# COMPUTER PROGRAMMING CONCEPTS

CS&IT 1101

Instructor: Shakar H. Salih
E-mail: shakar.salih@uhd.edu.iq

# LECTURE 5: BASIC ELEMENTS OF JAVA II

## OUTLINES

- Reading a Single Character
- String Methods
- Increment and Decrement
- Escape Character

## MORE EXAMPLE ABOUT SCANNER

- Write a program in java to add two integer numbers together by using scanner.
- Write a program in java to enter your name by using scanner.
- Write a program in java to enter your first and second name, age and weight.

## READING A SINGLE CHARACTER (1/2)

Every character in a string has a specific position in the string. The position of the first character is 0, the position of the second character is 1, and so on. The **length** of a string is the number of characters in it.

**Syntax:**

```
String Uni="University of Human Development";
    char oneChar=Uni.charAt(11);
    System.out.println( oneChar);
```

## READING A SINGLE CHARACTER (2/2)

- Suppose you want to store a character into a **char** variable using an input statement. During program execution, when you input the character, you do not include the single quotation marks.
- Suppose that **ch** is a **char** variable. Consider the following input statement:

1. For single word:

```
char str=name.next().charAt(3);
      System.out.println(str);
```

2. For multiLine words:

```
char str=name.nextLine().charAt(3);
      System.out.println(str);
```

## STRING METHODS (1/2)

1. **substring(int begin):**
   - Returns substring from begin index to end of the String.

```
String s="shakar";
System.out.println(s.substring(2)); //akar
```

2. **equals():**
   - To perform content comparison where case is important.

```
String s="java";
System.out.println(s.equals("java"));//true
```

3. **concat():**
   - Adding two strings we use this method

```
String s="shakar";
s= s.concat("Hussein");
System.out.println(s);// shakarHussein
```

## STRING METHODS (2/2)

4. **Int length:**
   - Returns the number of characters in the string.

```
String s="shakar";
System.out.println(s.length());// 6
```

## EXAMPLE

```
int firstNum=4;
int secondNum=2*firstNum+6;
char ch='A';
double z=(firstNum+1)/2.0;
secondNum = console.nextInt();  // input 8
z = console.nextDouble();          //16.3
firstNum = (int)(z) + 8;
secondNum = secondNum + 1;
ch = console.next().charAt(0);
firstNum = firstNum + (int)(ch);
```

## INCREMENT AND DECREMENT OPERATOR(1/3)

- Statements are frequently used to keep track of how many times certain things have happened.
- The ++ is increment operator.
- The **--** is decrement operator.
- Java provides the increment operator, ++, which increases the value of a variable by 1,
- The decrement operator, --, which decreases the value of a variable by 1
- Suppose **count** is an **int** variable. The statement:

```
count = count + 1;
```

## INCREMENT AND DECREMENT OPERATOR(2/3)

- Increment and decrement operators each have two forms: **pre** and **post**.
- The syntax of the increment operator is:

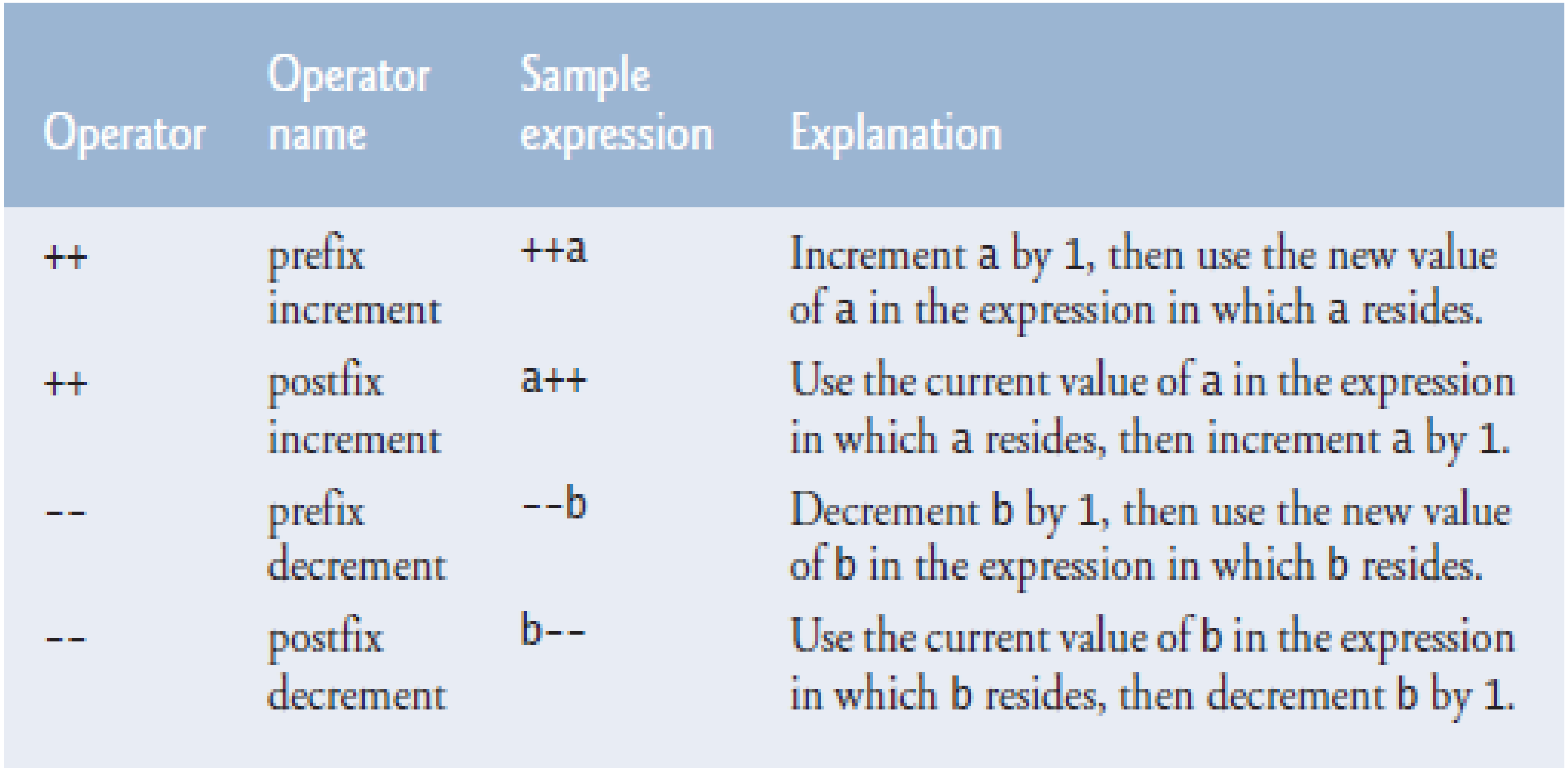
```
Pre-increment:     ++variable
Post-increment:      variable++
```

- The syntax of the decrement operator is:

```
Pre-decrement:     --variable
Post-decrement:      variable--
```

## INCREMENT AND DECREMENT OPERATOR(3/3)

- What is the difference between the **pre** and **post** forms of these operators?
- The difference becomes apparent when the variable using these operators is employed in an expression.
- Suppose that **x** is a variable of type **int**. If **++x** is used in an expression, first the value of **x** is incremented by **1**, and then the new value of **x** is used to evaluate the expression. On the other hand, if **x++** is used in an expression, first the current value of **x** is used in the expression, and then the value of **x** is incremented by **1**.



| Operator | Operator name | Sample expression | Explanation |
|---|---|---|---|
| ++ | prefix increment | ++a | Increment a by 1, then use the new value of a in the expression in which a resides. |
| ++ | postfix increment | a++ | Use the current value of a in the expression in which a resides, then increment a by 1. |
| -- | prefix decrement | --b | Decrement b by 1, then use the new value of b in the expression in which b resides. |
| -- | postfix decrement | b-- | Use the current value of b in the expression in which b resides, then decrement b by 1. |

## EXAMPLE 1

```
int c;
    c = 5;
    System.out.println( c );
    System.out.println(c++);
   System.out.println(c);

    c = 5;
    System.out.println( c );
    System.out.println( ++c );
    System.out.println( c );
```

## EXAMPLE 2

```
int x=5;
int y=++x;
```

- The first statement assigns the value **5** to **x**. To evaluate the second statement, which uses the pre-increment operator, first the value of **x** is incremented to **6**, and then this value, **6**, is assigned to **y**. After the second statement executes, both **x** and **y** have the value **6**.

## EXAMPLE 3

```
int x = 5;
int y = x++;
```

- As before, the first statement assigns **5** to **x**. In the second statement, the post-increment operator is applied to **x.** To execute the second statement, first the value of **x**, which is **5**, is used to evaluate the expression, and then the value of **x** is incremented to **6**. Finally, the value of the expression, which is **5**, is stored in **y**. After the second statement executes, the value of **x** is **6**, and the value of **y** is **5**.

## EXAMPLE 4

1.
```
int a=5;
int b=2 + (++a);
```

2.
```
int a=5;
int b=2 + (a++);
```

## EXAMPLE 5

1.
```
            int a=5;
            int b=6;
            int c;
      a= (b++) + 3;
      c= 2 * a + (++b);
      b= 2 * (++c) - (a++);
```

## ESCAPE CHARACTER

| | Escape | Sequence Description |
|---|---|---|
| \n | Newline | Cursor moves to the beginning of the next line |
| \t | Tab | Cursor moves to the next tab stop |
| \b | Backspace | Cursor moves one space to the left |
| \r | Return | Cursor moves to the beginning of the current line (not the next line) |
| \\ | Backslash | Backslash is printed |
| \’ | Single quotation | Single quotation mark is printed |
| \” | Double quotation | double quotation mark is printed |

# EXAMPLE 1

1. **System.out.print("Hello there. ");**

   **System.out.print("My name is James.");**

**If these statements are executed in sequence, the output is:**

```
Hello there. My name is James.
```

2. **System.out.print("Hello there.\n");**

   **System.out.print("My name is James.");**

**The output of these Java statements is:**

```
Hello there.
My name is James.
```

3. **System.out.print("Hello \nthere. \nMy name is James.");**

**is:**

```
Hello
there.
My name is James.
```

## EXAMPLE 2

4. System.out.println("The tab character is represented as \'\\t\'");

is:

```
The tab character is represented as '\t'
```

## PROGRAMING EXERCISES

1. Write a program that prints the following banner:

```
JJJJJJJJJJJJ      AAA     VV         VV      AAA
     JJ          AA AA     VV       VV      AA AA
     JJ         AA   AA     VV     VV      AA   AA
     JJ        AAAAAAAAA     VV   VV      AAAAAAAAA
 JJ  JJ       AA       AA     VV VV      AA       AA
 JJJJJJ      AA         AA     VVV      AA         AA
```

# THANK YOU…….

DO YOU HAVE ANY QUESTİONS ?