# OUTPUT STATEMENTS

CSC 111

# Outline

1. System.out.print
2. System.out.println
3. Escape Sequences
4. System.out.printf
5. Format Specifiers

# 3. OUTPUT STATEMENTS

| SYNTAX | System.out.print (expression); |
|---|---|
| Example | System.out.print (29/4); |

Output

```
1 7_
```

- Note the cursor position (_): it is on the same line as the output.
- Note that 29/4 is not enclosed between quotes ➔ it is evaluated.

| SYNTAX | System.out.println (expression); |
|---|---|
| Example | System.out.println (29/4); |

Output

```
1 7
2 _
```

- Note the cursor position (_): it is on the next line to the output.

| SYNTAX | System.out.println (); |
|---|---|

- Skips a line.

Output

```
1 _
```

# 3. OUTPUT STATEMENTS

## PROGRAM 1

```
1  // import necessary libraries
2  public class userOutput1
3  {
4     public static void main (String[] args)
5       {
6          // Declaration section: to declare needed variables
7          // Input section: to enter values of used variables
8          // Processing section: processing statements
9          // Output section: display program output
10            System.out.print ("Hello there");              //output line 1
11            System.out.println ("My name is Fatma"); //output line 1
12            System.out.println ("I am studying Java"); //output line 2
13      } // end main
14 } // end class
```

- The output of Program 1 is as follows:

```
1  Hello thereMy name is Fatma
2  I am studying Java
3  _
```

# 3. OUTPUT STATEMENTS

## ESCAPE SEQUENCES

- Escape sequences allow you to control the output.
- All escape sequences start with a backslash \ character.
- The following table shows some of the most commonly used escape sequences:

| Syntax | Escape Sequence | Description |
|---|---|---|
| \n | New line | Cursor moves to the beginning of the next line. |
| \r | Return | Cursor moves to the start of the current line. |
| \t | Tab | Cursor moves to the next tab stop. |
| \b | Backspace | Cursor moves one space to the left. |
| \\ | Backslash | Backslash is printed. |
| \’ | Single quote | Single quotation is printed. |
| \” | Double quotes | Double quote is printed |

# 3. OUTPUT STATEMENTS

## PROGRAM 2

```
1  // import necessary libraries
2  public class userOutput2
3  {
4     public static void main (String[] args)
5       {
6          // Declaration section: to declare needed variables
7          // Input section: to enter values of used variables
8          // Processing section: processing statements
9          // Output section: display program output
10            System.out.print ("Hello there \t");          //output line 1
11            System.out.print ("My name is Fatma\n");//output line 1
12            System.out.println ("I am studying \"Java\"");//output line 2
13      } // end main
14 } // end class
```

- The output of Program 2 is as follows:

```
1  Hello there          My name is Fatma
2  I am studying "Java"
3  _
```

# 3. PRINTF STATEMENT

| | |
|---|---|
| SYNTAX | System.out.printf (formatString); |
| Example 1 | System.out.printf (“Hello there!!”); |
| Output | 1 Hello there!!_ |

| | |
|---|---|
| SYNTAX | System.out.printf (formatString, argumentList); |

- The argumentList is a list of one or more arguments: constant values, variables, or expressions.
- If the argumentList has more than one argument, then the arguments are separated by commas.

# 3. PRINTF STATEMENT

## FORMATTING INTEGER NUMBERS

Example 1

```
int x = 120;
System.out.printf ("The value of x = %d", x);
```

- "The value of x = %d" is the formatString. x is the argumentList.
- %d is called a format specifier. There is one-to-one correspondence between the format specifier and the arguments in the argumentList.
- The format specifier used depends on the type of the argument. %d is used for the variables of type int.

Output

```
1 The value of x = 120_
```

Example 2

```
int cm = 25;
int mm = 2500;
System.out.printf ("There are %d mm in %d cm", mm, cm);
```

- The formatString is "There are %d mm in %d cm".
- The argumentList is mm, cm

Output

```
1 There are 2500 mm in 25 cm_
```

# 3. PRINTF STATEMENT

## FORMATTING FLOATING-POINT NUMBERS

- The default output of floating-point numbers is:
  - up to 6 decimal places for float values, and
  - up to 15 decimal places for double values.
- The %f is the format specifier used for floating-point numbers.
- %.2f specifies the number of digits printed after the decimal point (2 in this example).

Example 1

```
static final float PI = 3.14159f;
double radius = 7.534;
System.out.printf ("PI = %.3f and radius = %.1f ", PI, radius);
```

Output

```
1 PI = 3.142 and radius = 7.5_
```

- Note that the numbers are approximated.

Example 2

```
double area = 227.534;
System.out.printf ("Area = %8.2f", area);
```

Output

```
1 Area = ~~227.53_
```

# 3. PRINTF STATEMENT

## FORMATTING CHARACTER VARIABLES

- The %c is the format specifier used for char values.

Example

```
char option = 'Y';
System.out.printf ("Option = %c", option);
```

Output

```
Option = Y_
```

- The following are other format specifiers:

| Specifier | Description |
|---|---|
| %d | The result is formatted as a (decimal) integer |
| %f | The result is formatted as a decimal floating-number |
| %c | The result is a Unicode character |
| %s | The result is a string |
| %e | The result is formatted as a scientific notation |
| %n | Line separator |
| %% | Prints % |

Escape sequences are also used

# 3. PRINTF STATEMENT

## NOTES

- By default, the output is right-justified for all primitive data types.

Example 1

```
String name = "aly";
System.out.printf ("First name = %6s%n", name);
```

Output

```
1 First name = ~~~aly
2 _
```

- To print it left-justified, precede the width with a **–** sign:

Example 2

```
String name = "aly";
System.out.printf ("First name = %-6s%n", name);
```

Output

```
1 First name = aly~~~
2 _
```

Example 3

```
int year = 2015;
System.out.printf ("Academic year = %-6d-%6d", year, ++year);
```

Output

```
1 Academic year = 2015~~-~~2016_
```

# Self-Check Exercises (1)

- What is the output of the following program:

```
1  public class selfCheck1
2  {
3     public static void main (String[] args)
4       {
5          int num = 52033;
6          double x = 9234.8667;
7          String str = "Computer Science";
8          System.out.printf ("%-5d%-10.2f%-20s ***%n", num, x, str);
9          System.out.printf ("%-25s%-6d%-12.3f ***%n", str, num, x);
10         System.out.printf ("%-13.4f%-7d%-22s ***%n", x, num, str);
11      } // end main
12 } // end class
```

# Self-Check Exercises (2)

- A milk carton can hold 3.78 liters of milk. Each morning, a dairy farm ships cartons of milk to a local grocery store. The cost of producing one liter of milk is $0.38, and the profit of each carton of milk is $0.27. Write a program that does the following:
  - Prompts the user to enter the total amount of milk produced in the morning.
  - Outputs the number of milk cartons needed to hold milk (Round your answer to the nearest integer).
  - Outputs the cost of producing milk.
  - Output the profit for producing milk.