Lecture (05)
Loops

Dr. Ahmed ElShafee

Agenda

- Introduction
- For Loops
- While loops
- Do…While loop
- Difference between while and do…while loop

Introduction

- As we mentioned earlier, the programming you are doing now is sequential programming.
- This means that flow is downward, from top to bottom, with every line of code being executed, unless you tell Java otherwise.
- You saw in the last lecture that one way to "tell" Java not to execute every line is by using IF Statement to section off areas of code.
- in this lecture we will test loops.
- loop is one that forces the program to go back up again.
- If it is forced back up again you can execute lines of code repeatedly.

- As an example, suppose you wanted to add up the numbers 1 to 10.
- You could do it quite easily in Java like this:

int addition = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

- But you wouldn't really want to use that method if you needed to add up the
- numbers 1 to a 1000. Instead, you can use a loop to go over a line of code repeatedly until you've reached 1000.
- Then you can exit the loop and continue on your way.

For Loops

- The structure of the For Loop is this:

for (start_value; end_value; increment_number) {

//YOUR_CODE_HERE

}

- Inside of the round brackets you need three things:
  - the start value for the loop,
  - the end value for the loop,
  - and a way to get from one number to another.
- This is called the increment number, and is usually 1.
- But it doesn't have to be. You can go up in chunks of 10, if you want.

- After the round brackets you need a pair of curly brackets.
- The curly brackets are used to section off the code that you want to execute repeatedly.

For (loopVal =0; loopVal < end_value; loopVal++)

- The first part tells Java at what number you want to start looping.
- The second part uses some conditional logic:

loopVal < end_value

- The **for** loop will then keep going
- As long as it's true that **loopVal** is less than **end_value**, Java will keep looping over the code between the curly brackets.

- The final part between the round brackets of the **for** loop is this:

loopVal++

- Instead of saying **loopVal++** we could have wrote this:

loopVal = loopVal + 1

- Java will then add 1 to whatever is currently stored in the **loopVal** variable.
- Once it has added 1 to the value, it will store the result inside of the variable to the left of the equals sign.
- It is so common that the shorthand notation **variable++** was invented:

```
int some_number = 0;
some_number++;
```

ForLoop01

```
Output - IfLoop01 (run)
run:

Line number = 0
Line number = 1
Line number = 2
Line number = 3
Line number = 4
Line number = 5
Line number = 6
Line number = 7
Line number = 8
Line number = 9
Line number = 10BUILD SUCCESSFUL (total time: 0 seconds)
```

```
1  /*...*/
5  package forloop01;
6
7  /**...*/
11 public class ForLoop01 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n;
18         int start_value=0;
19         int end_value=11;
20         for(n=start_value;n<end_value;n++)
21         {
22             System.out.printf("\nLine number = %d", n);
23         }
24     }
25 }
```

- So we've trapped the program in a loop, and forced it to go round and round.
- Each time round the loop, 1 gets added to the **loopVal** variable.
- The loop keeps going round and round while the value inside of **loopVal** is less than the value in **end_value**.
- Whatever is inside of the loop's curly brackets is the code that will be executed over and over.
- And that is the whole point of the loop: to execute the curly bracket code over and over.
- The nNext example adds up the numbers 1 to 10.

ForLoop02

```
1  /*...*/
5  package forloop02;
6
7  /**...*/
11 public class ForLoop02 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n;
18         int start_value=0;
19         int end_value=11;
20         int addition=0;
21         for(n=start_value;n<end_value;n++)
22         {
23             addition=addition+n;
24         }
           System.out.printf("\nTotal = %d", addition);
26     }
27 }
```

```
Output - ifloop02 (run)
run:
Total = 55BUILD SUCCESSFUL (total time: 0 seconds)
```

Exercise

- Change you code so that the loop adds up the numbers 1 to a 100.
- The answer you should get is 5050.

```
1  /*...*/
5  package forloop03;
6
7  /**...*/
11 public class ForLoop03 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n;
18         int start_value=0;
           int end_value=101;
20         int addition=0;
21         for(n=start_value;n<end_value;n++)
22         {
23             addition=addition+n;
24         }
25         System.out.printf("\nTotal = %d", addition);
26     }
27 }
```

```
Output - ForLoop03 (run)
run:
Total = 5050BUILD SUCCESSFUL (total time: 1 second)
```

Exercise

- Write a times table program.
- The program should ask a user to input a number.
- This number is then used as the times table.
- So if the user enters 10, the 10 times table should be displayed.
- Your Output window should look something like this, when your program is run.

```
run:
Which times table do you want?
 10
1 times 10 = 10
2 times 10 = 20
3 times 10 = 30
4 times 10 = 40
5 times 10 = 50
6 times 10 = 60
7 times 10 = 70
8 times 10 = 80
9 times 10 = 90
10 times 10 = 100
BUILD SUCCESSFUL (total time: 3 seconds)
```

ForLoop04

```
Output - ForLoop04 (run)

Enter number to generate times table for it:5

0 * 5 = 0
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25BUILD SUCCESSFUL (total time: 2 seconds)
```

```
1  /*...*/
5  package forloop04;
6  import java.util.Scanner;
7  /**...*/
11 public class ForLoop04 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int x,y;
18         Scanner sc=new Scanner(System.in);
19         System.out.printf("\nEnter number to generate times table for it:");
20         x=sc.nextInt();
21         for(y=0;y<=x;y++)
22         {
23             System.out.printf("\n%d * %d = %d",y,x,x*y);
24         }
25     }
26 }
27
```

Exercise

- Write a program accepts user input the calculate factorial of that number.

ForLoop05

```
Output - ForLoop05 (run)
run:
Enter number to get its factorial:8
Factorial of 8 = 40320BUILD SUCCESSFUL (total time: 2 seconds)
```

```
1  /*...*/
5  package forloop05;
6  import java.util.Scanner;
7  /**...*/
11 public class ForLoop05 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int num,n,res;
18         res=1;
19         Scanner sc=new Scanner(System.in);
20         System.out.printf("\nEnter number to get its factorial:");
21         num=sc.nextInt();
22         for(n=num;n>0;n--)
23         {
24             res=res*n;
25         }
26         System.out.printf("\nFactorial of %d = %d",num,res);
27     }
28 }
```

Exercise

- Use a **for** loop to print out the odd numbers from 1 to 100.

Key:

- the modulus operator.
- Modulus is when you divide by a number and keep the remainder.
- So 10 Mod 3 is 1, because 10 divide by 3 is 3.
- The remainder is 1, and that's what you keep.
- The Modulus operator in Java is the percentage sign, rather confusingly. It's used like this:

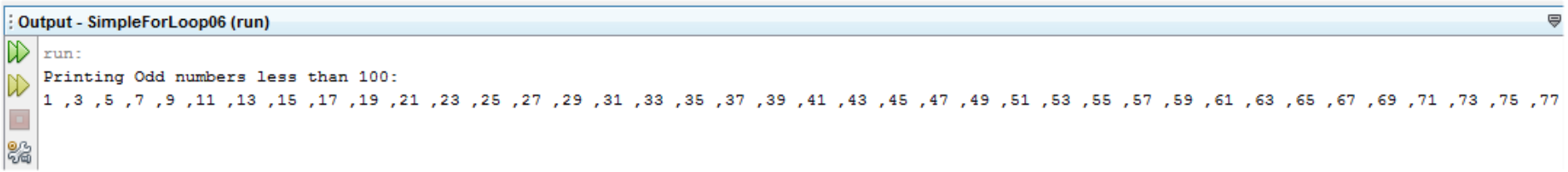
int remainder;

int total = 10

remainder = total %3

SimpleForLoop06

```
1  /*...*/
5  package simpleforloop06;
6
7  /**...*/
11 public class SimpleForLoop06 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n,rem;
18         System.out.printf("Printing Odd numbers less than 100:\n");
19         for(n=0;n<=100;n++)
20         {
21             rem=n%2;
22             if(rem!=0)
23             {
               System.out.printf("%d ,",n);
25             }
26         }
27     }
28 }
29
```


Output - SimpleForLoop06 (run)
run:
Printing Odd numbers less than 100:
1 ,3 ,5 ,7 ,9 ,11 ,13 ,15 ,17 ,19 ,21 ,23 ,25 ,27 ,29 ,31 ,33 ,35 ,37 ,39 ,41 ,43 ,45 ,47 ,49 ,51 ,53 ,55 ,57 ,59 ,61 ,63 ,65 ,67 ,69 ,71 ,73 ,75 ,77

ForLoop06

```
Output - ForLoop06 (run)
run:

Printing Odd numbers less than 100:
1 ,3 ,5 ,7 ,9 ,11 ,13 ,15 ,17 ,19 ,
21 ,23 ,25 ,27 ,29 ,31 ,33 ,35 ,37 ,39 ,
41 ,43 ,45 ,47 ,49 ,51 ,53 ,55 ,57 ,59 ,
61 ,63 ,65 ,67 ,69 ,71 ,73 ,75 ,77 ,79 ,
81 ,83 ,85 ,87 ,89 ,91 ,93 ,95 ,97 ,99 ,
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
1  /*...*/
5  package forloop06;
6
7  /**...*/
11 public class ForLoop06 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n,elements_line,rem;
18         System.out.printf("\nPrinting Odd numbers less than 100:\n");
19         elements_line=0;
20         for(n=0;n<=100;n++)
21         {
22             rem=n%2;
23             if(rem!=0)
24             {
25                 System.out.printf("%d ,",n);
26                 elements_line++;
27                 if(elements_line==10)
28                 {
29                     elements_line=0;
30                      System.out.printf("\n");
31                 }
32
33             }
34         }
35     }
```

Exercise

- Write a program that prints the primary numbers less than 100.

SimpleForLoop07

```
1  /*...*/
5  package simpleforloop07;
6
7  /**...*/
11 public class SimpleForLoop07 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int m,n;
18         Boolean prim;
19         System.out.printf("\nList of prim numbers less than 100 :\n{");
20         for(n=0;n<=100;n++)
21         {
22             prim=true;
23             for(m=2;m<n;m++)
24             {
25                 if((n%m)==0)
26                 {
27                     prim=false;
28                     break;
29                 }
30             }
31             if(prim)
32             {
33                 System.out.printf("%d, ",n);
34             }
35         }
```

```
36              System.out.printf("}\n");
37          }
38      }
39
```

Output - SimpleForLoop07 (run)

```
run:

List of prim numbers less than 100 :
{0, 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, }
BUILD SUCCESSFUL (total time: 0 seconds)
```

ForLoop07

```
1  /*...*/
5  package forloop07;
6
7  /**...*/
11 public class ForLoop07 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int m,n,elements_line=0;
18         Boolean prim;
19         System.out.printf("\nList of prim numbers less than 100 :\n{");
20         for(n=0;n<=100;n++)
21         {
22             prim=true;
23             for(m=2;m<n;m++)
24             {
25                 if((n%m)==0)
26                 {
27                     prim=false;
28                     break;
29                 }
30             }
31             if(prim)
```

```
32                  {
33                      System.out.printf("%d, ",n);
34                      elements_line++;
35                      if(elements_line==10)
36                      {
37                          elements_line=0;
38                          System.out.printf("\n");
39                      }
40                  }
41              }
42              System.out.printf("}\n");
43          }
44      }
45
```

Output - ForLoop07 (run)

```
List of prim numbers less than 100 :
{0, 1, 2, 3, 5, 7, 11, 13, 17, 19,
23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 83, 89, 97, }
BUILD SUCCESSFUL (total time: 0 seconds)
```

While loop

- Another type of loop you can use is called the While loop. **While** loops are a lot easier to understand than **for** loops. Here's what they look like:

while (*condition* **) {**

}

```
int n= 0;
for( n=0;n<5;n++) {
System.out.println("Pri
nting Some Text");
}
```

```
int n= 0;
while ( n< 5 ) {
System.out.println("Pri
nting Some Text");
n++;
}
```

- The condition to test is between the round brackets.
- We want to keep looping while the variable called **loopVal** is less than 5.
- Inside of the curly brackets our code first prints out a line of text.
- Then we need to increment the **loopVal** variable.
- If we don't we'll have an infinite loop, as there is no way for **loopVal** to get beyond its initial value of 0.

WhileLoop01

```
1  /*...*/
5  package whileloop01;
6
7  /**...*/
11 public class WhileLoop01 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n=0;
18         int end_value=11;
19         while(n<end_value)
20         {
21             System.out.printf("\nLine number = %d", n);
22             n++;
23         }
24     }
25 }
26
```

```
Output - WhileLoop01 (run)
Line number = 0
Line number = 1
Line number = 2
Line number = 3
Line number = 4
Line number = 5
Line number = 6
Line number = 7
Line number = 8
Line number = 9
Line number = 10BUILD SUCCESSFUL (total time: 0 seconds)
```

Do...While loop

```
int loopVal = 0;
do {
System.out.println("Printing Some Text");
loopVal++;
}
while ( loopVal < 5 );
```

- Java will loop round and round until the end condition is met.
- This time, the “while” part is at the bottom. But the condition is the same – keep looping while the value inside of the variable called **loopVal** is less than 5.

- The difference between the two is the code between the curly brackets of **do … while** will get executed at least once.
- With the while loop, the condition could already be met.

DoWhileLoop01

```
Output - WhileLoop01 (run)

Line number = 0
Line number = 1
Line number = 2
Line number = 3
Line number = 4
Line number = 5
Line number = 6
Line number = 7
Line number = 8
Line number = 9
Line number = 10BUILD SUCCESSFUL (total time: 0 seconds)
```

```
1  /*...*/
5  package dowhileloop01;
6
7  /**...*/
11 public class DoWhileLoop01 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int n=0;
18         int end_value=11;
19         do
20         {
21             System.out.printf("\nLine number = %d", n);
22             n++;
23         } while(n<end_value);
24     }
25 }
```

Difference between while and do…while loop

- If the loop conditional logic doesn't meet
- Do…while will guarantee executing the loop body for single time, on the other hand while will not execute body at all.

- Write a program get char for user, then print it's ASCII code, to quit the program user should press 'Q' button.

Loop01

```
Output - Loop01 (run)
run:

Enter a char to get its ASCII code (Q to Quit):Q
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
Output - Loop01 (run)

Enter a char to get its ASCII code (Q to Quit):r
r = 114 = 72
Enter Next Char(Q to Quit):Q
BUILD SUCCESSFUL (total time: 7 seconds)
```

```java
/*...*/
package loop01;
import java.util.Scanner;
/**...*/
public class Loop01 {

    /**...*/
    public static void main(String[] args) {
        char ch;
        String str;
        Scanner sc=new Scanner(System.in);
        System.out.printf("\nEnter a char to get its ASCII code (Q to Quit):");
        str=sc.next();
        ch=str.charAt(0);
        while(ch!='Q')
        {
            System.out.printf("%c = %d = %x\nEnter Next Char(Q to Quit):",ch,(int)ch,(int)ch);
            str=sc.next();
            ch=str.charAt(0);
        }
    }
}
```

Loop02

```
Output - Loop02 (run)
Enter a char to get its ASCII code (Q to Quit):Q
Q = 81 = 51
Enter Next Char(Q to Quit):Q
BUILD SUCCESSFUL (total time: 13 seconds)
```

```
Output - Loop02 (run)
Enter a char to get its ASCII code (Q to Quit):r
r = 114 = 72
Enter Next Char(Q to Quit):Q
BUILD SUCCESSFUL (total time: 3 seconds)
```

```java
/*...*/
package loop02;
import java.util.Scanner;
/**...*/
public class Loop02 {

    /**...*/
    public static void main(String[] args) {
        char ch;
        String str;
        Scanner sc=new Scanner(System.in);
        System.out.printf("\nEnter a char to get its ASCII code (Q to Quit):");
        str=sc.next();
        ch=str.charAt(0);
        do
        {
            System.out.printf("%c = %d = %x\nEnter Next Char(Q to Quit):",ch,(int)ch,(int)ch);
            str=sc.next();
            ch=str.charAt(0);
        }while(ch!='Q');
    }
}
```

exercise

- Write a program accepts two inputs from user, calculate permutation if applicable then print it, if not print error message

Loop03

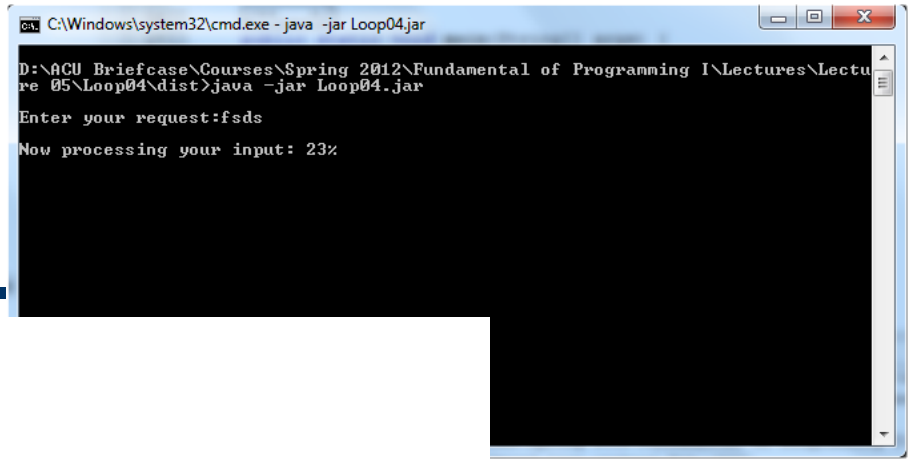
```
1  /*...*/
5  package loop03;
6  import java.util.Scanner;
7  /**...*/
11 public class Loop03 {
12
13     /**...*/
16     public static void main(String[] args) {
17         int x,n,num,r,permutation;
18         permutation=1;
19         Scanner sc=new Scanner(System.in);
20         System.out.printf("\nFor Permutation nPr:\nn=");
21         n=sc.nextInt();
22         System.out.printf("\nr=");
23         r=sc.nextInt();
24         if(r>n)
25         {
26             System.out.printf("\nInvalid input n should be >= r.");
27             System.exit(0);
28         }
29         num=n;
30         for(x=0;x<r;x++)
31         {
32             permutation=permutation*(num--);
33         }
34             System.out.printf("\nPermutation of %dP%d=%d\n",n,r,permutation);
35             System.exit(0);
36     }
37 }
```

```
Output - Loop03 (run)

For Permutation nPr:
n=5

r=3

Permutation of 5P3=60
BUILD SUCCESSFUL (total time: 4 seconds)
```

Exercise

- Write a program simulate a process using percentage indicator xx%.
- Program accept input from user (a string), start processing and counting from 0% to 100%.
- Then print “good bye”

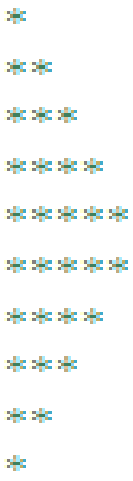
Loop04



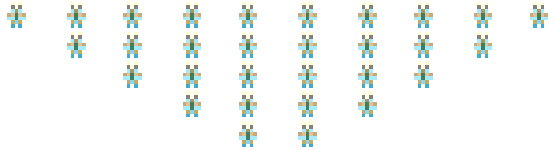

```
1  /*...*/
5  package loop04;
6  import java.util.Scanner;
7
8  /**...*/
12 public class Loop04 {
13
14     /**...*/
17     public static void main(String[] args) {
18         int x=0;
19         String user_input;
20         Scanner sc=new Scanner(System.in);
21         System.out.printf("\nEnter your request:", x);
22         user_input=sc.next();
23         System.out.printf("\nNow processing your input:", x);
24         System.out.printf("%3d%%", x);
25         long t;
26         for(x=0;x<=100;x++)
27         {
28             System.out.printf("\b\b\b\b%3d%%", x);
29             t=System.currentTimeMillis();
               while((System.currentTimeMillis()-t)<100);
31         }
32         System.out.printf("\nDone.");
33     }
34 }
```

Assignments

- Write a program prints the following

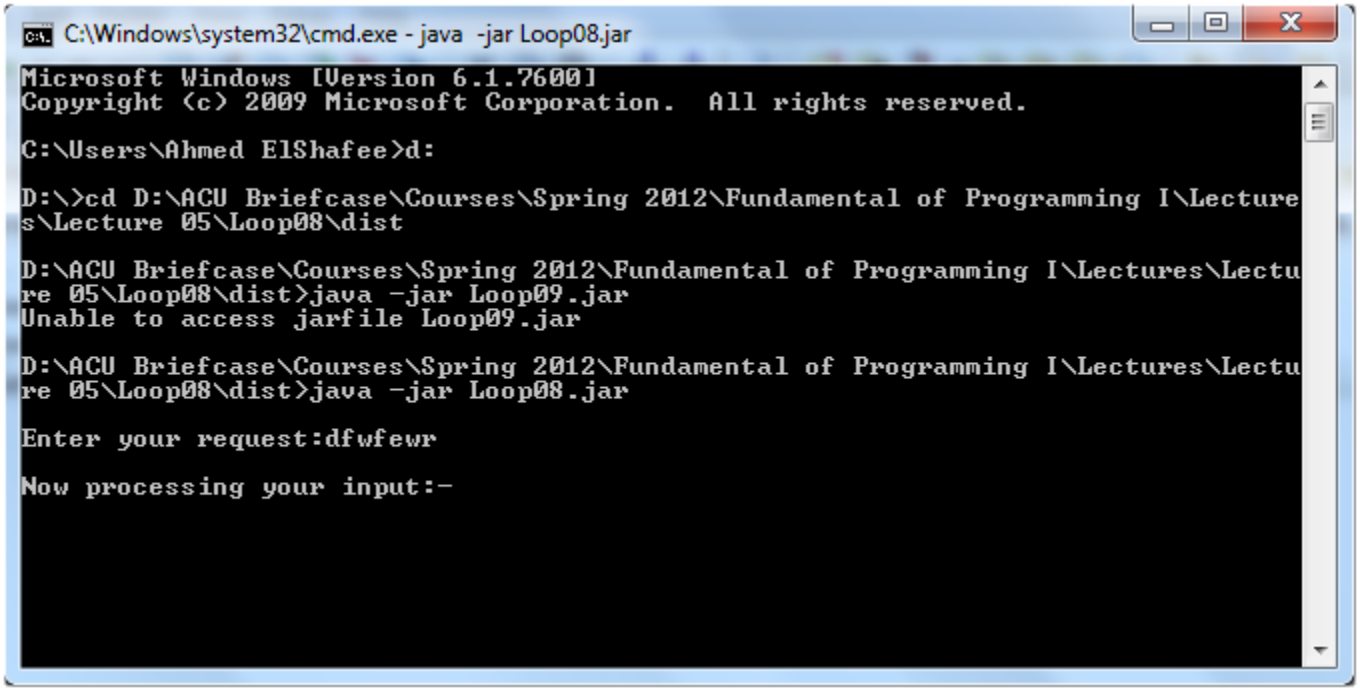


- Write a program prints the following



- Write a calculator program accepts user input (number1, operation, number2) keep running till user press enter only for the first number with out entering any number.

- Write a program simulate a process using the following rotating cursors -/|\ for a while Then print “good bye”



Thanks,

See you next Lecture, isA