Module 02 — Java Fundamentals

ITP 120 — Java Programming I

Northern Virginia Community College
Randy Michak

🎯 Learning Objectives

By the end of this module, you will be able to:

- Identify the parts of a Java program
- Use `System.out.println` and `System.out.print` to display output
- Declare variables and assign values to them
- Work with Java's primitive data types
- Perform arithmetic operations
- Use combined assignment operators
- Declare constants with `final`
- Create `String` objects
- Read keyboard input using `Scanner`
- Write comments to document your code

Parts of a Java Program

Every Java program has a specific structure. Even the simplest program follows rules about how it's organized. Let's look at the classic first program:

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

`// My first Java program` — A comment. Java ignores this line. It's for humans reading your code.

`public class Hello` — Defines a class named `Hello`. Every Java program needs at least one class, and the filename must match: `Hello.java`.

`{ }` — Curly braces mark the beginning and end of a block of code. The outer pair belongs to the class; the inner pair belongs to the method.

`public static void main(String[] args)` — The main method. This is where Java starts running your program. Every Java application must have one.

`System.out.println("Hello, World!");` — Prints text to the screen, then moves the cursor to the next line.

> ⚠️ **File Name Must Match the Class Name**
>
> If your class is named `Hello`, the file must be saved as `Hello.java` — exact spelling, exact capitalization. Java is strict about this.

> 📖 **Key Terms**
>
> - **Class** — A container that holds your program's code. Every Java program starts with a class definition.
> - **Method** — A named block of code that performs a task. The `main` method is the entry point.
> - **Statement** — A single instruction that Java executes. Every statement ends with a semicolon (`;`).

> 🧠 **Think of It Like a Building**
>
> The class is the building itself. The main method is the front door — it's where everyone enters. The statements inside are the instructions written on signs that tell you what to do once you walk in.

Printing Output

Java gives you two ways to print text to the screen:

Method	What It Does
`System.out.println()`	Prints the text, then moves the cursor to the next line
`System.out.print()`	Prints the text but the cursor stays on the same line

Example: println vs. print

```
System.out.println("First line");
System.out.println("Second line");
System.out.print("Stay ");
System.out.print("on same line");
```

```
First line
Second line
Stay on same line
```

Escape Sequences

Sometimes you need to include special characters in your output. Java uses escape sequences — a backslash followed by a character — to represent them:

Escape Sequence	What It Produces
`\n`	New line
`\t`	Tab
`\\`	Backslash
`\"`	Double quote

```
System.out.println("Name:\tJava");
System.out.println("She said \"hello\"");
System.out.println("Line one\nLine two");
```

```
Name:   Java
She said "hello"
Line one
Line two
```

Variables and Data Types

A variable is a named storage location in the computer's memory. Before you can use a variable in Java, you must declare it — tell Java its type and name.

📖 Variable Declaration

The general form:

```
dataType variableName;
```

You can also declare and assign a value at the same time (called initialization):

```
int age = 20;
```

> 🧠 **Variables Are Like Labeled Boxes**
>
> Think of a variable as a box with a label on it. The data type determines what size box you get (how much data it can hold). The name is the label. The value is what you put inside.

Primitive Data Types

Java has eight primitive (built-in) data types. Here are the ones you'll use most often:

Type	What It Stores	Example	Size
`int`	Whole numbers (no decimal)	`42`, `-7`, `0`	4 bytes
`double`	Decimal numbers	`3.14`, `-0.5`	8 bytes
`char`	A single character	`'A'`, `'7'`, `'!'`	2 bytes
`boolean`	`true` or `false`	`true`, `false`	1 bit*
`byte`	Small whole numbers (-128 to 127)	`100`, `-50`	1 byte
`short`	Medium whole numbers	`30000`	2 bytes
`long`	Large whole numbers	`2000000000L`	8 bytes
`float`	Decimal numbers (less precision)	`3.14f`	4 bytes

*The JVM typically uses more memory to store a boolean, but logically it represents a single bit of information.

> 💡 **For Beginners**
>
> Focus on these four for now: `int` for whole numbers, `double` for decimals, `char` for single characters, and `boolean` for true/false. You'll use these in almost every program.

Declaring Variables — Examples

```
int score = 95;
double price = 19.99;
char grade = 'A';
boolean passed = true;

System.out.println("Score: " + score);
System.out.println("Price: $" + price);
System.out.println("Grade: " + grade);
System.out.println("Passed: " + passed);
```

```
Score: 95
Price: $19.99
Grade: A
Passed: true
```

⚠️ **Common Mistakes**

- **Using a variable before declaring it.** Java won't know what `score` is if you haven't declared it first.
- **Forgetting the semicolon.** Every statement in Java ends with `;`. Leave it off and you'll get a compile error.
- **Mixing up `'A'` and `"A"`.** Single quotes are for `char`. Double quotes are for `String`.
- **Storing a decimal in an `int`.** `int price = 19.99;` won't compile. Use `double` instead.

Naming Rules

Java has rules for variable names:

- Must start with a letter, underscore (`_`), or dollar sign (`$`)
- After the first character, you can also use digits
- Cannot contain spaces

- Cannot be a Java reserved word (`int`, `class`, `public`, etc.)
- Case-sensitive — `Score` and `score` are different variables

> 💡 **Naming Convention: camelCase**
>
> Java programmers use camelCase for variable names: start with a lowercase letter, and capitalize the first letter of each additional word. No underscores, no spaces.
>
> ✅ `studentAge`, `totalScore`, `firstName`
> ❌ `student_age`, `TotalScore`, `firstname`

Arithmetic Operators

Java can do math. Here are the arithmetic operators you'll use:

Operator	Operation	Example	Result
`+`	Addition	`5 + 3`	`8`
`-`	Subtraction	`10 - 4`	`6`
`*`	Multiplication	`6 * 7`	`42`
`/`	Division	`15 / 4`	`3` (integer division!)
`%`	Modulus (remainder)	`15 % 4`	`3`

> ⚠️ **Integer Division Drops the Decimal**
>
> When you divide two `int` values, Java throws away the decimal part. It doesn't round — it just chops it off.
>
> ```
> int result = 15 / 4;   // result is 3, not 3.75
> ```
>
> To get the decimal, at least one number needs to be a `double`:
>
> ```
> double result = 15.0 / 4;   // result is 3.75
> ```

Operator Precedence

Java follows the same order of operations you learned in math class:

1. **Parentheses** `()` — evaluated first
2. **Multiplication, Division, Modulus** `* / %` — left to right
3. **Addition, Subtraction** `+ -` — left to right

```
int answer = 2 + 3 * 4;        // answer is 14 (not 20)
int answer2 = (2 + 3) * 4;      // answer2 is 20
```

Combined Assignment Operators

Java provides shortcuts for updating a variable using its current value:

Operator	Equivalent To	Example
`+=`	`x = x + value`	`score += 10;`
`-=`	`x = x - value`	`lives -= 1;`
`*=`	`x = x * value`	`price *= 2;`

Operator	Equivalent To	Example
`/=`	`x = x / value`	`total /= 3;`
`%=`	`x = x % value`	`num %= 5;`

```
int score = 100;
score += 25;    // score is now 125
score -= 10;    // score is now 115
score *= 2;     // score is now 230
System.out.println("Final score: " + score);
```

```
Final score: 230
```

Constants

A constant is a variable whose value cannot change after it's been assigned. In Java, you declare a constant using the `final` keyword:

```
final double TAX_RATE = 0.075;
final int MAX_ATTEMPTS = 3;
```

> 💡 **Constants Use ALL_CAPS**
>
> By convention, constant names are written in all uppercase with underscores between words: `TAX_RATE`, `MAX_SIZE`, `PI`. This makes them stand out from regular variables.

```
final double TAX_RATE = 0.075;
double subtotal = 50.00;
double tax = subtotal * TAX_RATE;
double total = subtotal + tax;

System.out.println("Subtotal: $" + subtotal);
System.out.println("Tax:      $" + tax);
System.out.println("Total:    $" + total);
```

```
Subtotal: $50.0
Tax:      $3.75
Total:    $53.75
```

⚠️ You Cannot Reassign a Constant

If you try to change a `final` variable after assigning it, Java will give you a compile error:

```
final int MAX = 10;
MAX = 20;  // ERROR: cannot assign a value to final variable
```

The String Class

A `String` is a sequence of characters — a word, a sentence, or any text. Unlike the primitive types, `String` is a class, so it starts with a capital S.

```
String firstName = "Alice";
String lastName = "Smith";
String fullName = firstName + " " + lastName;
System.out.println("Hello, " + fullName + "!");
```

```
Hello, Alice Smith!
```

The `+` operator joins (concatenates) strings together.

📖 **String Concatenation**

Concatenation means joining strings end-to-end using the `+` operator. When you use `+` between a string and a number, Java automatically converts the number to text and joins them.

```
int age = 20;
System.out.println("Age: " + age);  // prints: Age: 20
```

Reading Input with Scanner

Most programs need to get information from the user. Java's `Scanner` class reads input from the keyboard.

Three Steps to Use Scanner

Step 1: Import it at the top of your file (before the class):

```
import java.util.Scanner;
```

Step 2: Create a Scanner object inside your method:

```
Scanner keyboard = new Scanner(System.in);
```

Step 3: Use it to read different types of input:

Method	What It Reads
`nextInt()`	An `int` value
`nextDouble()`	A `double` value
`nextLine()`	An entire line of text (`String`)

Method	What It Reads
`next()`	A single word (`String` , up to the first space)

Complete Example: Getting User Input

```
import java.util.Scanner;

public class Greeting {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = keyboard.nextLine();

        System.out.print("Enter your age: ");
        int age = keyboard.nextInt();

        System.out.println("Hello, " + name + "! You are " + age + "
years old.");
    }
}
```

```
Enter your name: Maria
Enter your age: 19
Hello, Maria! You are 19 years old.
```

🔧 Try It Yourself

Write a program that asks the user for two numbers, then displays their sum, difference, product, and quotient. Use `Scanner` for input and `double` for the variables so the division works correctly.

Comments

Comments are notes in your code that Java ignores. They help other people (and future you) understand what the code does.

Style	Syntax	Use For
Single-line	`// comment here`	Short notes on one line
Multi-line	`/* comment here */`	Longer explanations spanning multiple lines
Javadoc	`/** comment here */`	Documentation for classes and methods (used by tools)

```
// This program calculates a tip
public class TipCalc {
    public static void main(String[] args) {
        double bill = 45.00;      // original bill amount
        double tipRate = 0.20;    // 20% tip
        double tip = bill * tipRate;

        /* Display the results
           to the user */
        System.out.println("Tip: $" + tip);
        System.out.println("Total: $" + (bill + tip));
    }
}
```

> 💡 **When to Comment**
>
> Good comments explain *why* you did something, not *what* the code does. If your code is clear, it speaks for itself. Use comments for tricky logic, important decisions, or anything that might confuse someone reading the code later.

📋 Module Summary

Module 02 Key Takeaways

- Every Java program needs a class and a main method
- The filename must match the class name exactly: `Hello.java` for `public class Hello`
- `System.out.println()` prints with a newline; `System.out.print()` stays on the same line
- Variables must be declared with a type before use: `int age = 20;`
- The four most common types: `int`, `double`, `char`, `boolean`
- Java uses camelCase for variable names and ALL_CAPS for constants
- Integer division drops the decimal — use `double` when you need precision
- Combined assignment operators (`+=`, `-=`, etc.) are shortcuts for updating variables
- The `final` keyword makes a variable a constant that cannot change
- `String` holds text; the `+` operator concatenates strings
- `Scanner` reads input from the keyboard — remember to import it first
- Comments (`//` and `/* */`) document your code for humans

Vocabulary Review

Term	Definition
Class	A container that holds your program's code; every Java program must have at least one
Method	A named block of code that performs a task; `main` is the entry point

Term	Definition
Statement	A single instruction; ends with a semicolon
Variable	A named storage location in memory
Data Type	Defines what kind of data a variable can hold (`int` , `double` , etc.)
Initialization	Declaring a variable and assigning it a value at the same time
Constant	A variable declared with `final` whose value cannot change
String	A sequence of characters (text), created with double quotes
Concatenation	Joining strings together using the `+` operator
Escape Sequence	A backslash combination that represents a special character (`\n` , `\t` , etc.)
Scanner	A Java class that reads input from the keyboard
camelCase	Naming convention for variables: `firstName` , `totalScore`

Module 02 — Knowledge Check

Answer the following questions. All answers can be found in this document.

1. What is the name of the method where every Java program begins running?

A. begin

B. main

C. start

D. run

2. What does the following code print?

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

A. `Hello` on one line, `World` on the next

B. `HelloWorld` with no space

C. An error

D. `Hello World` on one line

3. Which data type would you use to store the price of an item (like $19.99)?

A. `double`

B. `int`

C. `boolean`

D. `char`

4. Which of these is a valid Java variable name?

A. `2ndPlace`

B. `total score`

C. `totalScore`

D. `class`

5. What is the result of `15 / 4` when both values are `int` ?

A. 3.75

B. 3

C. 4

D. 3.0

6. What keyword makes a variable a constant in Java?

A. `static`

B. `const`

C. `fixed`

D. `final`

7. What does the `+` operator do when used between two `String` values?

A. Joins (concatenates) them into one string

B. Adds their lengths together

C. Compares them alphabetically

D. Causes a compile error

8. Which line is needed at the top of your program to use the `Scanner` class?

A. `include Scanner;`

B. `using Scanner;`

C. `import java.util.Scanner;`

D. `require java.Scanner;`

9. What is the value of `result` after this code runs?

```
int result = 2 + 3 * 4;
```

A. 20

B. 9

C. 14

D. 24

10. What does `\n` represent inside a Java string?

A. A null character

B. A new line

C. The letter n

D. A backslash followed by n

✅ Answer Key

Question	Answer	Explanation
1	**B**	The `main` method is where every Java program begins execution.
2	**D**	`print` keeps the cursor on the same line, then `println` adds the rest and moves to the next line. Output: `Hello World` on one line.
3	**A**	`double` stores decimal numbers like $19.99.
4	**C**	`totalScore` follows camelCase rules. Names cannot start with a digit, contain spaces, or be reserved words.
5	**B**	Integer division drops the decimal: 15 / 4 = 3, not 3.75.
6	**D**	The `final` keyword makes a variable a constant.
7	**A**	The `+` operator concatenates (joins) strings together.
8	**C**	`import java.util.Scanner;` is required to use the Scanner class.
9	**C**	Multiplication before addition: 3 * 4 = 12, then 2 + 12 = 14.
10	**B**	`\n` represents a new line inside a Java string.