Java Programming AP Edition

U2C5 Loops

WHILE LOOP

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

Motivation

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a **hundred times**:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?

Motivation

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
…
…
…
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

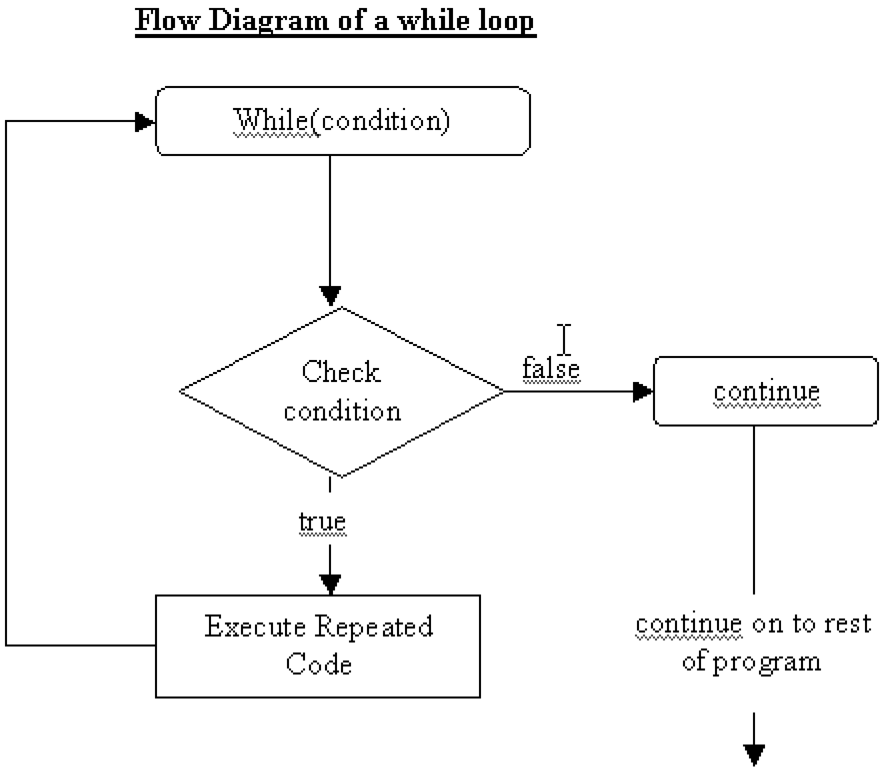
Solution to it: while-loop

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java");
  count++;
}
```

while loop

The syntax for the while loop is:

```
while (loop-continuation-condition) {
   // loop body
   Statement(s);
}
```



Comparison of if-statement and while-loop

```
int x = 0;                  // if-statement
if (x < 10) {
    System.out.println(“Welcome to Java.”);
}

int x = 0;                  // while-loop
while (x < 10) {
    System.out.println(“Welcome to Java.”);
    x++;
}
```

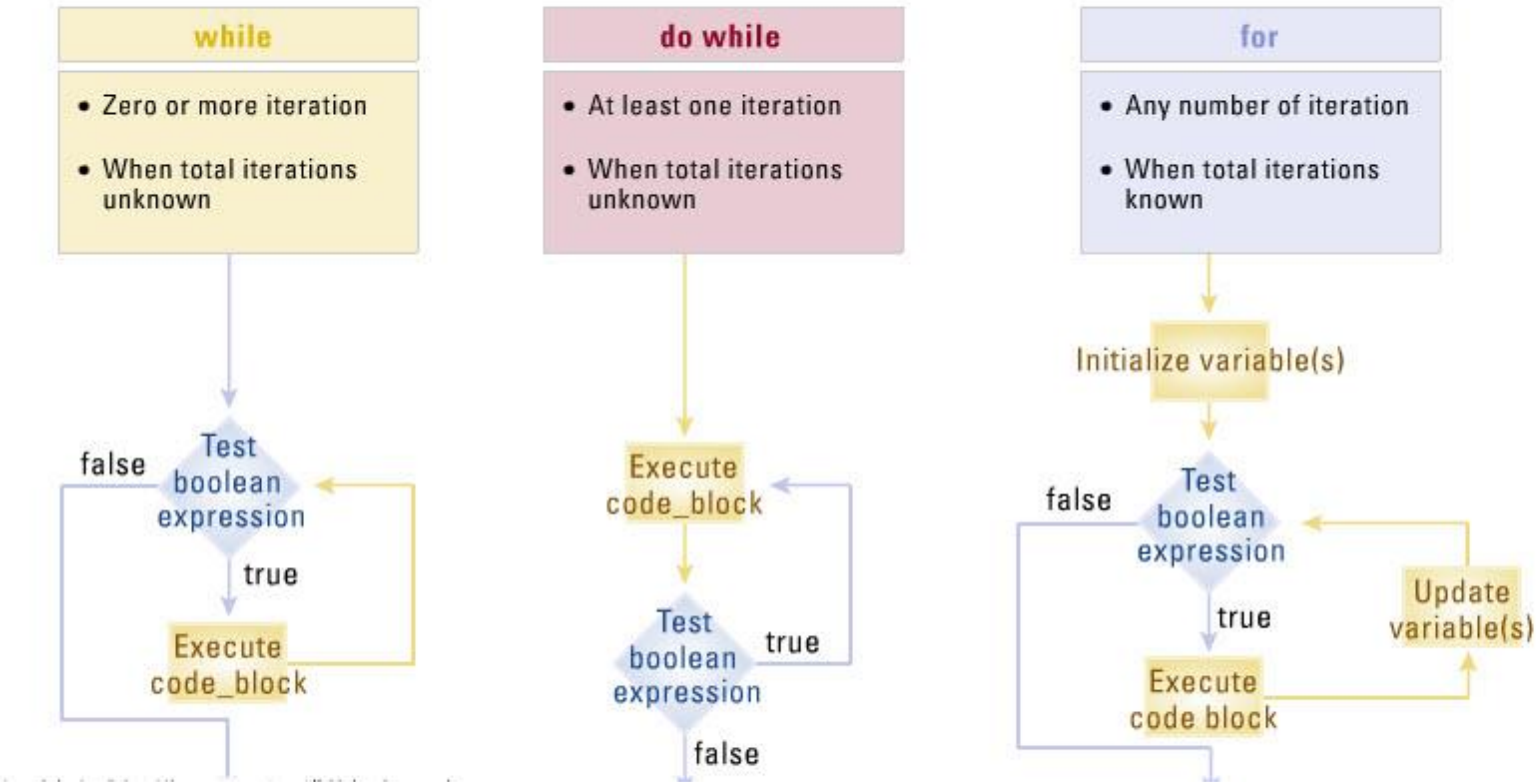
LOOP Structures Supported By Java

Loops:

- for-loop (later lecture)
- while-loop
- do-while-loop
- for-each-loop (chapter 7/8)

Loop Breaks: (later in other lecture)

- {} /* empty braces as pass function */
- Continue /* skip the rest of iteration */
- Break /* skip the rest of loop */
- Return /* skip the rest of function */
- System.exit(0); /* skip the rest of program */



while

- Zero or more iteration
- When total iterations unknown

do while

- At least one iteration
- When total iterations unknown

for

- Any number of iteration
- When total iterations known

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Initialize count

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is true

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Increase count by 1
count is 1 now

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is still true since count is 1

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Increase count by 1
count is 2 now

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is false since count is 2 now

Trace while-loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

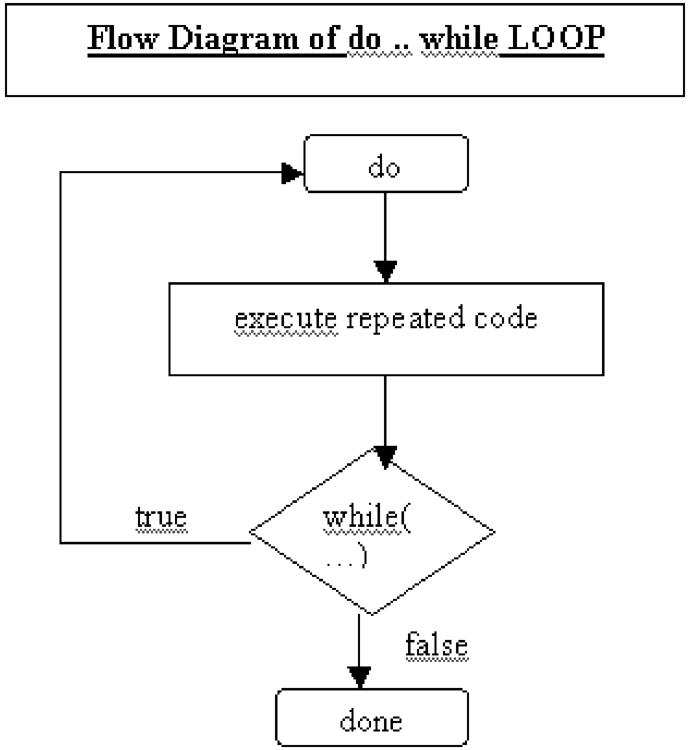
The loop exits. Execute the next statement after the loop.

do … while loop

The do-while loop is a variation of the while loop.

Its syntax is:

```
do {
   // loop body;
   statement(s);
} while (loop-continuation-condition);
```



Flow Diagram of do .. while LOOP

Difference between do-while-loop and while-loop

The difference between a while loop and a do-while-loop is the order in which the loop-continuation-condition is evaluated and the loop body executed. You can write a loop using either the while-loop or the do-while loop. Sometimes one is a more convenient choice than the other.

Menu.java (later lecture)