جامعة أم القرى
UMM AL-QURA UNIVERSITY

CS1701T: Computer Programming in Java

1st Semester 2026

Lecture 2

Algorithms and Introduction to Java (Part2)

Topics covered:

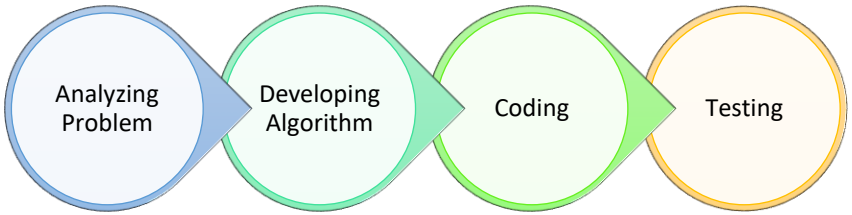
- The concept of algorithms for solving simple problems
- Testing, and debugging
- Documentation and program style

What Is Programming?

- Programming is a process of problem solving.
- We could think of programming as the process of breaking down a large, complex task into smaller and smaller subtasks.
- The process continues until the subtasks are simple enough to be performed with the electronic circuits provided by the hardware.
- The person who writes a program is called the programmer.
- The person who interacts with the program is called the user.

Problem Solving & Algorithm

- Problem solving is the process of identifying a problem, developing an algorithm, and eventually implementing a computer program

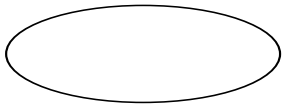
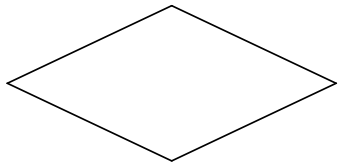


- An algorithm is a set of finite, sequential, and unambiguous steps usually described in natural language to solve a given problem.

- **Two most common methods to represent algorithms are:**
  - Flowcharts is a graphical representation of an algorithm
  - Pseudocode is a description of the instructions that human could understand and so the computer could execute later on

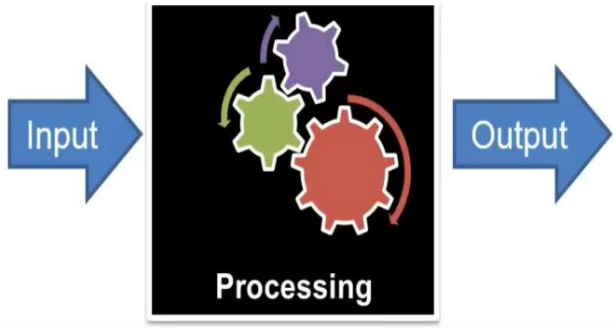
Problem Solving & Algorithm (cont.)

Flowchart Symbols

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the program

Problem Solving & Algorithm (cont.)

Problem Solving Steps:



Step1: Analyze the problem

- Understand the overall problem
- Define problem requirements
  - Does program require user interaction? If yes, What is the Input?
  - What is the expected Output?
- Design steps (algorithm) to solve the problem

Problem Solving & Algorithm (cont.)

Step2: Implement the algorithm

- Implement the algorithm in code (**coding**)
- Verify that the algorithm works (**testing**)

Step3: Maintenance

- Use and modify the program if the problem domain changes. (any changes required)

Note: If the problem is complex, divide it into sub problems and then analyze each sub-problem as above

Problem Solving & Algorithm (cont.)

Example 1:

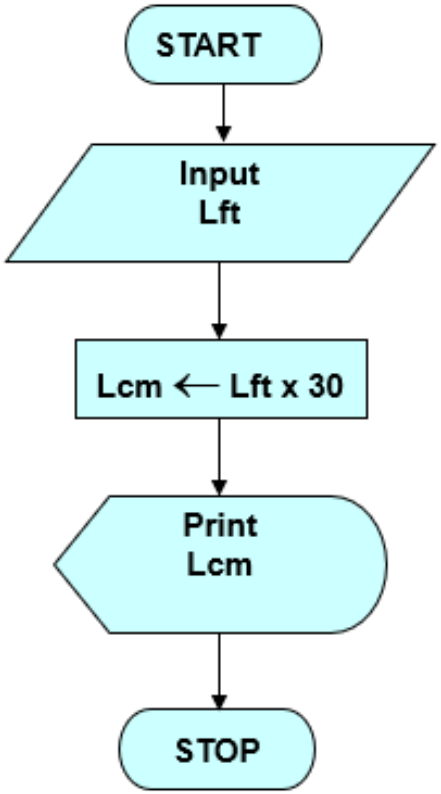
- Write a Pseudocode and draw a flowchart to convert the length from feet to centimeter.

Pseudocode:

- Input the length in feet (Lft)
- Calculate the length in cm (Lcm) by multiplying LFT with 30
- Print length in cm (Lcm)

Problem Solving & Algorithm (cont.)

Example 1 Flowchart:



Problem Solving & Algorithm (cont.)

Example 2:

- Write a Pseudocode and draw a flowchart to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks

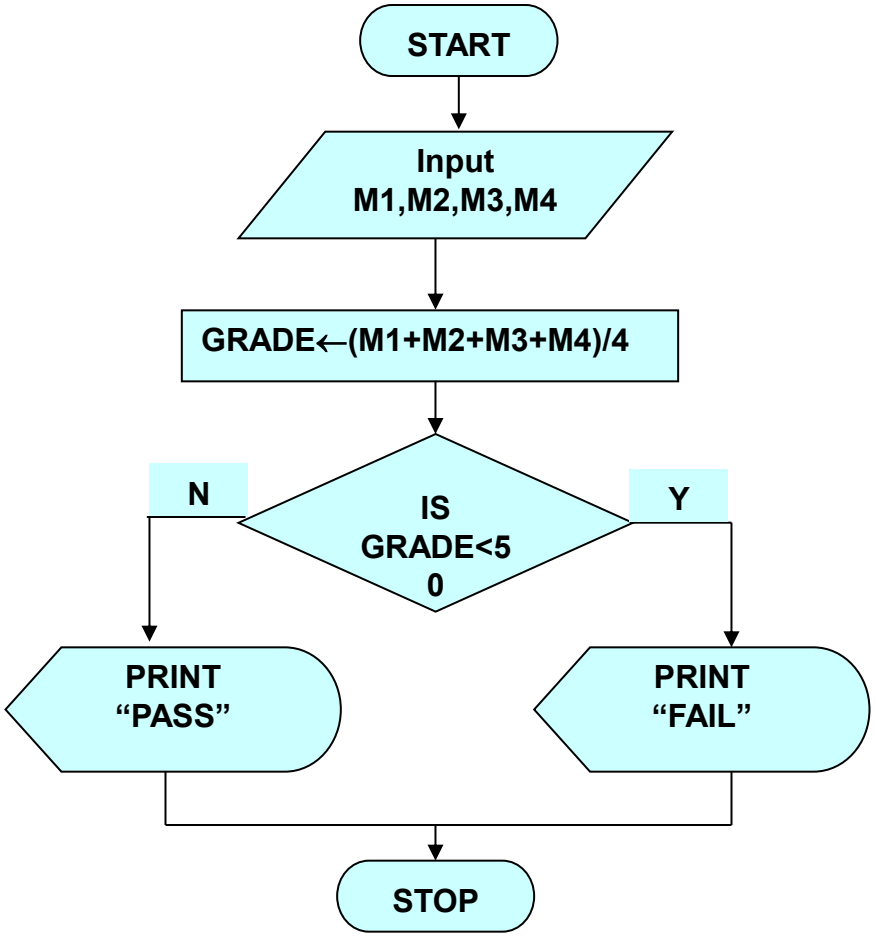
Pseudocode:

- Input a set of 4 marks
- Calculate their average by summing and dividing by 4
- if average is below 50, Print "FAIL"

  Else Print "PASS"

Problem Solving & Algorithm (cont.)

Example 2 Flowchart:



Programming Paradigms

- **Procedural/Functional paradigm:**
  - In which programs consist of a collection of procedures and functions that operate on data.
  - Example: Fortran, Pascal, and C

- **The object-oriented paradigm:**
  - In which programs are viewed instead as a collection of "**objects**" for which the data and the operations acting on that data are encapsulated into integrated units.
  - Example: Smalltalk, C++, and Java.

Programming Basics

Reserved Keywords

- **Reserved words** or **keywords** are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. **Reserved words** are always spelled with all lowercase letters.

- For example, when the compiler sees the word **class**, it understands that the word after class is the name for the class.

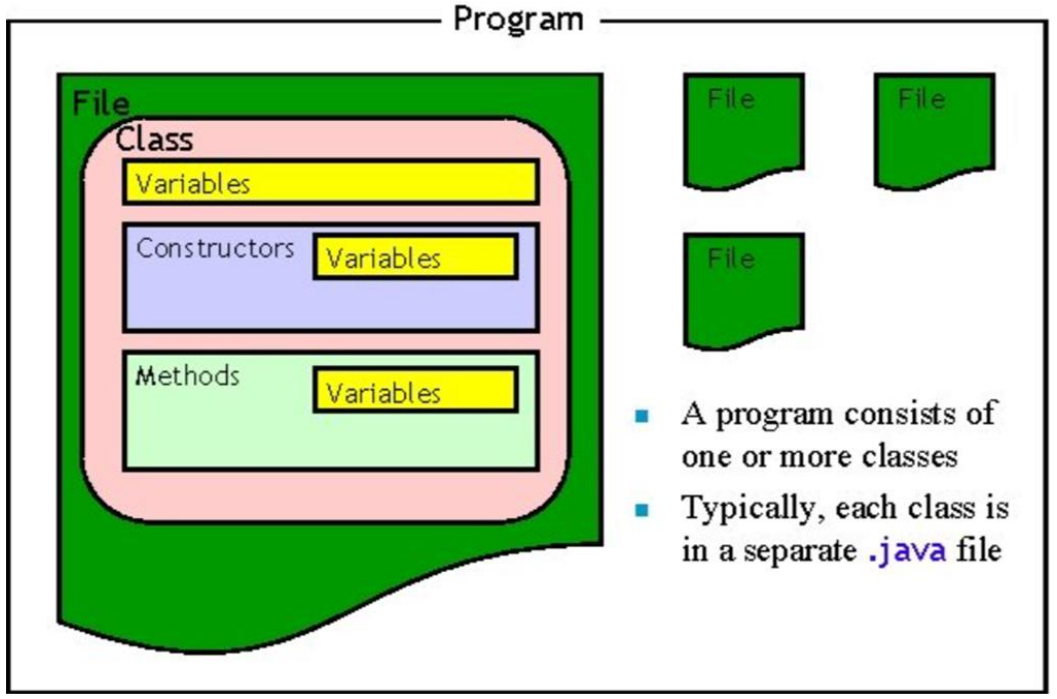
- We must use reserved words only for their intended purpose. For example, we can't use the word **class** for any other purpose than defining a class

Reserved Keywords (Cont.)

```
abstract    else        interface   switch
assert      enum        long        synchronized
boolean     extends     native      this
break       false       new         throw
byte        final       null        throws
case        finally     package     transient
catch       float       private     true
char        for         protected   try
class       goto        public      void
const       if          return      volatile
continue    implements  short       while
default     import      static
do          instanceof  strictfp
double      int         super
```

Compiling and Running a Java Program or Class

- A Java program consists of one or more classes, which must be compiled before running the program.
- Each class should be in a separate file.



Compiling and Running a Java Program or Class (Cont.)

- The name of the file should be the same as the name of the class.

- The class will contain the statement below (only ONE class must include the main method)

```
public static void main(String[] args)
```

Testing and Debugging

- The best way to write a correct program is to carefully design the necessary objects and the algorithms for the objects' methods.
- Then you carefully translate everything into a programming language such as Java.
- So, to eliminate errors is to avoid them in the first place
- Test your program with appropriate test cases (some where the answer is known), discover and fix any errors, then retest.

Programming Errors

- A **mistake** in a program is called a **bug**.

- The process of eliminating errors is called **debugging**.

- Three kinds of errors:
  - Syntax errors
  - Runtime errors
  - Logic errors

Programming Errors (Cont.)

Syntax Errors:

- The **syntax** of a programming language is the set of grammatical rules for the correct way to write a program.
- Thus, syntax errors are the grammatical mistakes in a program
- The **compiler** catches syntax errors and prints an error message.
- Example: missing semicolon , unclosed quotation mark, ..

```
1  public class Main
2  {
3      public static void main(String[] args) {
4          System.out.println("Hello World")
5      }
6  }
```

```
1  public class Main
2  {
3      public static void main(String[] args) {
4          System.out.println("Hello World);
5      }
6  }
```

Programming Errors (Cont.)

Runtime Errors:

- Errors that are detected when your program is running, but not during compilation
- When the computer detects an error, it terminates the program and prints an error message.
- The error message might not be easy to understand, but at least you will know that something is wrong!
- Example: attempting to divide by 0.

```
1  public class Main
2  {
3      public static void main(String[] args) {
4          System.out.println(1/0);
5      }
6  }
```

Programming Errors (Cont.)

Logic Errors:

- Errors that are not detected during compilation or while running, but which cause the program to produce incorrect results.

- Logic errors will not give you any error messages. For this reason, logic errors are the hardest kind of error to locate.

- Example: an attempt to calculate a Fahrenheit temperature from a Celsius temperature by multiplying by 1.8 and adding 23 instead of 32.

$$°F = °C \times 1.8 + 32$$

```
1  public class Main
2  {
3      public static void main(String[] args) {
4          System.out.print("Celsius 35 in Fahrenheit degree is: ");
5          System.out.print(35*1.8+23);
6      }
7  }
```

Documentation and Program Style

Basic Java Program Structure

- There are 5 components that we usually include in a Java program, which are:

```
(1)  [ Documentation Statement(s) ]
(2)  [ Package Statement ]
(3)  [ Import Statement(s) ]
(4)  public class className {
(5)      public static void main(String[] args) {
             [ Body of Main Function ]
         }
     }
```

Basic Java Program Structure (Cont.)

(1) Documentation Statement(s):

- It includes "**comments**" lines that we use to describe the whole idea behind the program and its methods
- The **compiler** ignores these comments during the time of execution.
- It helps other programmers to easily understand your code!

Basic Java Program Structure (Cont.)

- **Documentation Statement(s):**
  - There are three types of comments that Java supports:-

Single line Comment	Multi-line Comment	Javadoc Comment
• begins with // • Everything after these symbols and to the end of the line is treated as a comment and is ignored by the compiler. • Example: double radius; //in cm	• A multi-lines comment can begin with /* and end with */ • Everything between these symbols is treated as a comment and is ignored by the compiler. • Example: /* This program should only be used on math problems */	-

Basic Java Program Structure (Cont.)

(2) Package Statement(s):

- A package is a library or container of a group of related classes (think of it as a folder in a file directory) that could be user-defined or have been defined by java already (built-in).
- They are **optional** and used to declare classes in a collection called package
- **Only one package** statement is allowed in a program
- Package statements must be **at the beginning of the code** before any class or interface declaration!
- Syntax: `package name;`
- Example: `package people;`

Note, this statement declares that all the classes and interfaces defined in this source file are a part of the **people** package

Basic Java Program Structure (Cont.)

(3) Import Statement(s):

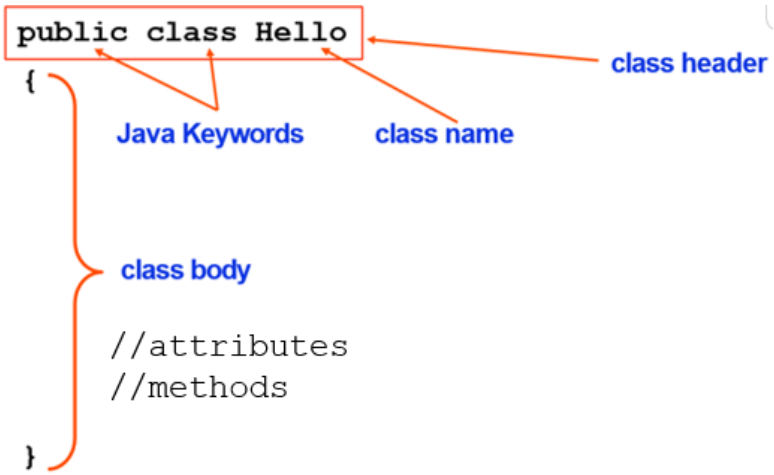
- They are used to refer to set of classes that are located in other packages
- Always write the import statements **after the package** statement and **before the class declaration**.
- Syntax: `import packageName.className;`
- Example: `import java.util.Date; //imports the date class`

Basic Java Program Structure (Cont.)

(4) Class Definition(s):

- A **class** is a collection of variables and methods that operate on the attributes or fields.

- Syntax:



- The **public** keyword means this class is accessible by any other classes

- The **curly brackets** are used to make sure that the statements belong to one class

Basic Java Program Structure (Cont.)

- **Class Definition(s):**
  - **Class declaration (header)**
    - Every Java program consists of **at least one class** that you define
    - **class** keyword introduces a class declaration and is immediately followed by the **class name**
    - Java class declaration normally contain **one or more methods**.
    - One of these classes must have the **main method**
  - **Class name**
    - By convention, **begin with a capital letter**
    - **Camel Case** is a style used to capitalizing the initial letter of each word and omitting spaces (e.g., SampleClassName).
    - A class name is an identifier—a series of characters consisting of letters, digits, underscores (_) and dollar signs ($) that does not begin with a digit and does not contain spaces
    - Java is **case sensitive**—uppercase and lowercase letters are distinct

Basic Java Program Structure (Cont.)

(5) Main Method:

- The **main method** is the place where the program execution starts. Meaning the computer follows the order specified by the Java statements.

- The main method is an essential part of any program in Java and where the **JVM** starts running the program

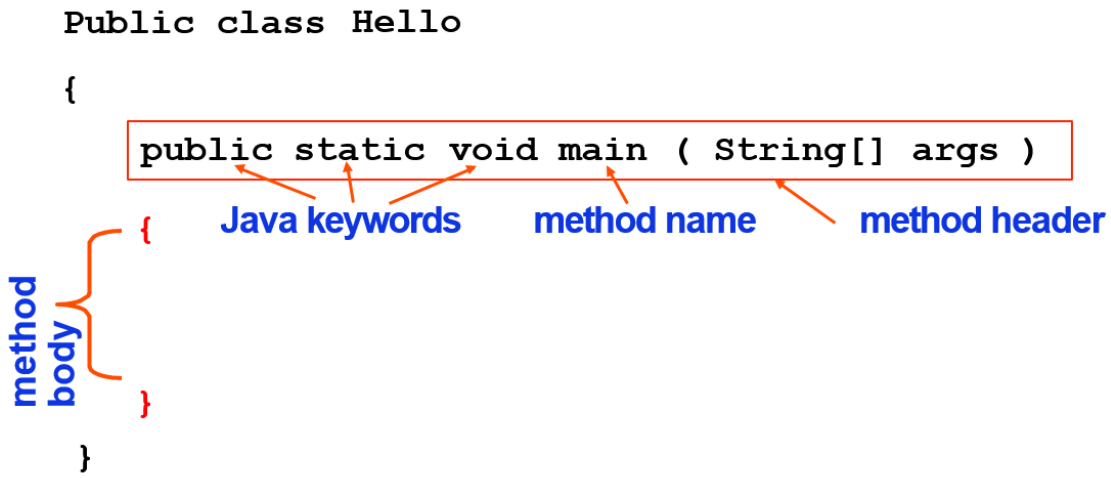
- The main method is declared **inside the class definition**

- **Where**:
  - **public** means that this method can be used outside of this class
  - **static** means that there is no need to create an object to access thiss method
  - **void** means that this method does not return any value
  - **String[] args** is an array named as args where each element is a string. We use it to pass the input to the program when we run the Java code through a console

- Declaring main as **static** allows the **JVM** to invoke main without creating an instance of the class

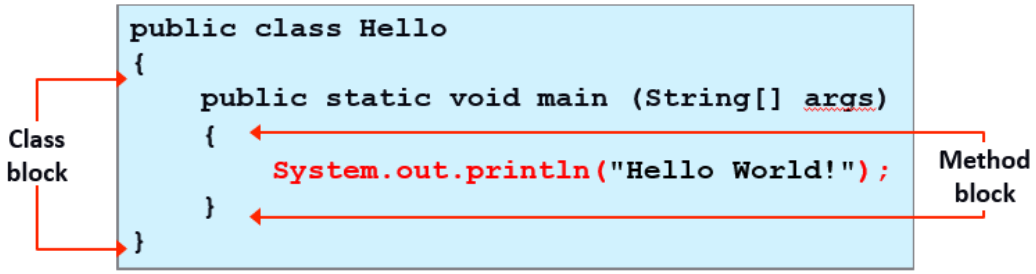
Basic Java Program Structure (Cont.)

- **Main Method:**
  - Syntax:



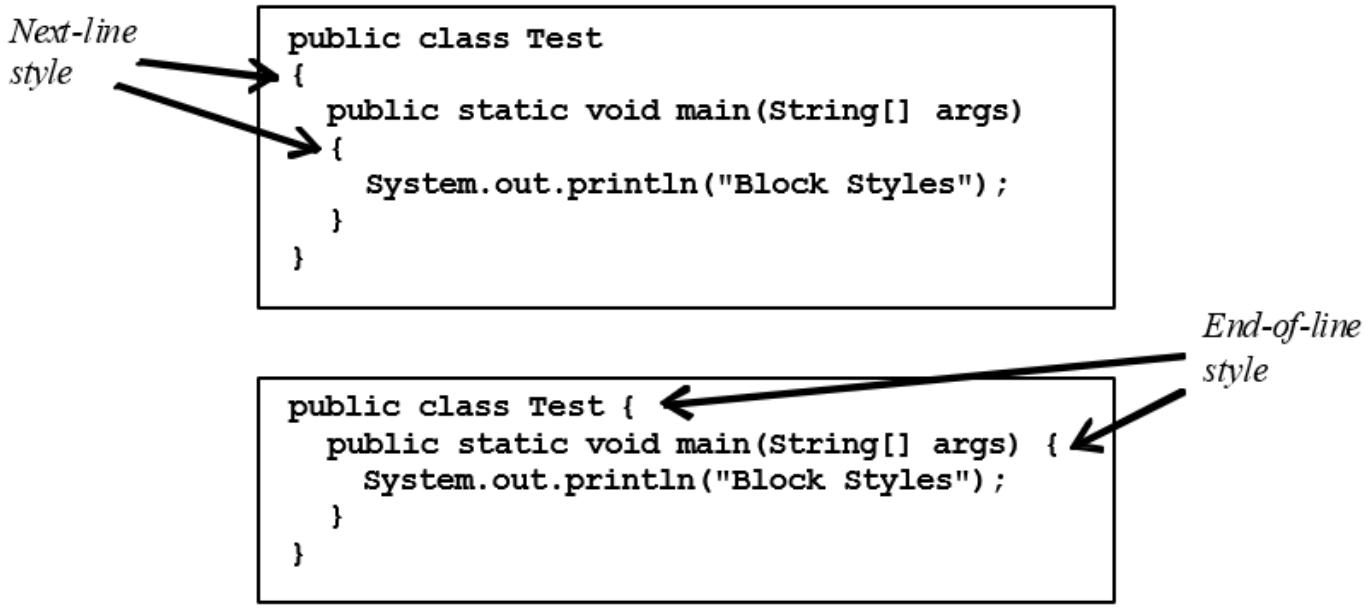
Basic Java Program Structure (Cont.)

- **Block:**
  - It is a pair of curly braces in a program forms a block that groups components of a program
  - Examples: Class block, method block, statement block, ...
  - A left brace **{** begins the body of every class declaration
  - A corresponding right brace **}** must end each class declaration
  - It is a syntax error if braces do not occur in the matching pairs
  - Example:



Basic Java Program Structure (Cont.)

- **Block:**
  - You may use **next-line** or **end-of-line** style for braces
  - You should **stick with one style** in the whole code



Basic Java Program Structure (Cont.)

- **Block:**
  - **Which code do you think it is more readable?**

Code Sample 1:

```
public class Test1 {
     public static void main ( String[] args )
          {  System.out.println("On a withered branch");
          System.out.println("A crow has just
          alighted:");  System.out.println("Nightfall in
          autumn.");
     }
}
```

Code Sample 2:

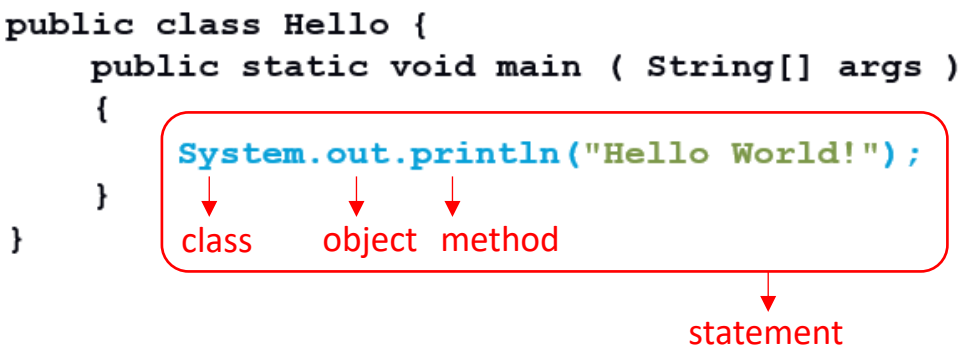
```
public          class
              Test1 {
     public       static void main ( String[] args )
          {         System.out.println(    "On a withered
          branch");
          System.   out.println("A crow has just
          alighted:");       System.out.println("Nightfall
               in autumn.");
     }}
```

Basic Java Program Structure (Cont.)

- **Statement:**
  - Is a command for the computer to do something.
  - **MOST statements** in Java should be followed by a semicolon (;)
  - Methods are built out of statements.
  - The statements in a method are placed between braces { and }
  - There are several types of statements in Java. For example, **declaration, assignment, control, and loop statements**.

Basic Java Program Structure (Cont.)

- **Statement:**
  - **Example Explanation:**



- Instructs the computer to display a string of characters contained between the double quotation marks on the screen
- **White-space characters** in strings **are not ignored** by the compiler
- The string in the parentheses is the argument to the method
- Positions the output cursor at the beginning of the next line on the screen (command window)

Basic Java Program Structure (Cont.)

- **Statement:**
  - **Example Explanation:**
    - Strings cannot span multiple lines of code

```
System.out.println("Hello
                   World!");
```

A First Java Application

LISTING 1.1 A Sample Java Program

```
import java.util.Scanner;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in);
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();



        System.out.println("The sum of those two numbers is");
        System.out.println(n1 + n2);
    }
}
```

Gets the `Scanner` *class from the package (library)* `java.util`

Name of the class—your choice. "This program should be in a file named `FirstProgram.java`*"*

Sends output to screen

Says that `n1` *and* `n2` *are variables that hold integers (whole numbers)*

Readies the program for keyboard input

Reads one whole number from the keyboard

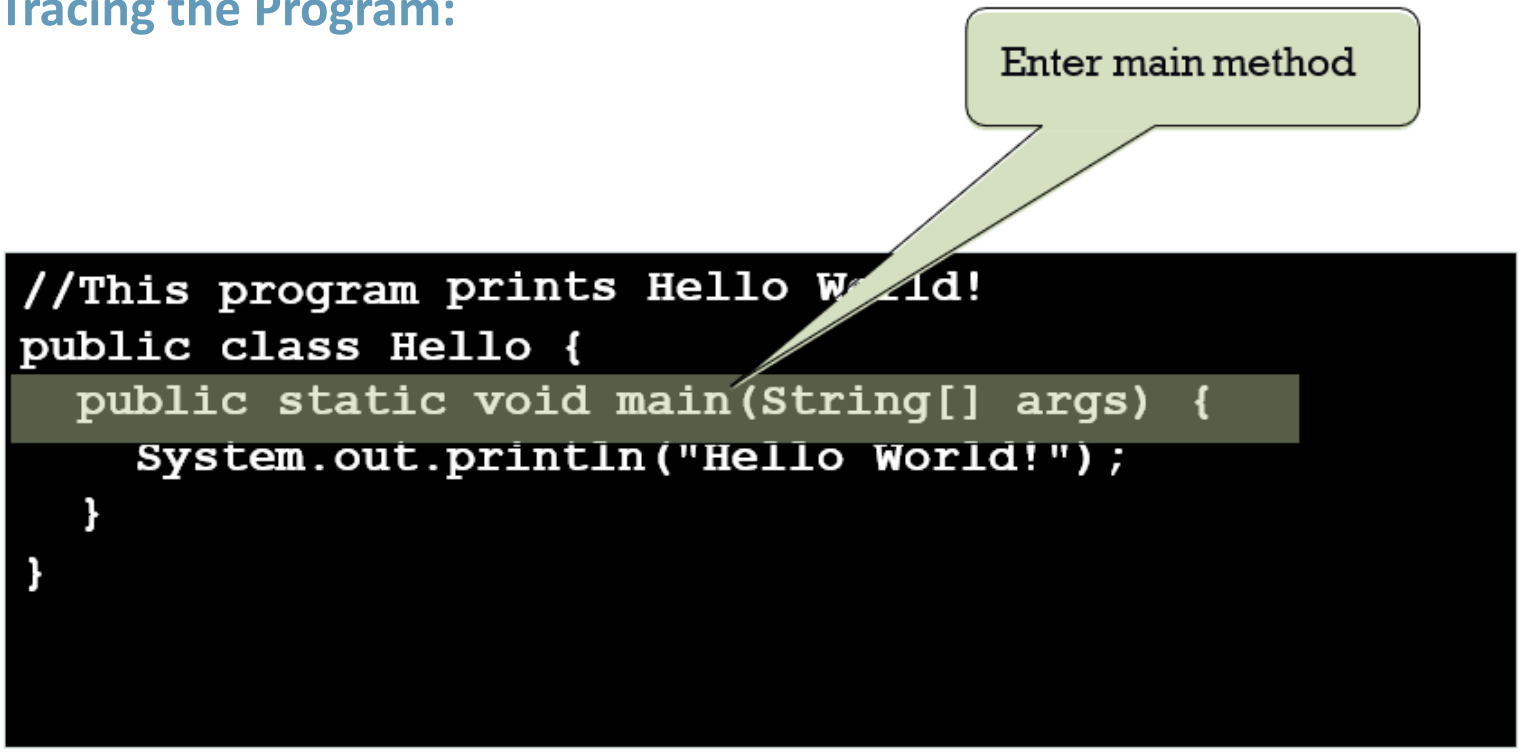
A First Java Application (Cont.)

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

Sample screen output

A First Java Application (Cont.)

- Tracing the Program:



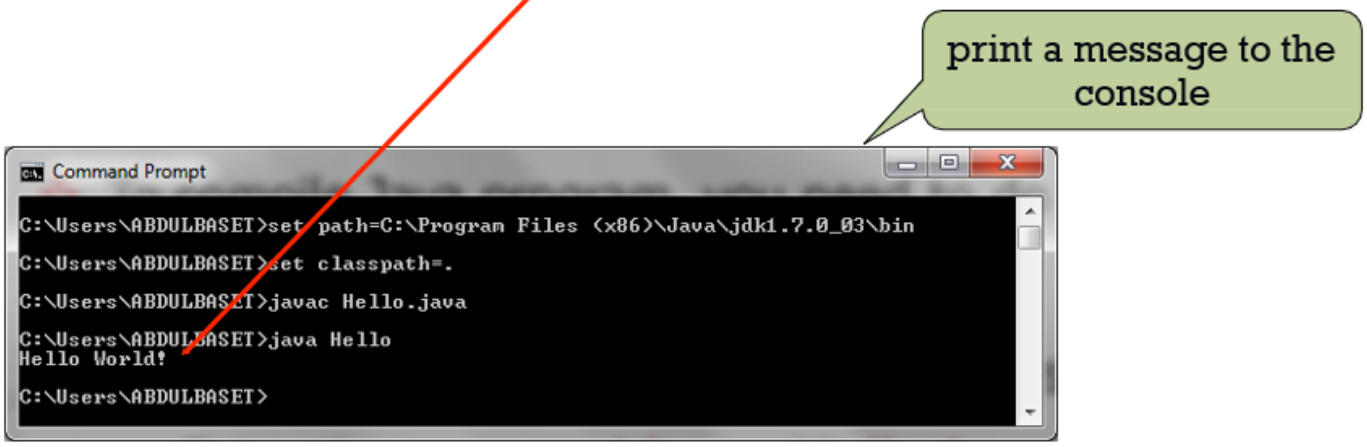
A First Java Application (Cont.)

- Tracing the Program:

A First Java Application (Cont.)

- **Tracing the Program:**

```
//This program prints Hello World!
public class Hello {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```



Recap

- **Proper Indentation and Spacing:**
  - Use **blank lines** to separate segments of the code .
  - Use **space characters** and **code indentations** to enhance program readability
  - Write each statement in a separate line , allowing end of line comment.
  - For example:

```
System.out.println("Hello World!");  // print a string
```

Recap (Cont.)

- **Appropriate Comments:**
  - Starts your program with a comment that states the purpose of the program, the author, time and date of the last program modifications
  - Write a comment before each class, documenting the purpose of the class
  - Write a comment before each method, documenting the purpose of the method
  - Write an end of line (single line) comment, documenting the purpose of the statement

Recap (Cont.)

- In every Java source file, you might see the following 5 components:

```
//comments
package name;

import className;

public class className{
      //attributes
      //methods
      public static void main (String[]args){
      //statements
      }
}
```

Questions?