BIRZEIT UNIVERSITY

Loops

By: Mamoun Nawahdah (Ph.D.)
2022

Opening Problem

Problem:

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");

...
...
...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

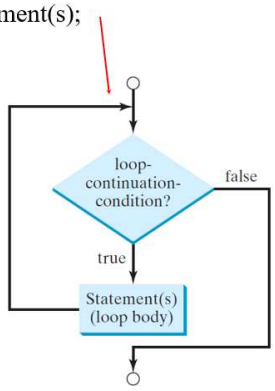
2

Introducing while Loops

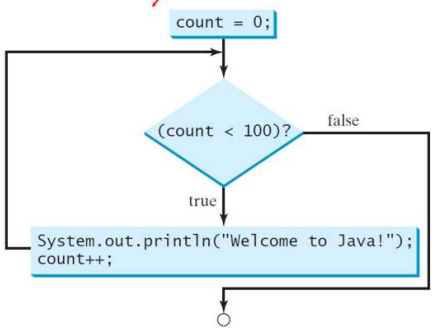
```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java");
  count++;
}
```

while Loop Flow Chart

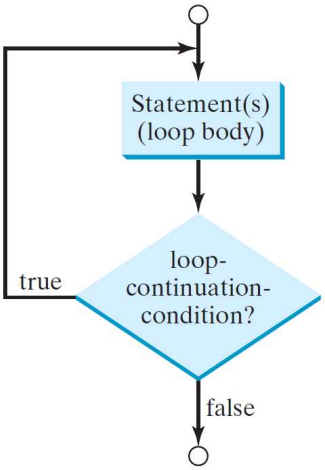
```
while (loop-continuation-condition) {
  // loop-body;
  Statement(s);
}
```




```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```



do-while Loop




```
do {
    // Loop body;
    Statement(s);
} while (loop-continuation-condition);
```

5

Problem: Guessing Numbers

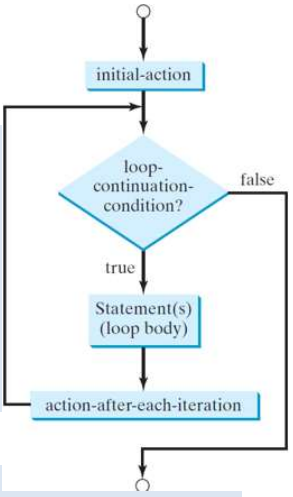
- Write a program that assumes an integer between 0 and 100, inclusive.
- The program prompts the user to enter a number continuously until the number matches the assumed number.
- For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently.

Ending a Loop with a Sentinel Value

- Often the number of times a loop is executed is not predetermined.
- You may use an input value to signify the end of the loop.
- Such a value is known as a **sentinel** value.
- Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

for Loops

```
for ( initial-action ;
      loop-continuation-condition ;
      action-after-each-iteration ) {
        // loop body;
        Statement(s);
}
```




```
for (int i = 0 ; i < 100 ; i++) {
    System.out.println( "Welcome to Java!");
}
```

8

Note

❖ The **initial-action** in a for loop can be a list of zero or more comma-separated expressions.

❖ The **action-after-each-iteration** in a for loop can be a list of zero or more comma-separated statements.

❖ Therefore, the following two for loops are correct:

```
for ( int i = 1 ;  i < 100 ;  System.out.println(i++)) ;
```

```
for (  int i = 0 , j = 0 ;  (i + j < 10) ;  i++, j++  ) {
        // Do something
}
```

9

Note

❖ If the **loop-continuation-condition** in a for loop is omitted, it is implicitly **true**.

❖ Thus the statement given below in (a), which is an **infinite loop**, is correct.

```
for ( ; ; ) {
  // Do something
}
```

Equivalent

```
while (true) {
  // Do something
}
```

10

Caution

❖ Adding a **semicolon** at the end of the for clause before the loop body is a common mistake, as shown below:

Logic Error

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

11

Caution

❖ Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10);
{
    System.out.println("i is " + i);
    i++;
}
```

Logic Error

❖ In the case of the do loop, the following semicolon is needed to end the loop:

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10);
```

Correct

12

Which Loop to Use?

- Use the one that is most intuitive and comfortable for you.
- In general, a for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
- A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

Nested Loops

- Problem: Write a program that uses nested for loops to print a multiplication table.

×	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Finding the Greatest Common Divisor

- Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor (GCD).
- Solution:
  - Let the two input integers be n1 and n2.
  - You know number 1 is a common divisor, but it may not be the greatest commons divisor.
  - So you can check whether k (for k = 2, 3, 4, and so on) is a common divisor for n1 and n2, until k is greater than n1 or n2.

break

```
public class TestBreak {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      sum += number;
      if (sum >= 100)
        break;
    }

    System.out.println("The number is " + number);
    System.out.println("The sum is " + sum);
  }
}
```

16

continue

```
public class TestContinue {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      if (number == 10 || number == 11)
        continue;
      sum += number;
    }

    System.out.println("The sum is " + sum);
  }
}
```

17

Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:

- For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
- Determine whether a given number is prime.
- Count the prime numbers.
- Print each prime number, and print 10 numbers per line.

18

Problem: Checking Palindrome

- A string is a **palindrome** if it reads the same forward and backward. The words “mom,” “dad,” and “noon,” for instance, are all palindromes.
- The problem is to write a program that prompts the user to enter a string and reports whether the string is a palindrome.
- One solution is to check whether the first character in the string is the same as the last character. If so, check whether the second character is the same as the second-to-last character. This process continues until a mismatch is found or all the characters in the string are checked, except for the middle character if the string has an odd number of characters.