data*[*i*] * *data*[*i*]; *i*++;} **while** (*data*[*i*]!=0);





Iterative_sample2.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {
        int x = 20;
        System.out.println("The original value of x is:" + x);
        while(x <= 18)
                x+=5;
        System.out.println("The while-do test value of x is:" + x);
        x=20;
        do
                x+=5;
        while (x<=18);
        System.out.println("The do-while test value of x is:" + x);

        System.out.println("The do-while iterative test of x is:");
        x = 20;
        do
        {
                x -= 2;
                System.out.println(x);
        } while(x > 10);
    }
}
```

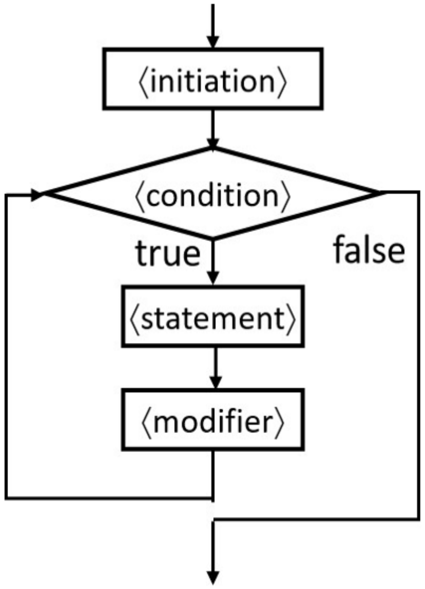
Iterative_sample2.class

```
The original value of x is:20
The while-do test value of x is:20
The do-while test value of x is:25
The do-while iterative test of x is:
18
16
14
12
10
```

Iterative Statement

- **for loop statement,** syntax:

  ⟨for loop statment⟩ ::= for (⟨initiation⟩; ⟨condition⟩; ⟨modifier⟩)
  {⟨statement⟩;}



✓ Execute ⟨initiaton⟩ to set the initial value of loop variable; evaluate ⟨condition⟩; if ⟨condition⟩ is **true**, then execute ⟨statement⟩; otherwise, the loop terminates; after completion of ⟨statement⟩, execute ⟨modifier⟩ and than evaluate ⟨condition⟩ again.

✓ In general, the number of iterations of the ⟨statement⟩ in while and do-while loop statements cannot be determined, so they are called indefinite loops; however, the number of iterations of ⟨statement⟩ in the for loop statement is determinable , so the for loop statement is also called a definite loop.

✓ **for** (*i*=0; *i*<100; *i*++) {*square_sum* += *data*[*i*] * *data*[*i*];}

- The above statement "{*square_sum* += *data*[*i*] * *data*[*i*];} " is executed 100 time.

✓ **for** (*i*=0; *i*<100; *i*++) {*square_sum* += *data*[*i*] * *data*[*i*]; i = *data*[*i*];}

Modify the loop variable so that the number of iteratons of ⟨statement⟩ cannot be determined.

- Variable *i* is called a **loop variable**. Do not update the value of loop variable in ⟨statement⟩.

Iterative_sample3.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {
        int oddSum = 0;
        int evenSum = 0;
        int total = 0;

        //loop through the numbers
        for(int i=1; i <= 10; i++)
        {
                total += i;

                if(i % 2 != 0) //odd number
                        oddSum += i;
                else
                        evenSum += i;
        }

        System.out.println(“Total = " + total);
        System.out.println("Odd sum = " + oddSum);
        System.out.println("Even sum = " + evenSum);
    }
}
```

Iterative_sample3.class

```
Total = 55
Odd sum = 25
Even sum = 30
```

Iterative Statement

- **Infinite Loops**：If the Boolean expression of *while*, *do-while*, or *for loop* remains true, then the loop will run forever, resulting in an infinite loop.

- **Nested Loops:** Loops can be nested. When nested, the inner loop iterates from beginning to end for each single iteration of the outer loop.

```
for (int row = 1; row < n ; row++){
        for (int column = 1; column < m; column++){
                System.out.print("<“ + row + "," + column + ">");
        }
        System.out.println();
}
```

Iterative_sample4.java

```
import java.io.*;
import java.util.Scanner; // Scanner is in the java.util package
public class test {
    public static void main(String[] args) {

        for (int row = 1; row <=6; row++){
                for (int column = 1; column <=3; column++){
                        System.out.print("<"+row + "," + column+"> ");
                }
        System.out.println();
        }

    }
}
```

Iterative_sample4.class

```
<1,1> <1,2> <1,3>
<2,1> <2,2> <2,3>
<3,1> <3,2> <3,3>
<4,1> <4,2> <4,3>
<5,1> <5,2> <5,3>
<6,1> <6,2> <6,3>
```

The *break* and *continue* Statements

- **break statement** ： It consists of the keyword *break* followed by a semicolon. When executed, *the break statement ends the nearest enclosing switch or loop statement*.
  - labeled break : If an enclosing loop statement is labeled with an *Identifier*, then the following version of the break statement will exit the labeled loop, even if it is not the innermost enclosing loop.

```
label1: for(int i=0;i<=n;i++){
                System.out.println("i="+i);
                for(int j=0;j<=n;j++){
                        System.out.println("j="+j);
                        break label1;
                }
        }
```

- **continue statement** ： It consists of the keyword *continue* followed by a semicolon. When executed, *the continue statement ends the current loop body iteration of the nearest enclosing loop statement*.
- **exit statement** ： It will immediately end the program as soon as it is invoked: *System.exit(0);*
  - The exit statement takes one integer argument
  - By tradition, a zero argument is used to indicate a normal ending of the program

03

Programming Practice

Practice 1： Draw Figures

① **Draw Triangle (三角形)**

Write a Java program that reads in a positive integer side from 3 to 29, and draws an isosceles triangle(等腰三角形) with side length side and base length 2*side-1. Output 10 blanks on the left side of the figure, use ‘&' to mark the sides of the isosceles triangle, and blanks to mark the interior points(內點)of the isosceles triangle.

② **Draw Square (正方形)**

Write a Java program that reads in a positive integer side from 2 to 30 and draws a square with each side of length side. Output 10 blanks on the left side of the figure, use “#*' to mark the sides of the square and blanks to mark the interior points of the square.

③ **Draw Rhombus (菱形)**

Write a Java program that reads in a positive integer side from 3 to 29 and draws a solid rhombus (diamond) with side of length side. Output 10 blanks on the left side of the figure, using ‘$' to mark the sides of the rhombus, and '@' to mark the interior points of the rhombus.

Practice 1： Draw Figures

```
Enter the side of an isosceles triangle (between 3 and 29): 6

               *
              * *
             *   *
            *     *
           *       *
          ***********

Enter the side of a square (between 2 and 30): 5

          *****
          *   *
          *   *
          *   *
          *****
Enter the side of a rhombus (between 3 and 29): 5

              *
             *@*
            *@@@*
           *@@@@@*
          *@@@@@@@*
           *@@@@@*
            *@@@*
             *@*
              *
```

Practice 2：Guess Number

- Write a Java program that randomly generates a positive integer from 1 to 1000 and guesses the value of this integer.
  If the guessed number is wrong, output a message to tell the player that the number he/she guessed is larger or smaller than the answer; if the guessed number is correct, tell the player that he/she guessed correctly and how many times he/she guessed.
  The number guessed in each step must be hinted to get closer and closer to the answer.

```
// Package for random number generator.
import java.util.Random;

// Initial a random number seed.
Random random = new Random();

// Generate the answer to be guessed.
answer = random.nextInt(1000) + 1 ;
```

Practice 2：Guess Number

```
>>> Enter an interger between 1 and 1000: 800
Your guess 800 is too large. Try again.

>>> Enter an interger between 1 and 1000: 600
Your guess 600 is too small. Try again.

>>> Enter an interger between 1 and 1000: 620
Your guess 620 is too large. Try again.

>>> Enter an interger between 1 and 1000: 616
Your guess 616 is too large. Try again.

>>> Enter an interger between 1 and 1000: 610
Congratulations! Your guess 610 is correct.

You guess 5 times.
```

04

▶ Assignment #1

Assignment #1: Draw Tree

- A Christmas tree can be designed as a multi-layer solid isosceles triangle, the uppermost triangle has the smallest side length, and the lower triangle side length is larger. For two adjacent triangles, the apex of the lower triangle will overlap the bottom of the triangle above it. Finally, a rectangular trunk will be immediately adjacent to the bottom of the lowermost triangle. Write a Java program to draw a Christmas tree as defined below.
  - ① A Christmas tree can be 2 to 5 tiers. (layer)
  - ② The side of the uppermost triangle can be 3 to 6 points long. (side)
  - ③ A lower triangle side may be 1 to 5 points larger than it upper one. (growth)
  - ④ The width of the trunk is an odd integer from 3 to 9. (width)
  - ⑤ The height of the trunk can be 4 to 9 points. (height)
- Use '#' to mark the points on the sides of the triangle, '@' to mark the points inside the triangle, and '|' to mark the points of the trunk.
- Write comments in your program solution. Also, write a report to explain how you develop your assignment solution.

Assignment #1: Draw Tree

```
Enter the number of tiers (2 to 5): 3
Enter the side of top triangle (3 to 6): 4
Enter the growth of two adjacent triangles (1 to 5): 2
Enter the width of the trunk (odd number, 3 to 9): 5
Enter the height of the trunk (4 to 10): 6

                 #
                #@#
               #@@@#
              #######
                #@#
               #@@@#
              #@@@@@#
             #@@@@@@@#
            ###########
                #@#
               #@@@#
              #@@@@@#
             #@@@@@@@#
            #@@@@@@@@@#
           #@@@@@@@@@@@#
          ###############
               |||||
               |||||
               |||||
               |||||
               |||||
               |||||
```