CS 180 Recitation

Repetition Statements
Feb 17, 2012
Repetition/Iteration Statements

• Used to create loops in code.
• Loop is a set of instructions that are executed repeatedly until a termination condition is met.
  • while statement
  • do-while statement
  • for statement
while statement

• General form

    while ( Condition ) {
        // body of loop
    }

• Condition is any Boolean expression.

• The “body of loop” will be executed as long as the Condition is true.

• When Condition becomes false, the control passes to the statement immediately after the while statement.
public class WhileExample1
{
    public static void main(String[] args) {
        int x = 1;
        System.out.println("Before Loop");
        while (x < 4) {
            System.out.println(x);
            x++;
        }
        System.out.println("After Loop");
    }
}
while – Example 1 cont’d.

Output:

Before Loop
1
2
3

After Loop
public class WhileExample2
{
    public static void main(String[] args) {
        int x = 1;
        System.out.println("Before Loop");
        while (x < 1) {
            System.out.println(x);
            System.out.println(x);
            x++;
        }
        System.out.println("After Loop");
    }
}

while Example2 cont’d.

Output:
Before Loop
After Loop

• None of the statements in body of loop were executed.

• Body of loop can also be empty.
do-while statement

- General form
  
  ```
  do {
      // body of loop
  } while (Condition);
  ```

- Condition and “body of loop” are the same as in while statement.

- Unlike while statement, do-while statement can guarantee the body of loop is executed at least once.

- while loops check for termination condition at the beginning of loop.

- do-while loops check for termination condition at the end of loop (makes sure body of loop is executed at least once).
do-while Example3

public class DoWhileExample3
{
    public static void main(String[] args) {
        int x = 1;
        System.out.println("Before Loop");
        do {
            System.out.println(x);
            x++;
        } while (x < 1);
        System.out.println("After Loop");
    }
}
do-while Example3 cont’d.

Output:
Before Loop
1
After Loop

• Loop was executed once even though the termination condition was always true

• Useful in handling loops which are sure to be executed at least once (handling user input)
for statement

• General form

```for (Initialization; Condition ; Iteration) {
    // body of loop
    }
```

• Initialization – Used to declare and initialize a variable to use within the body of loop.

• Condition – Used to test for the termination condition. Condition must resolve to some boolean value.

• Iteration – Statements to be executed at the end of each loop.
for statement con’t.

- When loop first starts, Initialization is executed. Execution of Initialization part happens only once.
- Next, Condition is evaluated. If Condition is true the body of loop gets executed, else the loop terminates immediately.
- If condition was true in the previous step, the body of loop is executed and the Iteration statements are executed once at the end of loop.
- Control goes back to Condition evaluation, and the process continues until Condition becomes false.
public class ForExample4
{
    public static void main(String[] args) {
        for (int x = 1; x <= 10; x++) {
            if (x%2 == 0) {
                System.out.println(x);
            }
        }
    }
}

for – Example4 cont’d.

- Output
  2
  4
  6
  8
  10

- Note: Scope of variable 'x' is inside the for loop only.
for loops with comma

- By default only one statement can be present in a for loop Initialization and Iteration section.
- Useful when for loop execution depends on interaction between multiple variables instead of just one.
- Example follows..
for loop with comma - Example5

public class ForCommaExample5
{
    public static void main(String[] args) {
        for (int x=1, y=1; x <= 5; x++, y+=x)
            System.out.println("Sum so far = " + y);
    }
}

Output:
Sum so far = 1
Sum so far = 3
Sum so far = 6
Sum so far = 10
Sum so far = 15
Nested Loops - Example6

```java
public class NestedLoopExample6 {
    public static void main(String[] args) {
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 2; j++) {
                System.out.println("i = " + i + " j = " + j);
            }
        }
    }
}
```
Nested Loops – Example6 cont’d.

• Output
  i = 0 j = 0
  i = 0 j = 1
  i = 1 j = 0
  i = 1 j = 1
  i = 2 j = 0
  i = 2 j = 1

• Inner Loop is executed 6 times.
• Outer Loop is executed 3 times.