Lecture 3

While Loop

Loops

- Computers are often used to automate repetitive tasks. Repeating tasks without making errors is something that computers do well and people do poorly.

- Running the same code multiple times is called **iteration**.

While loop

- The **while** loop loops through a block of code as long as a specified condition is **true**

Syntax:

```
while (condition) {
    // code block to be executed
}
```

Example:

```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

while statement

- The expression in parentheses is called the condition. The statements in braces are called the **body**. The flow of execution for a **while** statement is:

    1. Evaluate the condition, yielding **true** or **false**.

    2. If the condition is **false**, skip the body and go to the next statement.

    3. If the condition is **true**, execute the body and go back to step 1.

- This type of flow is called a **loop**, because the last step loops back around to the first.

Exercise

Ex1: Print number from 1 to 10

Ex2: Print number from 1 to 100

Ex3: Print number from 50 to 100

Ex4: Print only even number from 1 to 100

Ex5: Print sum of number between 1 to 100

Ex6: Print sum of even number between 1 to 100

Ex7: Print sum of even number between 50 to 100

Ex8: Print 10 factorial, 10! = 1*2*3*4*5*6*7*8*9*10

while statement

- The body of the loop should change the value of one or more variables so that, eventually, the condition becomes **false** and the loop terminates.
- Otherwise the loop will repeat forever, which is called an **infinite loop**. An endless source of amusement for computer scientists is the observation that the directions on shampoo, “Lather, rinse, repeat,” are an infinite loop.

The Do/While Loop

- The **do/while** loop is a variant of the **while** loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

```
do {
    // code block to be executed
} while (condition);
```

Example:

```
int i = 0;
do {
    System.out.println(i);
    I++;
} while (i < 5);
```

Exercise 9

- Exercise 9: Alter the example so that it prints all the numbers from n to 1 in reverse order, as in 10 9 … … 2 1. (Hint: To decrement a value inside the loop, use the i = i – 1; statement.)

Output Example (input in **bold**):
Enter a number: **10**
10 9 8 7 6 5 4 3 2 1

Exercise 10

- Exercise 10: Write a program to print all the numbers from n1 to n2, where n1 and n2 are two numbers specified by the user. n1 should less than n2 (Hint: You'll need to prompt for two values n1 and n2; then initialize i to n1 and use n2 in the loop condition.)

  Output Example (input in **bold**):
  Enter a first number: **10**
  Enter a second number: **20**
  10 11 12 13 14 15 16 17 18 19 20

Exercise 11

- Exercise 11: Write a program to print all the numbers between n1 to n2, where n1 and n2 are two numbers specified by the user. (Hint: n1 and n2 can be any integer, so we should decide which one is smaller, and initialize i to the smaller number)

  Output Example 1 (input in **bold**):
  Enter a first number: **10**
  Enter a second number: **20**
  10 11 12 13 14 15 16 17 18 19 20

  Output Example 2 (input in **bold**):
  Enter a first number: **20**
  Enter a second number: **10**
  20 19 18 17 16 15 14 13 12 11 10

Exercise 12

- Exercise 12: Take a user input, and print all the $1^3,2^3,\ldots,n^3$

Output Example (input in **bold**):
Enter a number: **5**
1 8 27 64 125

Exercise 13

Write a program that displays:
a) all numbers from 0 to 100 divisible by 11

Output Example:
0 11 22 33 44 55 66 77 88 99

Exercise 14

Write a program that displays a range of numbers specified by user.
a) Request a minimum and maximum value between 1 and 100.
b) Validate that min < max via an input loop.
c) Output all numbers between and including the maximum and minimum.

Output Example:

```
Display a Range
---------------
Enter a minimum value between 1 and 100: -10
Enter a maximum value between 1 and 100: 20
Values must be within specified range.
Enter a minimum value between 1 and 100: 50
Enter a maximum value between 1 and 100: 40
Maximum value must be greater than minimum value.
Enter a minimum value between 1 and 100: 34
Enter a maximum value between 1 and 100: 54
54 53 52 51 50 49 48 47 46 45 44 43 42 41 39 38 37 36 35 34
```

Exercise 15

Write a program to guess a number between 1 and 10.
a) Set a constant integer variable equal to the number 7.
b) Request a number from 1 to 10.
c) Validate the input value via a do while loop.
d) If the user guesses correctly, inform the user and end the program.
e) If the user guesses incorrectly, inform the user and repeat request.

Output example:
Guess a Number

Pick an integer from 1 to 10: ***-5***
INVALID INPUT! PLEASE RE-ENTER.
Pick an integer from 1 to 10: ***20***
INVALID INPUT! PLEASE RE-ENTER.
Pick an integer from 1 to 10: ***6***
Oops! 6 is not the correct number.
Pick an integer from 1 to 10: ***3***
Oops! 3 is not the correct number.
Pick an integer from 1 to 10: ***7***
Congratulations! 7 is the correct number.

Exercise 16

- Ask user input an integer, and test if it is a prime number. (prime test)

- prime test: a natural number greater than 1 that has no positive divisors other than 1 and itself.

Output Example 1 (input in **bold**):
Enter a first number: **11**
11 is a prime number

Output Example 2 (input in **bold**):
Enter a first number: **25**
25 is not a prime number

Exercise 17

Write a program that requests an integer and displays: - all factors of the input (numbers that evenly divide the input) (i.e. 1,2,3,5,6,10,15 are all factors of 30) - if the input is prime (input that is only divisible by 1 and itself)
a) Request an integer value between 1 and 100.
b) Validate input value type and range via a do while loop.
c) Display all factors of the value.
d) Indicate if the number is prime

Output Example1:
Enter an integer between 1 and 100: ***5.3***
Invalid value.
Enter an integer between 1 and 100: ***103***
Invalid value.
Enter an integer between 1 and 100: ***56***
Factors of 56:
1 2 4 7 8 14 28 56

Output Example2:
Enter an integer between 1 and 100: ***0***
Invalid value.
Enter an integer between 1 and 100: ***5***
Factors of 5:
1 5
5 is a prime number.

Exercise 18

- Take an integer, and print backtrack. Example: input is 12345, then print 54321.

Output Example (input in **bold**):
Enter a number: **145671**
176541

Exercise 19

- Take a user input, check how many digit is the integer.

Output Example (input in **bold**):
Enter a number: **145671**
There are 6 digits in 145671

Exercise 20

- Take two user input, check they have same number of digit, if not let the user re-enter two number until these have same number of digits.

  Output Example (input in **bold**):
  Enter a first number: **1**
  Enter a second number: **21**
  Invalid input, re-enter two numbers
  Enter a first number: **1222**
  Enter a second number: **89**
  Invalid input, re-enter two numbers
  Enter a first number: **123**
  Enter a second number: **456**
  123 and 456 both have 3 digits.

Exercise 21

- Take two user input, and print two number alternatively reverse. The two integer should be same digit. If not ask user enter again. Example: input is 12345, and 67890, output is 5049382716.

  Output Example (input in **bold**):
  Enter a first number: **123**
  Enter a second number: **456**
  362514

Exercise 22

- Take two user input, and print two number alternatively. (first you need to check they have same number of digit, if not let the user re-enter two number) Example: input is 12345, and 67890, output is 1627384950.

Output Example (input in **bold**):
Enter a first number: **1**
Enter a second number: **21**
Invalid input, re-enter two numbers
Enter a first number: **1222**
Enter a second number: **89**
Invalid input, re-enter two numbers
Enter a first number: **123**
Enter a second number: **456**
142536

For Loop

For loop

- The loops we have written so far have several elements in common. They start by initializing a variable, they have a condition that depends on that variable, and inside the loop they do something to update that variable.

- This type of loop is so common that there is another statement, the **for** loop, that expresses it more concisely.

For Loop

- When you know exactly how many times you want to loop through a block of code, use the **for** loop instead of a **while** loop:

Syntax:

```
for (statement 1; statement 2; statement 3) {
    // code block to be executed
}
```

Example:

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

For Loop

- **for** loops have three components in parentheses, separated by semicolons: the initializer, the condition, and the update.

1. The *initializer* runs once at the very beginning of the loop.
2. The *condition* is checked each time through the loop. If it is **false**, the loop ends. Otherwise, the body of the loop is executed (again).

3. At the end of each iteration, the *update* runs, and we go back to step 2.

- The **for** loop is often easier to read because it puts all the loop-related statements at the top of the loop.

While vs For

- There are no definitive rules stating when to use which loop. Here is some recommendations.
- Use while loop if
  - Do not know number of times to loop in advance
  - Loop condition is base on user input
- Use for loop if…
  - Number of times to loop is known in advance
  - If nesting loops is needed for logical flow
- There is another difference between **for** loops and **while** loops: if you declare a variable in the initializer, it only exists inside the **for** loop.

Break

- You have already seen the **break** statement used in an earlier chapter of this tutorial. It was used to "jump out" of a **switch** statement.

- The **break** statement can also be used to jump out of a **loop**.

```
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        break;
    }
    System.out.println(i);
}
```

Continue

- The **continue** statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        continue;
    }
    System.out.println(i);
}
```

Exercise

Ex1: Print number from 1 to 10

Ex2: Print number from 1 to 100

Ex3: Print number from 50 to 100

Ex4: Print only even number from 1 to 100

Ex5: Print sum of number between 1 to 100

Ex6: Print sum of even number between 1 to 100

Ex7: Print sum of even number between 50 to 100

Ex8: Print 10 factorial, 10! = 1*2*3*4*5*6*7*8*9*10

Nested For loop

Syntax:

```
for(initialize; comparison; update){
    Statement1;
    for(initialize; comparison; update)
        Statement2;
}
```

Nested For loop

Syntax:

```
 for(int i=0; i<10; i++){
   some code;
   for(int j=0; j < 10; j++){
     some code;
   }
   some code;
}
```

Nested For loop

Syntax:

```
 for(int i=0; i<10; i++){
   some code;
   for(int j=0; j < i; j++){
     some code;
   }
   some code;
}
```

Exercise

- Exercise 14: Ask user input an integer, and print the number as following.
  Enter integer: 4
  1
  1 2
  1 2 3
  1 2 3 4

- Exercise 15: Ask user input an integer, and print the alphabet as following.
  Enter integer: 4
  a
  a b
  a b c
  a b c d

- Exercise 16: Ask user input an integer, and print the number as following.
  Enter integer: 4
  1 2 3 4
  1 2 3
  1 2
  1

Exercise

- Exercise 17: Ask user input an integer, and print the even numbers as following.
  Enter integer: 4
  2
  2 4
  2 4 6
  2 4 6 8

- Exercise 18: Ask user input an integer, and print the alphabet as following.
  Enter integer: 7
  ```
  a
  b c
  d e f
  g h i j
  k l m n o
  p q r s t u
  v w x y z a b
  ```

- Exercise 19: Ask user input an integer, and print the number as following.
  Enter integer: 4
  ```
  1  2  3  4
  5  6  7
  8  9
  10
  ```

Exercise

- Exercise 20: A table of triangle number (user supplies height)
  Enter integer: 4
  1 = 1
  1 2 = 3
  1 2 3 = 6
  1 2 3 4 = 10

- Exercise 21: A table of factorials (user supplies height)
  Enter integer: 4
  1!=1=1
  2!=1 2=2
  3!=1 2 3=6
  4!=1 2 3 4=24

- Exercise 22: A table of 2's power (user supplies number)
  Enter integer: 4
  2 to power 0 is 1
  2 to power 1 is 2
  2 to power 2 is 4
  2 to power 3 is 8

Exercise

- Exercise 23: Rectangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

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

- Exercise 24: Given a value n, output the following image
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

```
*
 *
  *
   *
    *
```

Exercise

- Exercise 25: Given a value n, output the following image
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

```
    *
   *
  *
 *
```

- Exercise 26: Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

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

Exercise

- Exercise 27: Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5
  ```
  *****
  ****
  ***
  **
  *
  ```

- Exercise 28: Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5
  ```
      *
     **
    ***
   ****
  *****
  ```

Exercise

- Exercise 29: Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

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

- Exercise 30: Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

  Enter height: 5

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

Exercise

- Exercise 31: Another Triangle of stars (user supplies height)
  a) Request a value for n.
  b) Output the following image.

Enter height: 5

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

Exercise

- Exercise 32: Another Triangle of stars (user supplies height)

Enter height: 7

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