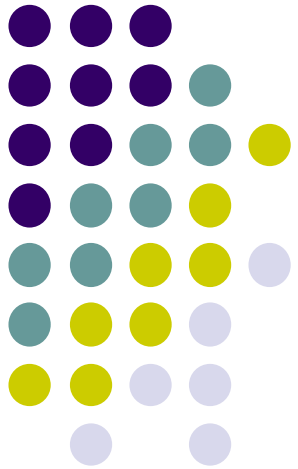
WIX1002
Fundamentals of Programming

Chapter 4

Flow of Control (Repetition)



Contents

- Introduction
- while
- do-while
- for
- break
- continue
- label
- Common Error

Introduction

- A repetition flow specifies that an action is to be repeated while some condition remains true.
- In Java, **while, do-while and for** statement are used for the repetition flow.
- There are two types of loop namely count-controlled loop and sentinel-controlled loop.
- **Count-controlled loop** executed the statements for a fixed number of times.
- **Sentinel-controlled loop** executed the statements repeatedly until the sentinel is encountered.

while

- A while statement executes a block of code repeatedly. A **condition** controls how often the loop is executed.

```
while (condition)
        statement;

// use brace {                          more than 1 statements
while (condition) {
        statement1;
        statement2;
        statement3;
}
```

while

```
int number=1, sum=0;
while (number<=10) {
   sum+=number;
   number++;
}

boolean status = true;
while(status) {
   number = k.nextInt();
   if (number < 0 )
      status = false;
}
```

do-while

- A do-while statement executes the body of the loop **at least once** and perform condition check after the body statements have been executed.

```
do
    statement;
while (condition);
```

```
// use brace {                          more than 1 statements
do {
    statement1;
    statement2;
} while (condition);
```

do-while

```
int number=1, sum=0;
do {
   sum+=number;
   number++;
} while (number<=10);

boolean status = true;
do {
   number = k.nextInt();
   if (number > 0 )
      status = false;
} while(status);
```

for

- A for statement is suitable for **count-controlled** loops. It is used to step through some integer variable in equal increments or decrements

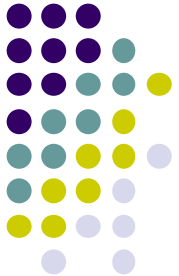
```
for (initialization; condition; update)
        statement;

// use brace {                          more than 1 statements
for (initialization; condition; update) {
        statement1;
        statement2;
        statement3;
}
```

for

```
for (int num = 1; num <= 5; num++)
   System.out.println("Counter is " + num);

for (int i=10; i>0; i--) {
   sum +=i;
   counter++;
}
```



break

- A break statement **ends the nearest enclosing loop** statement

```
for (count = 1; count <= 10; count++) {
   if ( count == 5 )
      break;  // break the loop when count is equal to 5
   sum += count;
}
```

continue

- A continue statement **ends the current loop body iteration** of the nearest enclosing loop statement and proceeds with the next iteration of the loop

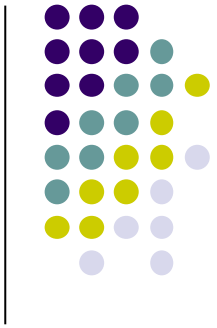
```
for (int count = 1; count <= 10; count++ ) {
   if ( count == 5 )
      continue;  // skip remaining statement in the loop
                 // when count = 5
   sum+=count;
}
```

label

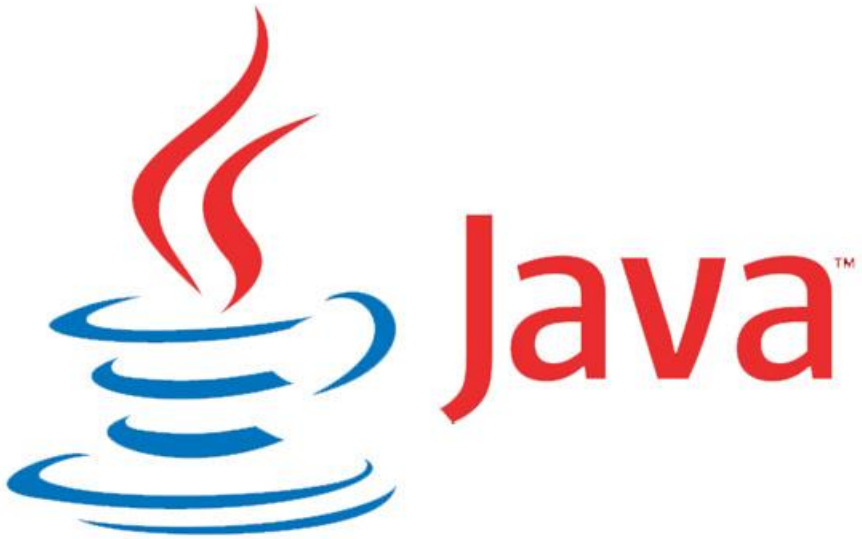
- A label statement is used to **label a loop statement.** The label statement can be used by the break statement and the continue statement

```
stop: {   // label statement
for (int row = 1; row <= 10; row++) {
   for (int column = 1;column <= 5; column++) {
      if ( row == 5 )
         break stop; // break the stop label statement
      counter++;
   }
}
```

Common Error



- **An off by one error**
  - The loop iterates once too often or once too few times. Check the condition and / or the initial value of the counter.
- **An infinite counting loop**
  - The counter is going the wrong way or doesn't change at all. Make sure the counter change correctly in loop
- **An infinite sentinel loop**
  - The new data are not input at the end of the loop body


Java™