Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Loop
2. While Loop
3. Do-While Loop
4. For Loop
5. Nested Loop

Loop

- Loop can execute a set lines or statements as long as the specified condition is satisfied.
- It mainly minimizes lines of code.
- There are mainly two types of loops:
  - **Entry control loop**: This type of loop checks the condition at the time of entry, and if the condition is true, control transfers into the body of the loop. Example: for Loop and while Loop.
  - **Exit control loop**: An exit control loop is a loop in which the loop body is executed first, and then, the given condition is checked afterwards. Example: do-while loop.

While Loop

- In Java, while loop is used to iterate a part of the program several times.
- Here, condition is evaluated first, and if it returns true, the statements or lines inside while loop are execute.
- If the number of iteration is not fixed, it is recommended to use while loop.
- Syntax:

```
while (condition)
{
        statement 1;
        statement 2;
}
```

While Loop (Cont…)

- In while loop, it is not compulsory to use increment or decrement statement inside the body of the loop. However, at some point, condition must return false.
- A while loop evaluates the condition inside the parenthesis ().
- If the condition is true, the code/statement inside the while loop is executed.
- The condition is evaluated again.
- This process continues until the condition is false.
- When the condition is false, the loop stops.

While Loop (Cont…)

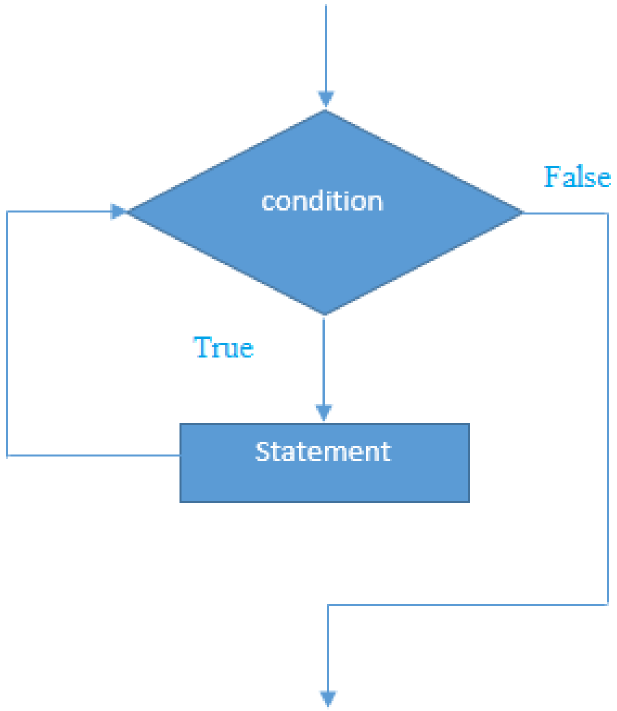
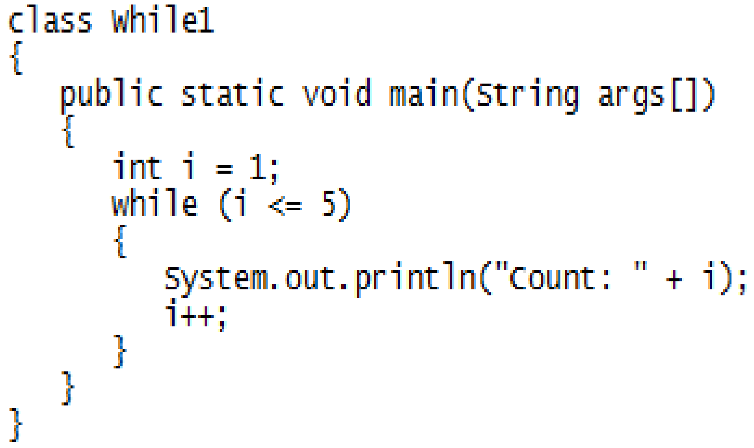


Fig. 1: Flowchart of while loop

While Loop (Cont…)



```
class While1
{
   public static void main(String args[])
   {
      int i = 1;
      while (i <= 5)
      {
         System.out.println("Count: " + i);
         i++;
      }
   }
}
```

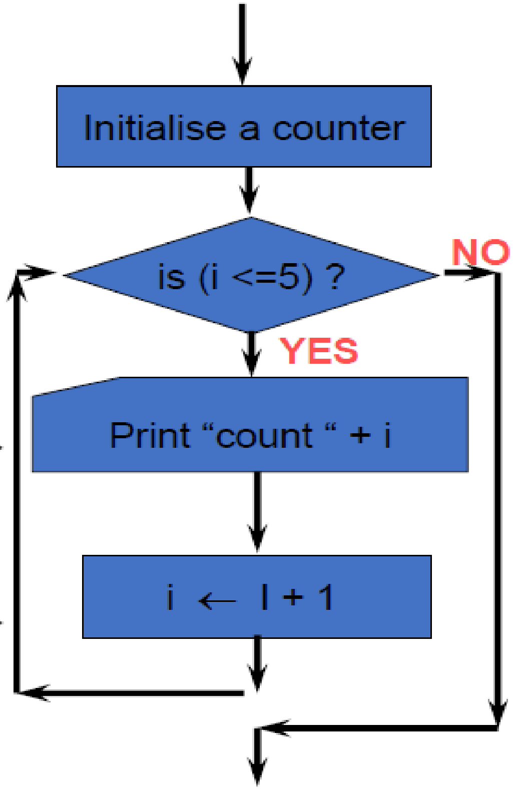
While Loop (Cont…)

- Output

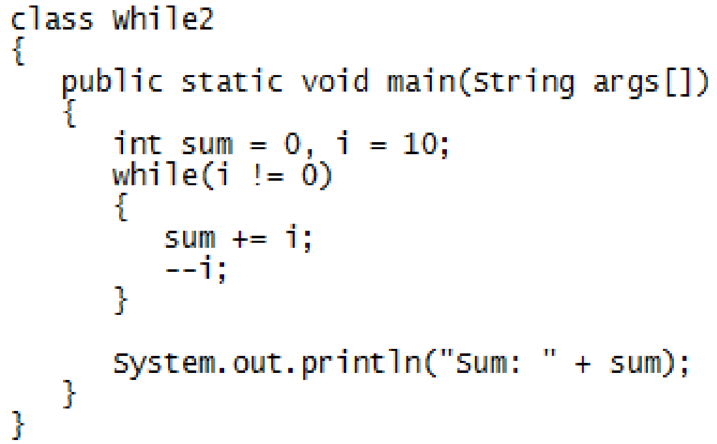
```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

While Loop (Cont…)

```
int i = 1;

while (i <= 5)
{

System.out.println ("count : " + i);


   i++;
}
```



While Loop (Cont…)



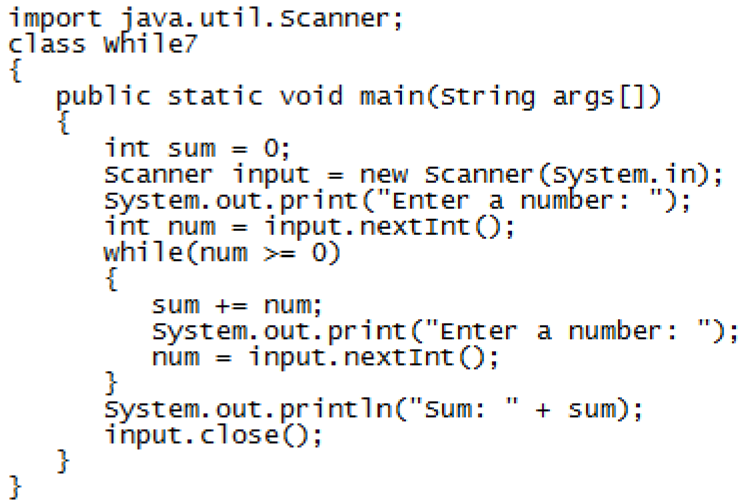
```
class while2
{
   public static void main(String args[])
   {
      int sum = 0, i = 10;
      while(i != 0)
      {
         sum += i;
         --i;
      }

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

While Loop (Cont…)

- Output

```
Sum: 55
```

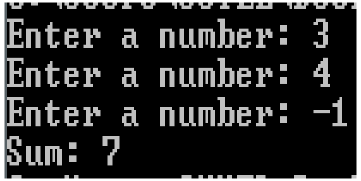
While Loop (Cont…)



```
import java.util.Scanner;
class While7
{
   public static void main(String args[])
   {
      int sum = 0;
      Scanner input = new Scanner(System.in);
      System.out.print("Enter a number: ");
      int num = input.nextInt();
      while(num >= 0)
      {
         sum += num;
         System.out.print("Enter a number: ");
         num = input.nextInt();
      }
      System.out.println("Sum: " + sum);
      input.close();
   }
}
```

While Loop (Cont…)

- Output



```
Enter a number: 3
Enter a number: 4
Enter a number: -1
Sum: 7
```

Do-While Loop

- Do-while loop is similar to while loop.
- This loop executes the lines or statements once before checking the condition. If the condition is true, it repeats the loop as long as the given condition is true.
- We can use do-while loop, if the number of iteration is not fixed and we must have to execute the loop at least once.
- Syntax:

```
do
{
    statement 1;
    statement 2;
}while (condition);
```

Do-While Loop (Cont…)

- The body of the do-while loop is executed at first. Then, the condition is evaluated.
- If the condition is true, the body of the loop inside the “do statement” is executed again.
- The condition is evaluated once again. If the condition is true, the body of the loop inside the “do statement” is executed again.
- This process continues until the condition is false and the loop gets stop.

Do-While Loop (Cont…)

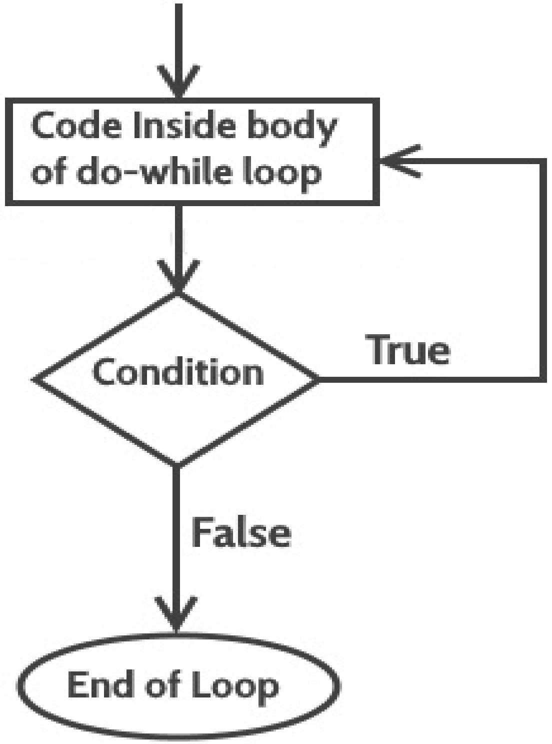
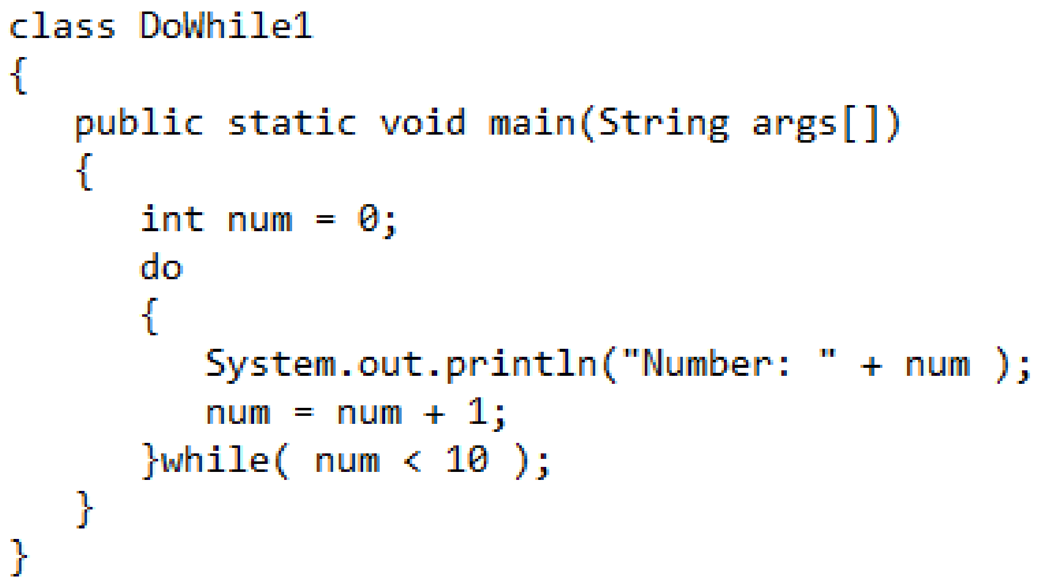


Fig. 2: Flowchart of do-while loop

Do-While Loop (Cont…)



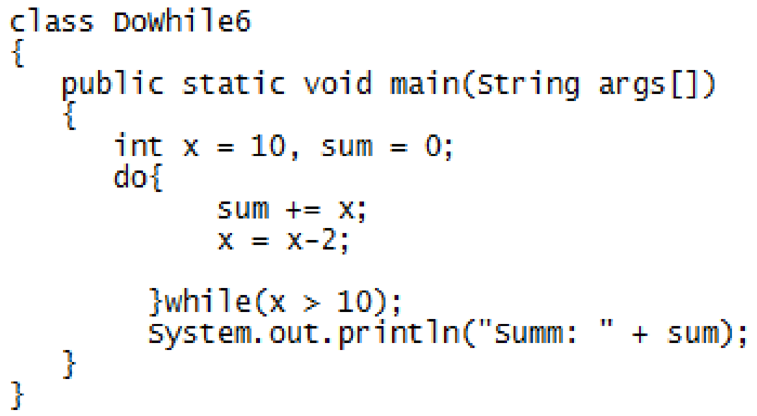
```
class DoWhile1
{
    public static void main(String args[])
    {
        int num = 0;
        do
        {
            System.out.println("Number: " + num );
            num = num + 1;
        }while( num < 10 );
    }
}
```

Do-While Loop (Cont…)

- Output

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

Do-While Loop (Cont…)



```
class DoWhile6
{
   public static void main(String args[])
   {
      int x = 10, sum = 0;
      do{
            sum += x;
            x = x-2;

        }while(x > 10);
        System.out.println("Summ: " + sum);
   }
}
```

Do-While Loop (Cont…)

- Output

```
Sum: 10
```

For Loop

- For loop is concise version of while loop.
- For loop is used, when we know exactly for how many times the code block will be executed.
- Syntax:

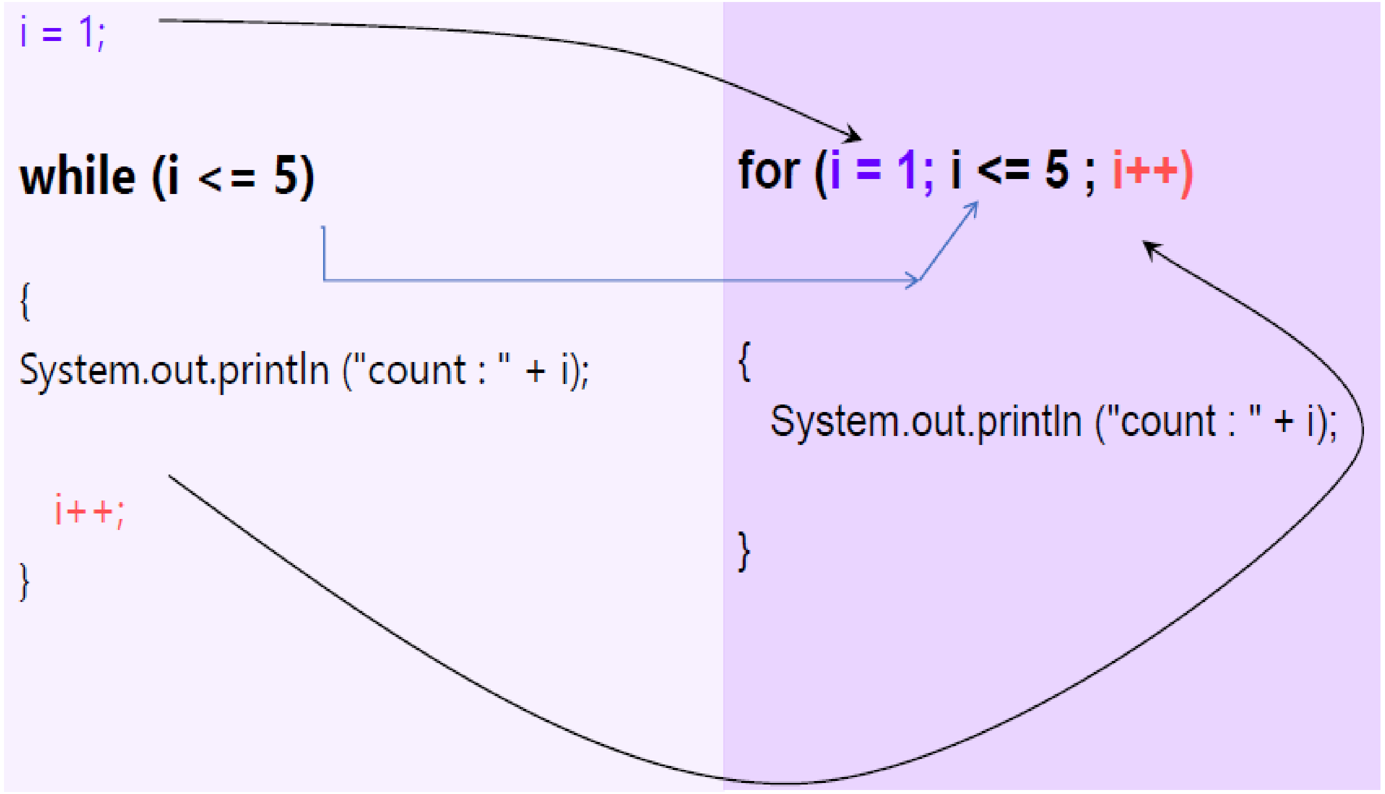
```
for(Condition 1; Condition 2; Condition 3)
{
   statement 1;
}
```

Condition 1 is executed one time before the execution of the code block/statement. **Condition 2** defines the condition for executing the code block. **Condition 3** is executed every time after the code block/statement has been executed.

For Loop (Cont…)

- There are mainly four parts in the entire for loop:
  - **Initialization:** It is the initial condition, which is executed once when the loop starts. Here, we can initialize the variable or we can use an already initialized variable. It is an optional condition.
  - **Condition:** It is the second condition, which is executed each time to evaluate the condition of the loop. It continues execution until the condition is false. It must return a Boolean value, i.e., either true or false. It is an optional condition.
  - **Statement:** The statement of the loop is executed each time until the second condition is false.
  - **Increment/Decrement:** It increments or decrements the variable value. It is also an optional condition.

For Loop (Cont…)



```
i = 1;

while (i <= 5)

{
System.out.println ("count : " + i);

   i++;
}
```

```
for (i = 1; i <= 5 ; i++)

{
   System.out.println ("count : " + i);

}
```

For Loop (Cont…)

- There is another way to write a for loop for an array (traverse).
- It works on elements basis, not index basis.
- Values of the array cannot be changed.
- Syntax:

```
for (Data_Type Variable_Name: Array_Name)
{
    Statement 1;
    Statement 2;
 }
```

For Loop (Cont…)

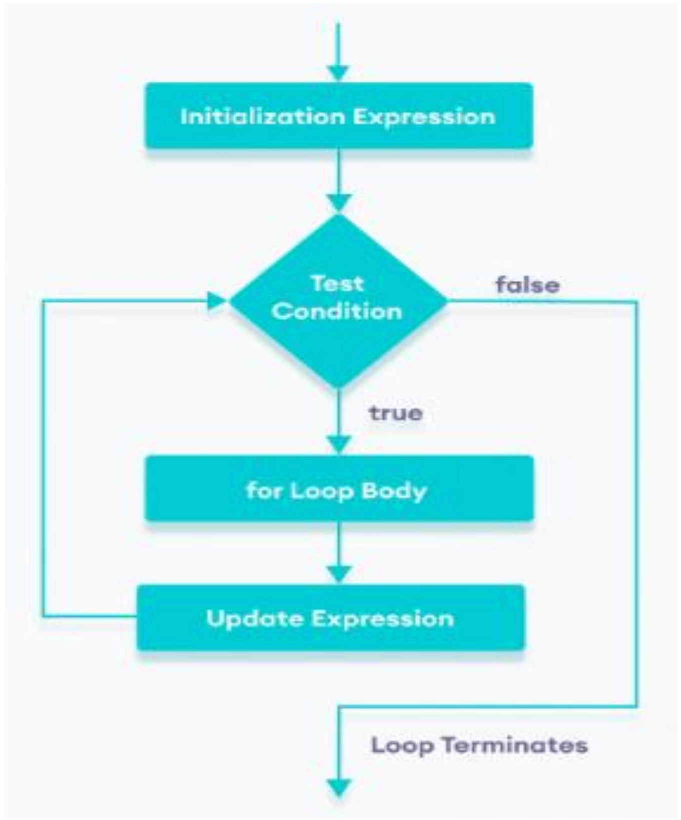
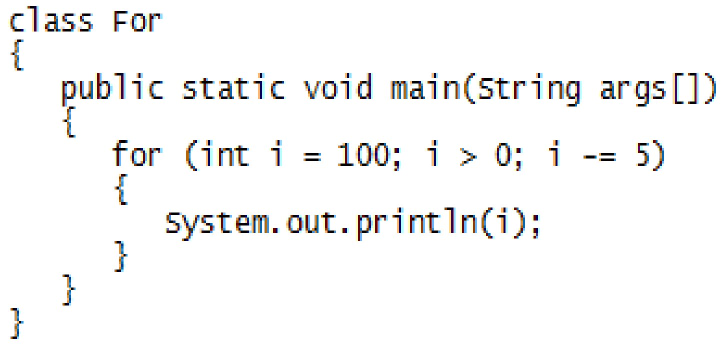


Fig. 3: Flowchart of for loop

For Loop (Cont…)



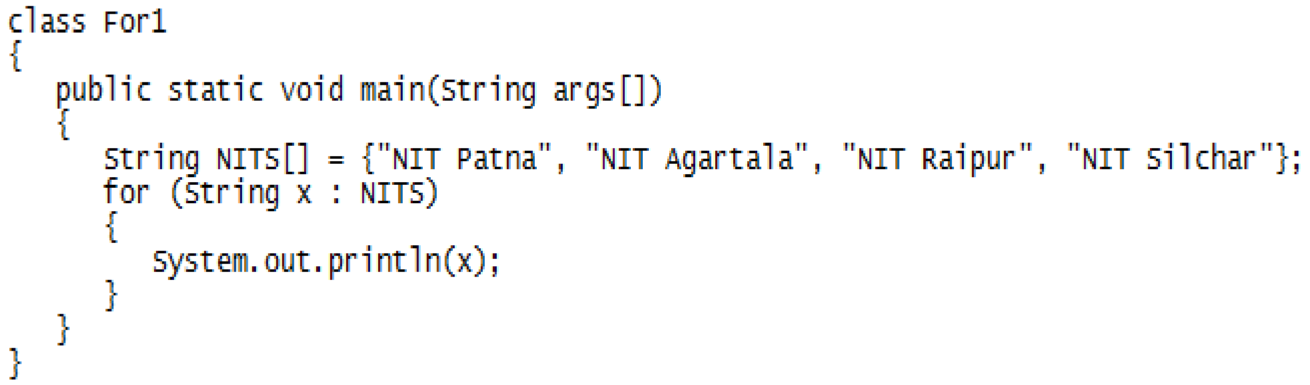
```
class For
{
   public static void main(String args[])
   {
      for (int i = 100; i > 0; i -= 5)
      {
         System.out.println(i);
      }
   }
}
```

For Loop (Cont…)

- Output

```
100
 95
 90
 .
 .
 .
 5
```

For Loop (Cont…)



```
class For1
{
   public static void main(String args[])
   {
      String NITS[] = {"NIT Patna", "NIT Agartala", "NIT Raipur", "NIT Silchar"};
      for (String x : NITS)
      {
         System.out.println(x);
      }
   }
}
```

For Loop (Cont…)

- Output

```
NIT Patna
NIT Agartala
NIT Raipur
NIT Silchar
```

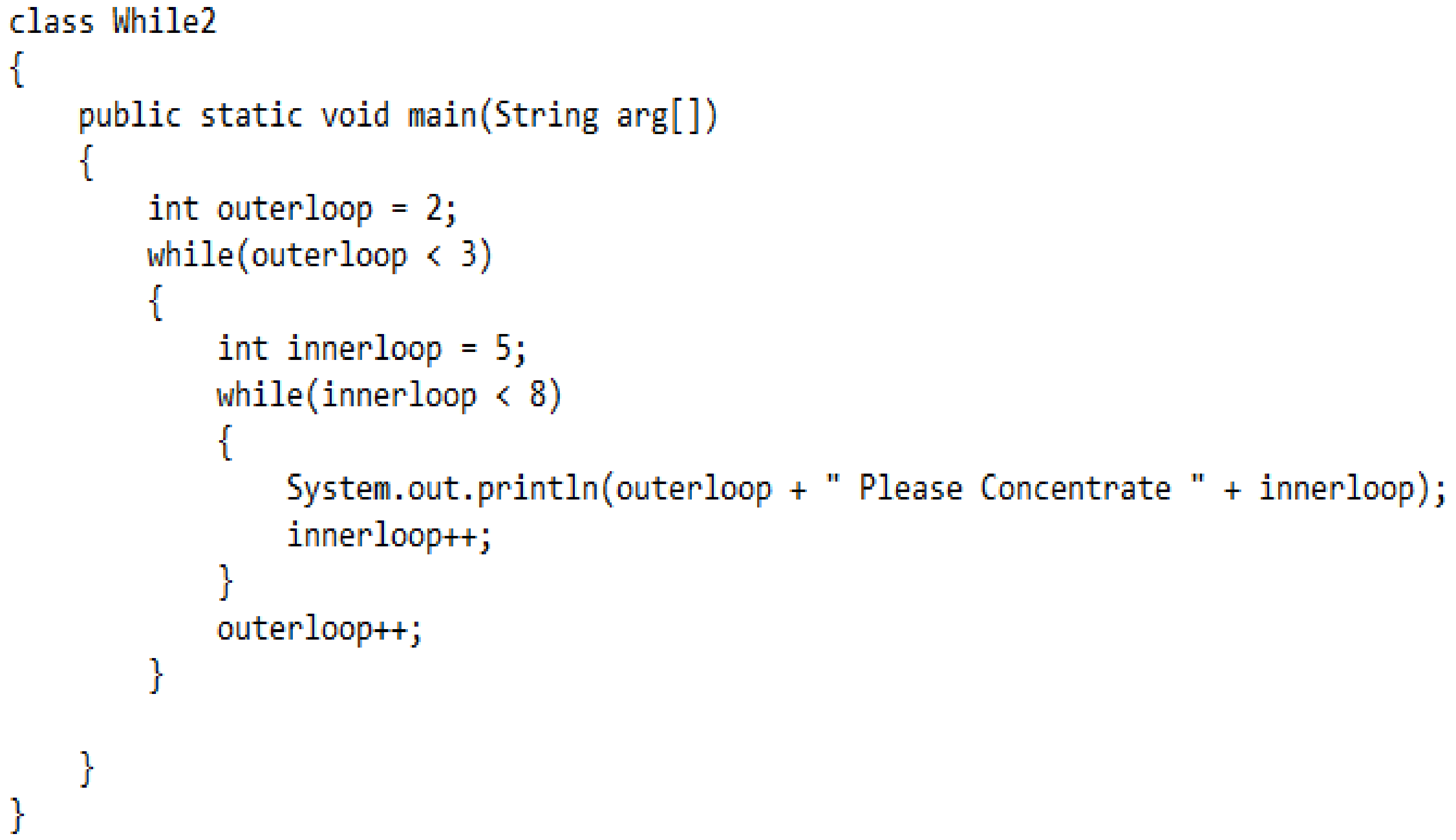
Nested Loop

- Similar to nested if-else statement, loop can be nested as well.

- The body of a loop can contain another loop.

- For each iteration of the outer loop, the inner loop iterates completely.

Nested Loop (Cont…)



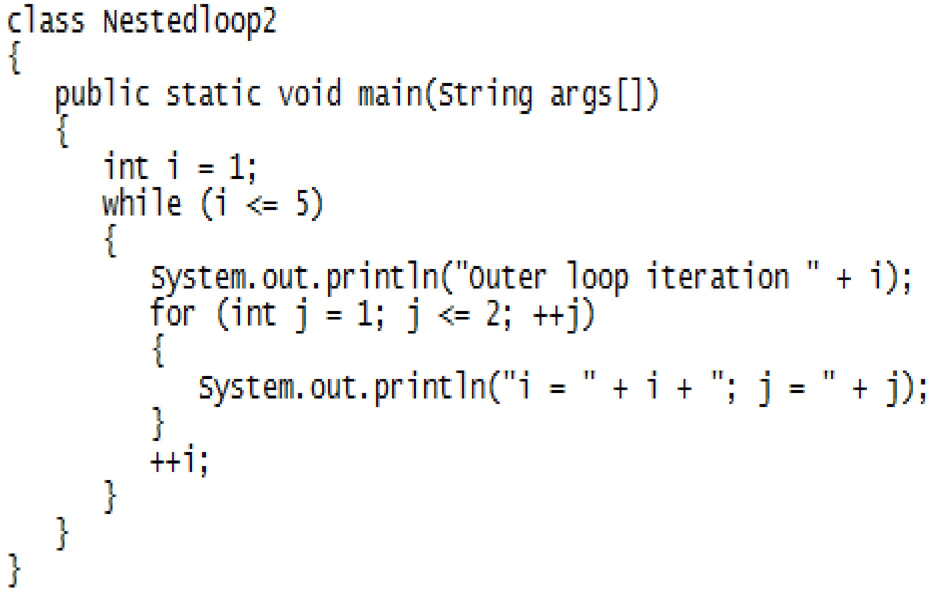
```
class While2
{
    public static void main(String arg[])
    {
        int outerloop = 2;
        while(outerloop < 3)
        {
            int innerloop = 5;
            while(innerloop < 8)
            {
                System.out.println(outerloop + " Please Concentrate " + innerloop);
                innerloop++;
            }
            outerloop++;
        }

    }
}
```

Nested Loop (Cont…)

- Output

```
2 Please Concentrate 5
2 Please Concentrate 6
2 Please Concentrate 7
```

Nested Loop (Cont…)



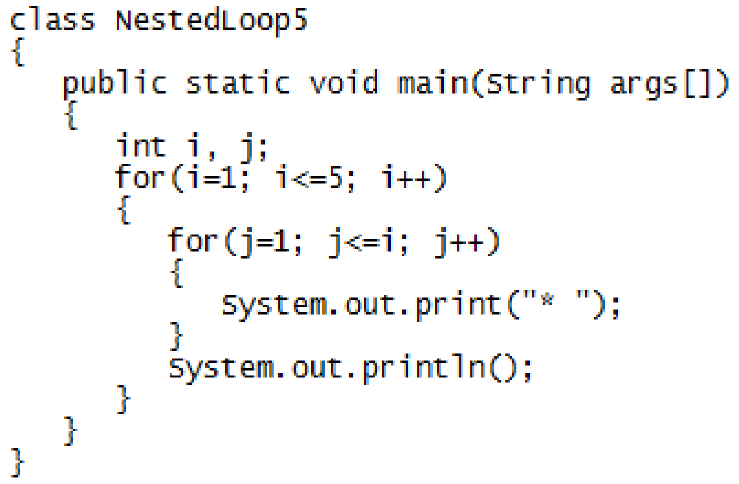
```
class Nestedloop2
{
   public static void main(String args[])
   {
      int i = 1;
      while (i <= 5)
      {
         System.out.println("Outer loop iteration " + i);
         for (int j = 1; j <= 2; ++j)
         {
            System.out.println("i = " + i + "; j = " + j);
         }
         ++i;
      }
   }
}
```

Nested Loop (Cont…)

- Output

```
Outer loop iteration  1
i = 1; j = 1
i = 1; j = 2
Outer loop iteration  2
i = 2; j = 1
i = 2; j = 2
Outer loop iteration  3
i = 3; j = 1
i = 3; j = 2
Outer loop iteration  4
i = 4; j = 1
i = 4; j = 2
Outer loop iteration  5
i = 5; j = 1
i = 5; j = 2
```

Nested Loop (Cont…)



```
class NestedLoop5
{
    public static void main(String args[])
    {
        int i, j;
        for(i=1; i<=5; i++)
        {
            for(j=1; j<=i; j++)
            {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Nested Loop (Cont…)

- Output

```
*
* *
* * *
* * * *
* * * * *
```

thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.