Sukkur IBA

Dr. Faheem Akhtar Rajput

Object Oriented Programming (JAVA)

Lecture 12

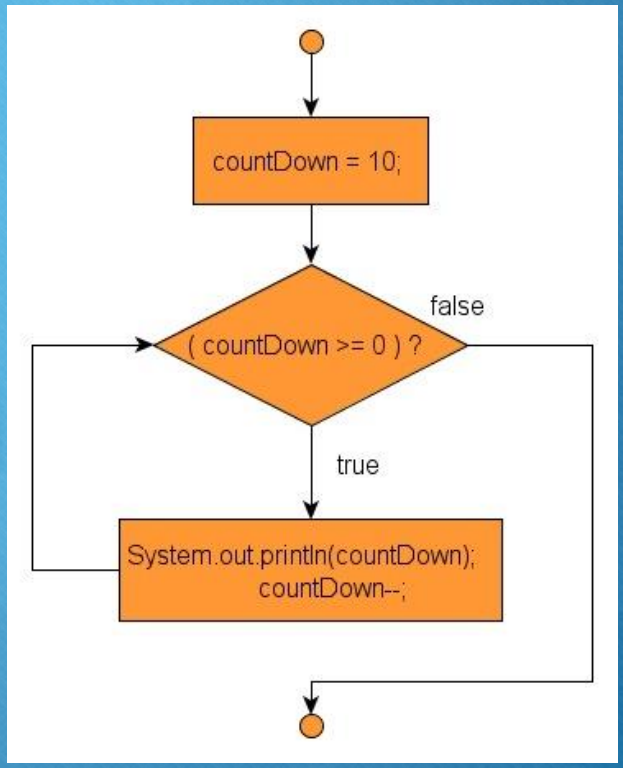
Iteration Statements

- Java's iteration statements are:
  - For
  - While
  - Do-While

While (1/2)

- The while loop is Java's most fundamental loop statement
- It repeats a statement or block while its controlling expression is true.
- The condition can be any Boolean expression.

```
while(condition) {
   // body of loop
}
```



While (2/2)

```
// Demonstrate the while loop.
class While {
  public static void main(String args[]) {
    int n = 10;

    while(n > 0) {
      System.out.println("tick " + n);
      n--;
    }
  }
}
```

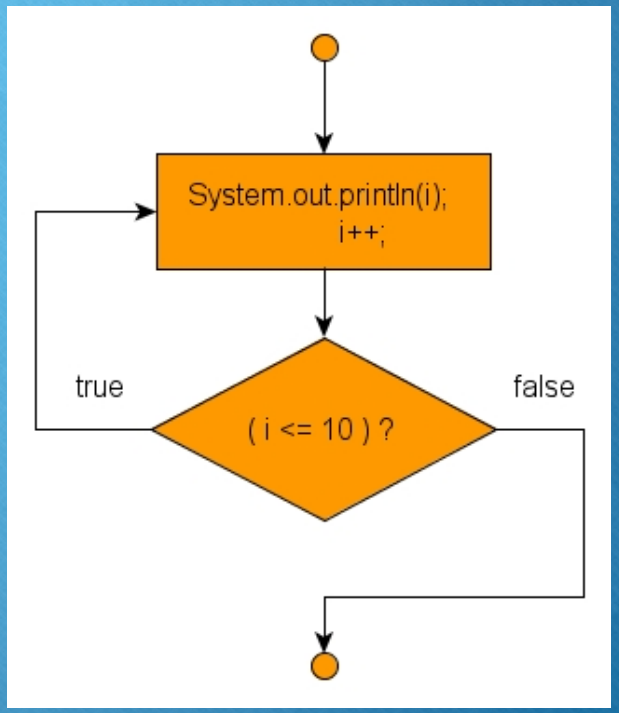
```
// The target of a loop can be empty.
class NoBody {
  public static void main(String args[]) {
    int i, j;

    i = 100;
    j = 200;

    // find midpoint between i and j
    while(++i < --j) ; // no body in this loop

    System.out.println("_____ is " + i);
  }
}
```

OUTPUT
Midpoint is 150

do-while [1/2]

- Sometimes it is desirable to execute the body of a loop at least once
- This can be achieved because its conditional expression is at the bottom of the loop.

```
do {
   // body of loop
} while (condition);
```



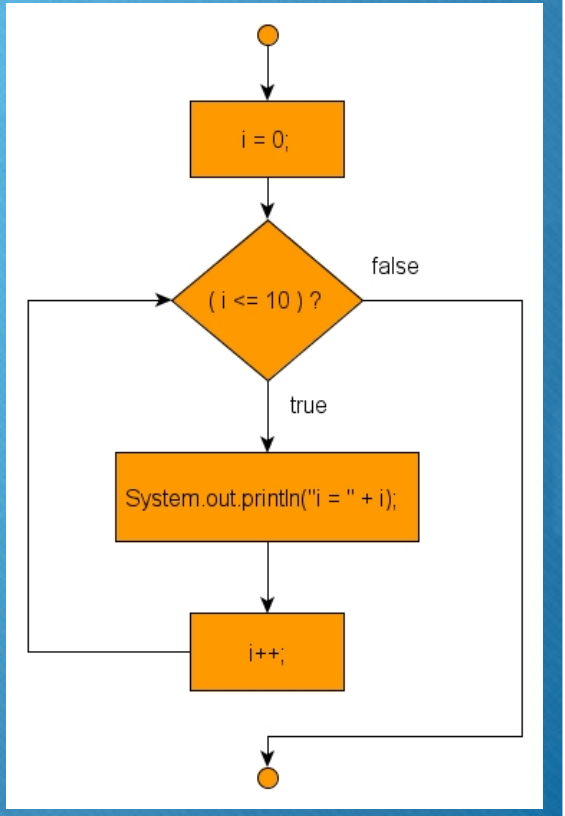
do-while [2/2]

```
class doWhile {
   public static void main(String[] args){
      int count = 11;
      do {
         System.out.println("Count is: " + count);
         count++;
      }  while (count < 11);
   }
}
```

for

- There are two forms of for loop in java:
  - Traditional for loop
  - for-each

```
for(initialization; condition; iteration) {
        // body
}
```



for

- Declaration of controls is possible inside the loop

for(int n=10; n>0; n--)

- multiple statements in both the initialization and iteration portions of the for.

for(a=1, b=4; a<b; a++, b--)

for – Some variations

```
boolean done = false;
for(int i=1; !done; i++) {
        // ...
       if(interrupted()) done = true;
}
```

```
for( ; ; ) {
    // ... Body of loop
}
```

```
// Parts of the for loop can be empty.
class Practice {
  public static void main(String args[]) {
    int i;
    boolean done = false;

    i = 0;
    for( ; !done; ) {
      System.out.println("i is " + i);
      if(i == 10) done = true;
      i++;
    }
  }
}
```

for-each

- designed to cycle through a collection of objects, it is useful for Arrays mostly (in strictly sequential fashion)
- Added after jdk1.5

for(type itr-var : collection) statement-block

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int sum = 0;
for(int i=0; i < 10; i++) sum += nums[i];
```

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int sum = 0;
for(int x: nums) sum += x;
```

for-each

```
// Use a for-each style for loop.
class Practice {
  public static void main(String args[]) {
    int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    int sum = 0;

    // use for-each style for to display and sum the values
    for(int x : nums) {
      System.out.println("Value is: " + x);
      sum += x;
    }

    System.out.println("Summation: " + sum);
  }
}
```

```
Value is: 1
Value is: 2
Value is: 3
Value is: 4
Value is: 5
Value is: 6
Value is: 7
Value is: 8
Value is: 9
Value is: 10
Summation: 55
```

for-each with break

```
// Use break with a for-each style for.
class ForEach2 {
  public static void main(String args[]) {
    int sum = 0;
    int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

    // use for to display and sum the values
    for(int x : nums) {
      System.out.println("Value is: " + x);
      sum += x;
      if(x == 5) break; // stop the loop when 5 is obtained
    }
    System.out.println("Summation of first 5 elements: " + sum);
  }
}
```

```
Value is: 1
Value is: 2
Value is: 3
Value is: 4
Value is: 5
Summation of first 5 elements: 15
```

Nested Loops

- Like all other programming languages, Java allows loops to be nested.

```
// Loops may be nested.
class Nested {
  public static void main(String args[]) {
    int i, j;

    for(i=0; i<10; i++) {
      for(j=i; j<10; j++)
        System.out.print(".");
      System.out.println();
    }
  }
}
```

```
..........
.........
........
.......
......
.....
....
...
..
.
```

Jump Statements [1/4]

- Using break
- Using break to Exit a Loop

```
// Using break to exit a loop.
class BreakLoop {
  public static void main(String args[]) {
    for(int i=0; i<100; i++) {
      if(i == 10) break; // terminate loop if i is 10
      System.out.println("i: " + i);
    }
    System.out.println("Loop complete.");
  }
}
```

OUTPUT

```
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9 Loop complete.
```

Jump Statements [2/4]

- Using break as a Form of Goto
- break label;

```
// Using break as a civilized form of goto.
class Break {
  public static void main(String args[]) {
    boolean t = true;

    first: {
      second: {
        third: {
          System.out.println("Before the break.");
          if(t) break second; // break out of second block
          System.out.println("This won't execute");
        }
        System.out.println("This won't execute");
      }
      System.out.println("This is after second block.");
    }
  }
}
```

OUTPUT
Before the break.
This is after second block.

Jump Statements [3/4]

- Using continue

```
listing 27
// Demonstrate continue.
class Continue {
  public static void main(String args[]) {
    for(int i=0; i<10; i++) {
      System.out.print(i + " ");
      if (i%2 == 0) continue;
      System.out.println("");
    }
  }
}
```

OUTPUT

```
0 1
2 3
4 5
6 7
8 9
```

Jump Statements [4/4]

- The **return** statement is used to explicitly return from a method.

```
// Demonstrate return.
class Return {
  public static void main(String args[]) {
    boolean t = true;

    System.out.println("Before the return.");

    if(t) return; // return to caller

    System.out.println("This won't execute.");
  }
}
```

OUTPUT
Before the return.

Week Review

- Arithmetic, Bitwise, Relational, Logical & Operators
- Three Control Statement
  - Selection (IF, SWITCH)
  - Iteration (while, do-while, for and for-each)
  - Jump (break, continue, return)

Questions?

Sukkur IBA

Thanks…