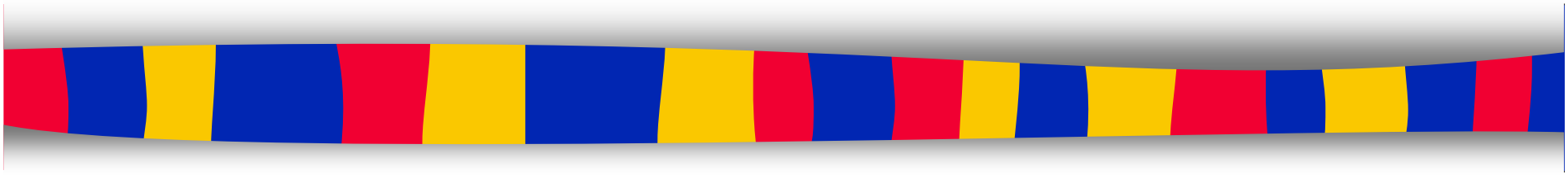
# COMP-248
## Object Oriented Programming I



## Lecture 03:
## Java Fundamentals 2

Emad Shihab, PhD



Parts of the slides are taken from Prof. L. Kosseim

# Last class

1. **Comments**
2. **Identifiers**
3. **Indentation**
4. **Primitive Types**
5. **Variables**
6. Output & Input
7. Assignment
8. Arithmetic Expressions
9. (later) More Assignment Operators
10. (later) Assignment Compatibility
11. Strings

# 6- Output & Input (chap. 2)

**`System.out.print`**
Displays what is in parenthesis

**`System.out.println`**
Displays what is in parenthesis
Advances to the next line

Examples:

```
System.out.print("hello");
System.out.print("you");

System.out.println("hello");
System.out.println("you");

System.out.println();

int price = 50;
System.out.print(price);

char initial = 'L';
System.out.println(initial);
```

```
helloyouhello
you

50L
```

Output

# Multiple output

```
System.out.println("hello" + "you");
double price = 9.99;
int nbItems = 5;
System.out.println("total = " + price*nbItems + "$");
```

**???**

helloyou
total = 49.95$

Output

**`print`** and **`println`**, + is the concatenation…
you need parenthesis for the + to be addition

```
int x = 1, y = 2;
System.out.println("x+y="+x+y);
System.out.println("x+y="+(x+y));
```

**???**

x+y=12
x+y=3

Output

cannot *cut* a string over several lines

```
System.out.println("this is a
                    long string"); // error!
System.out.println("this is a" +
                    "long string"); // ok
```

# Escape sequences (p. 42)

```
System.out.println ("I said "Hi" to her.");
```

**???**

Output

To print a double quote character

- Use an ***escape sequence***
  - sequence is a series of characters that represents a special character
  - begins with a backslash character (\)
  - considered as 1 single character

```
System.out.println ("I said \"Hi\" to her.");
```

**???** I said "Hi" to her.

Output

# Escape sequences

- Some Java escape sequences:

| Escape Sequence | Meaning |
|---|---|
| `\b` | backspace |
| `\t` | tab |
| `\n` | newline |
| `\"` | double quote |
| `\'` | single quote |
| `\\` | backslash |

# Just checking...

- What will the following statement output?

```
System.out.print("one\ntwo\nthree\n");
```

a)one two three

b)one\ntwo\nthree\n

c)"one\ntwo\nthree\n“

d) one
   two
   three

e)onetwothree

# Just checking...

- What statement will result in the following output?

```
Read the file "c:\windows\readme.txt"
```

```
System.out.print
a) ("Read the file "c:\windows\readme.txt");
b) ("Read the file "c:\windows\readme.txt"");
c) ("Read the file "c:\\windows\\readme.txt");
d) ("Read the file \"c:\\windows\\readme.txt\"");
e) ("Read the file \"c:\windows\readme.txt\"");
```

# Console Input (p. 76)

Since Java 5.0, use the **`Scanner`** class

The keyboard is represented by the **`System.in`** object

```
import java.util.Scanner;

public class MyProgram
{
   public static void main (String[] args)
   {
      Scanner myKeyboard = new Scanner(System.in);
      …
      String name = myKeyboard.next();
      int age = myKeyboard.nextInt();
      …
```

1. Create an object of class Scanner
2. Reads one **word** from the keyboard
3. Reads an integer from the keyboard

# To read from a Scanner

To read *tokens*, use a *nextSomething()* method

- **`nextBoolean()`,**
- **`nextByte`,**
- **`nextInt()`,**
- **`nextFloat()`,**
- **`nextDouble()`,**
- **`next()`,**
- **`nextLine()`**
- **…**

tokens are delimited by whitespaces
(ie blank spaces, tabs, and line breaks)

Note: no **`nextChar()`**

```
import java.util.Scanner;
…
Scanner myKeyboard = new Scanner(System.in);
System.out.println("Your name:");
String name = myKeyboard.next();
System.out.println("Welcome " + name + " Enter your age:");
int age = myKeyboard.nextInt();
```

# Example: ScannerDemo.java

```
//*********************************************************
// Author: W. Savitch (modified by L. Kosseim)
//
//  This program demonstrates how to read tokens from
//  the console with the Scanner class
//*********************************************************

import java.util.Scanner; // we need to import this class

public class ScannerDemo
{
   public static void main(String[] args)
   {
      // let's declare our scanner
      Scanner keyboard = new Scanner(System.in);
```

# Example: ScannerDemo.java

```
   // let's ask the user for some input
   System.out.println("Enter the number of pods followed by");
   System.out.println("the number of peas in a pod:");

   // let's read the user input
   int numberOfPods = keyboard.nextInt();
   int peasPerPod = keyboard.nextInt();

   // let’s do some calculations
   int totalNumberOfPeas = numberOfPods*peasPerPod;

  // let's display some output
   System.out.print(numberOfPods + " pods and ");
   System.out.println(peasPerPod + " peas per pod.");
   System.out.println("The total number of peas = " + totalNumberOfPeas);
  }
}
```

# A note about `nextLine`

`nextLine` reads the remainder of a line of text starting where the last reading left off

This can cause problems when combining it with different methods for reading from the keyboard such as `nextInt`

ex:

```
Scanner keyboard = new Scanner(System.in);
int n = keyboard.nextInt();
String s1 = keyboard.nextLine();
String s2 = keyboard.nextLine();
```

input:

```
2
Heads are better than
1 head.
```

what are the values of `n`, `s1`, and `s2`?

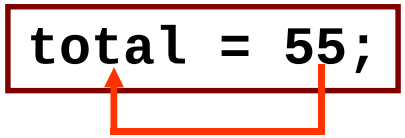
need an extra invocation of `nextLine` to get rid of the end of line character after the 2

# Today, we will see:

1. Comments
2. Identifiers
3. Indentation
4. Primitive Types
5. Variables
6. Output & Input
7. **Assignment**
8. Arithmetic Expressions
9. (later) More Assignment Operators
10. (later) Assignment Compatibility
11. Strings

# 7- Assignment (p. 16)

- Used to change the value of a variable
- The assignment operator is the **=** sign




- Syntax: Variable = Expression;

- Semantics:
  - the expression on the right is evaluated
  - the result is stored in the variable on the left (overwrite any previous value)
  - The assignment expression is worth the entire value of the RHS

# Example

```
public class Geometry
{
   //  Prints the number of sides of several geometric shapes.
   public static void main (String[] args)
   {
      int sides = 7;  // declaration with initialization
      System.out.println("A heptagon has " + sides + " sides.");

      sides = 10;  // assignment statement
      System.out.println("A decagon has " + sides + " sides.");

      sides = 10+2;
      System.out.println("A dodecagon has " + sides + " sides.");
   }
}
```

filename???

**???**

A heptagon has 7 sides.
A decagon has 10 sides.
A dodecagon has 12 sides.

Output

# Difference with the math =

- In Java, = is an operator
- In math, = is an equality relation

In math… a+6 = 10  ok
In Java…  a+6 = 10;

In math… a = a+1  always false
In Java… a = a+1;

In math… a = b and b = a  same thing!
In Java… a = b;  and b = a;

# Examples

- Declarations:

```
int x;
int y = 10;
char c1 = 'a';
char c2 = 'b';
```

- Statements:

```
x = 20+5;
y = x;
c1 = 'x';
c2 = c1;
```

# Just checking...

To swap the content of 2 variables…

```
int x = 10;
int y = 20;
```

```
x = y; y = x;
```

```
y = x; x = y;
```

Both A) & B) will work

Neither A) nor B) will work

# Today, we will see:

1. Comments
2. Identifiers
3. Indentation
4. Primitive Types
5. Variables
6. Output & Input
7. Assignment
8. **Arithmetic Expressions**
9. (later) More Assignment Operators
10. (later) Assignment Compatibility
11. Strings

# 8- Arithmetic Expressions

- An expression is a combination of one or more operands and their operators
- Arithmetic operators:

| | |
|---|---|
| Addition | + |
| Subtraction | - |
| Multiplication | * |
| Division | / |
| Remainder | % |

# Division and Remainder

- The division operator (/) can be:
  - Integer division if both operands are integers

```
10 / 8        equals?   1
8 / 12        equals?   0
```

  - Real division

```
10.0 / 8      equals?   1.25
8 / 12.0      equals?   0.6667
```

- The remainder operator (%)

  returns the remainder after the integer division

```
10 % 8        equals?   2    (10÷8 = 1 remainder 2)
8 % 12        equals?
```

# Let's put it all together

- Purpose: Convert Fahrenheit to Celsius
- Algorithm:

  Ask and read the temperature in Fahrenheit

  Calculate the temperature in Celsius (1 Celsius = (Fahr – 32) * 5/9)
- Display temperature in Celsius

- Variables and constants:

| Data | Identifier | Type | var or const? |
|---|---|---|---|
| Temperature in Fahrenheit | `tempInF` | `double` | |
| Temp in Celsius | `tempInC` | | |
| Celsius conversion rate | | | |
| | | | |

# The Java program

```
//*************************************************************
//  Temperature.java       Author: your name
//  A program to convert degrees Fahrenheit in degrees Celsius.
//*************************************************************

public class Temperature {
   public static void main (String[] args)
   {
     // Declaration of variables and constants
          double tempInF=0;
          double tempInC=0;
          final double CELSIUS_CONST = 5/9;

    // step 1: Ask and read the temperature in Fahrenheit
          Scanner myKeyboard = new Scanner (System.in);
          System.out.println("Enter temp in F:");
          tempInF = myKeyboard.nextDouble();
          myKeyboard.close();

     // step2: Calculate the temperature in Celsius
          tempInC = (tempInF - 32)*CELSIUS_CONST;

     // step3: Display temperature in Celsius
          System.out.println("Temp in C is: " + tempInC);
```

filename???

# Today, we will see:

1. Comments
2. Identifiers
3. Indentation
4. Primitive Types
5. Variables
6. Output & Input
7. Assignment
8. Arithmetic Expressions
9. (later) More Assignment Operators
10. (later) Assignment Compatibility
11. **Strings**

# 11- Strings (p. 33)

- So far we have seen only primitive types

- A variable can be either:
  - a primitive type (ex: **`int`**, **`float`**, **`boolean`**, …)
  - or a reference to an object (ex: String, Array, …)

- A character string:
  - is an object defined by the **`String`** class
  - delimited by double quotation marks ex: "hello", "a"

```
System.out.print("hello"); // string of characters

System.out.print('a');     // single character
```

# Declaring Strings

- Declare a reference to a String object

```
String title;
```

- Declare the object itself (the String itself)

```
title = new String("content of the string");
```

This calls the String *constructor*, which is a special method that sets up the object

# Declaring Strings

- Because strings are so common, we don't have to use the **new** operator to create a **String** object

```
String title;
title = new String("content of the string");
```

```
String title = new String("content of the string");
```

```
String title;
title = "content of the string";
```

```
String title = "content of the string";
```

- These special syntax works only for strings

# Strings

- Once a string is created, its value cannot be modified (i.e., the object is immutable)
  - cannot lengthen/shorten it
  - cannot modify its content

- The String class offers:
  - the + operator (string concatenation)
    - ex: `String solution = "The answer is " + "yes";`
  - many methods to manipulate strings

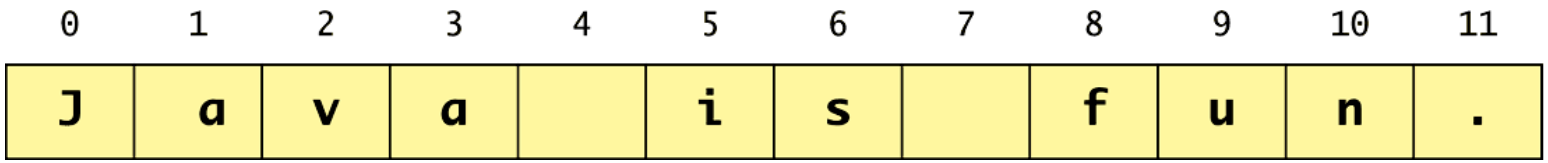
```
length()                    // returns the nb of characters in a string
concat(String str)          // returns the concatenation of the string and str
toUpperCase()               // returns the string all in uppercase
replace(char oldChar, char newChar)  // returns a new string
                            // where all occurrences of oldChar have been replaced by newChar
```

... see p. 38

# String indexes start at zero

**Display 1.5 String Indexes**

The 12 characters in the string `"Java is fun."` have indexes `0` through `11`.




*Notice that the blanks and the period count as characters in the string.*

# Example

```
public class StringTest  {
    public static void main (String[] args)  {
      String string1 = new String ("A string");
      String string2 = "";
      String string3, string4, string5;

      System.out.println("Content of string1: \"" + string1 + "\"");
      System.out.println("Length of string1: " + string1.length());
      System.out.println("Content of string2: \"" + string2 + "\"");
      System.out.println("Length of string2: " + string2.length());

      string2 = string1.concat(" abc");
      string3 = string2.toUpperCase();
      string4 = string3.replace('A', 'X');
      string5 = string4.substring(3, 10);

      System.out.println(string2);
      System.out.println(string3);
      System.out.println(string4);
      System.out.println(string5);
   }
}
```

**???**

Content of string1: "A string"

Length of string1: 8

Content of string2: ""

Length of string2: 0

A string abc

A STRING ABC

X STRING XBC

TRING X

Output

# Question

- What does the following code produce, given the input String:

  This is a test

```
….
String inputLine;

Scanner myKeyboard = new Scanner (System.in);
System.out.println("Enter a line of text: ");
inputLine = myKeyboard.next();
myKeyboard.close();

System.out.println("The text you enetered is: " + inputLine);
…
```

# Next class

1. Comments
2. Identifiers
3. Indentation
4. Primitive Types
5. Variables
6. Output & Input
7. Assignment
8. Arithmetic Expressions
9. (later) **More Assignment Operators**
10. (later) Assignment Compatibility
11. Strings