

MODULE 1

BASIC ELEMENTS OF JAVA

IN THIS CHAPTER, YOU WILL:

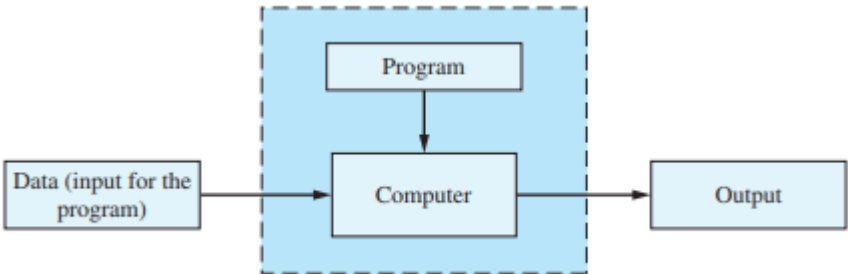
- ✓ Become familiar with the basic components of a Java program, including methods, special symbols, and identifiers
- ✓ Explore primitive data types
- ✓ Discover how to use arithmetic operators
- ✓ Examine how a program evaluates arithmetic expressions
- ✓ Explore how mixed expressions are evaluated
- ✓ Learn about type casting
- ✓ Become familiar with the String type
- ✓ Learn what an assignment statement is and what it does
- ✓ Discover how to input data into memory by using input statements
- ✓ Become familiar with the use of increment and decrement operators
- ✓ Examine ways to output results using output statements
- ✓ Learn how to import packages and why they are necessary
- ✓ Discover how to create a Java application program
- ✓ Learn how to understand and correct syntax errors
- ✓ Explore how to properly structure a program, including using comments to document a program
- ✓ Learn how to avoid bugs using consistent and proper formatting, and code walk-through
- ✓ Learn how to do a code walk-through

A **computer program**, or a program, is a sequence of statements intended to accomplish a task.

Programming is a process of planning and creating a program.

People who write programs—that is, **programmers**—find this second view to be more useful when they design a program.

Figure 1 – Running a Program



PROGRAMMING LANGUAGES, COMPILERS, AND INTERPRETERS

HIGH LEVEL LANGUAGE

Most modern programming languages are designed to be relatively easy for people to understand and use. Such languages are called **high-level languages**. **Java** is a high-level language.

C++, C#, COBOL, Python, and Ruby, are also high-level languages.

Compiler - A program that translates a program written in a high-level language into the equivalent machine language. (In the case of Java, this machine language is the bytecode.)

Interpreters - translate and execute portions of code at a time

LOW LEVEL LANGUAGE

Machine Language -The language that the computer can directly understand.

Assembly language is a symbolic form of machine language that is easier for people to read.

Assembler - A program that translates a program written in assembly language into an equivalent program in machine language such languages are called **low-level languages**.

Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

PROCESSING A JAVA PROGRAM

Java has two types of programs—**applications** and **applets**.

An **application** is just a regular program.

An **applet** sounds as though it would be a little apple, but the name is meant to convey the idea of a little application.

PROGRAMMING WITH THE PROBLEM ANALYSIS–CODING– EXECUTION CYCLE

Algorithm

- A step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.
- An algorithm is a set of directions for solving a problem.

EXAMPLE 1.1

In this example, we design an algorithm to find the perimeter and area of a rectangle. To find the perimeter and area of a rectangle, you need to know the rectangle’s length and width. The perimeter and area of the rectangle are then given by the following formulas:

perimeter = 2 * (length + width)
area = length *width

- The algorithm to find the perimeter and area of the rectangle is:
1. Start
 2. Get the length of the rectangle.

3. Get the width of the rectangle.
4. Find the perimeter using the following equation: $\text{perimeter} = 2 * (\text{length} + \text{width})$
5. Find the area using the following equation: $\text{area} = \text{length} * \text{width}$
6. Stop

Two methods of representing an algorithm.

1. Pseudocode

- ✓ is a mixture of English and Java.
- ✓ is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

Example 1.2

To check positive or negative number

```

BEGIN

READ num

IF (num>0) THEN

DISPLAY num is positive

ELSE





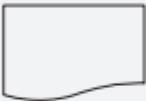


DISPLAY num is negative

END IF

END
  
```

2. Flowchart - is a pictorial representation of an algorithm.

Key Flowchart Modeling Concepts

FLOWCHART SYMBOL	MEANING
	A terminator shows the start and stopping points of the program.
	An arrow shows the direction of the process flow.
	A rectangle represents a process step or activity.
	A diamond indicates a decision point in the process.
	This symbol represents a document or report.
	This rhombus represents data used as inputs/outputs to/from a process.
	This cylinder represents a database.

PROGRAMMING METHODOLOGIES

The approach to analyzing such complex problems, planning for software development and controlling the development process

Types of Programming Methodologies

Procedural Programming

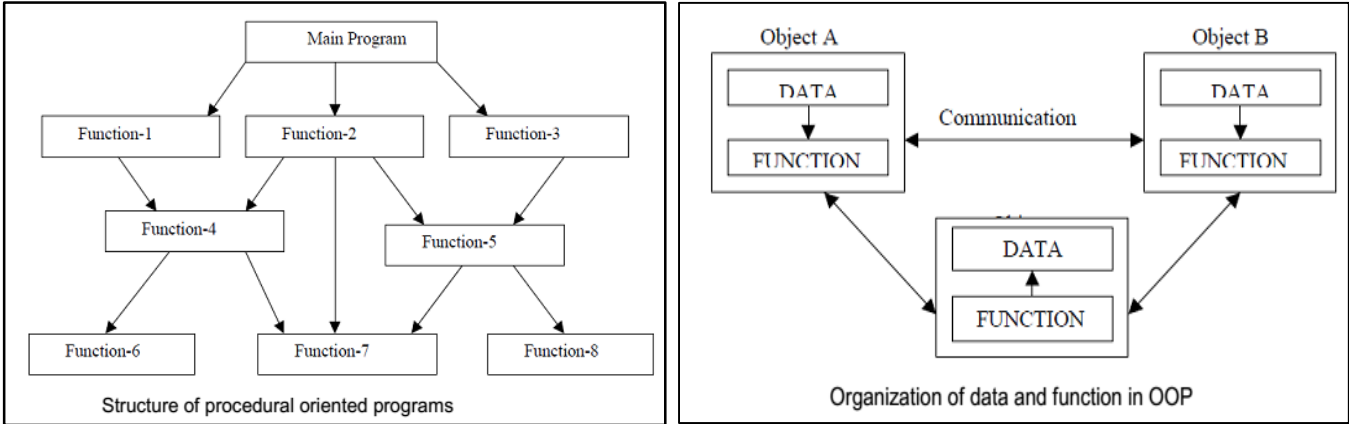
Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity.

Example – For a calculator program that does addition, subtraction, multiplication, division, square root and comparison, each of these operations can be developed as separate procedures. In the main program each procedure would be invoked on the basis of user’s choice.

Object-oriented Programming

Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

Example – If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.



EXERCISE 1.0

In a whole bond paper or MS Word. Answer the following question.

RUBRICS:

CREATIVITY	20
COMPLETENESS	20
ACCURACY	20
CONTENT AND ORIGINALITY	40
TOTAL	100%

1. Create an Algrothim, Pseudocode and Flowchart to check positive or negative number
2. What did your learn in Module 1.

MODULE 2

A First Java Application Program

Consider the following Java (application) program:

```
//*****
// This is a simple Java program. It displays three lines
// of text, including the sum of two numbers.
//*****

public class ASimpleJavaProgram
{
    public static void main(String[] args)
    {
        System.out.println("My first Java program.");
        System.out.println("The sum of 2 and 3 = " + 5);
        System.out.println("7 + 8 = " + (7 + 8));
    }
}
```

The text “First Java Program” is a **literal string** of characters—a series of characters that will appear in output exactly as entered.

BASICS OF JAVA PROGRAM

COMMENT

Comments can be used to identify the authors of the program, give the date when the program is written or modified, give a brief explanation of the program, and explain the meaning of key statements in a program.

ASimpleJavaProgram, given in the previous section, contains the following comments:

```
//*****
// This is a simple Java program. It displays three lines
// of text, including the sum of two numbers.
//*****
```

A Java program has two common types of comments—single-line comments and multiple-line comments. **Single-line comments** begin with `//` and can be placed anywhere in the line. Everything encountered in that line after `//` is ignored by the compiler.

Multiple-line comments are enclosed between `/*` and `*/`. The compiler ignores anything that appears between `/*` and `*/`.

SPECIAL SYMBOL

+	-	*	/
.	;	?	,
<=	!=	=	>=

The **first row** includes mathematical symbols for addition, subtraction, multiplication, and division. The **second row** consists of punctuation marks taken from English grammar. Note that the comma is a special symbol. . The **third row** contains symbols used for comparisons.

IDENTIFIERS

Identifiers are names of things, such as variables, constants, and methods that appear in programs. Some identifiers are predefined; others are defined by the user. All identifiers must obey Java’s rules for identifiers.

Identifier: A Java identifier consists of letters, digits, the underscore character (`_`), and the dollar sign (`$`) and must begin with a letter, underscore, or the dollar sign.

The following are legal identifiers in Java:

```
first
conversion
payRate
counter1
$Amount
```

Examples of Illegal Identifiers

Illegal Identifier	Description
employee Salary	There can be no space between employee and Salary .
Hello!	The exclamation mark cannot be used in an identifier.
one+two	The symbol + cannot be used in an identifier.
2nd	An identifier cannot begin with a digit.

RESERVED WORDS

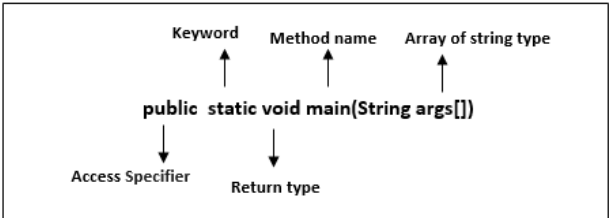
Reserved words are also called keywords. The letters in a reserved word are always lowercase. Like the special symbols, each reserved word is considered a single symbol. Furthermore, reserved words cannot be redefined within any program; that is, they cannot be used for anything other than their intended use.

JAVA KEYWORDS

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

METHOD

The `main()` is the starting point for JVM to start execution of a Java program. Without the `main()` method, JVM will not execute the program. The syntax of the `main()` method is:



public: It is an access specifier. We should use a `public` keyword before the `main()` method so that JVM can identify the execution point of the program. If we use `private`, `protected`, and `default` before the `main()` method, it will not be visible to JVM.

static: You can make a method static by using the keyword `static`. We should call the `main()` method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the `main()` method.

void: In Java, every method has the return type. `Void` keyword acknowledges the compiler that `main()` method does not return any value.

main(): It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the `main()` method.

String args[]: The `main()` method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

DATA TYPES

Data type: A set of values together with a set of operations on those values

PRIMITIVE DATA TYPE

The primitive data types are the fundamental data types in Java.

There are three categories of primitive data types:

- Integral, which is a data type that deals with integers, or numbers without a decimal part (and characters)
- Floating-point, which is a data type that deals with decimal numbers
- Boolean, which is a data type that deals with logical values

Integral data types are further classified into five categories: char, byte, short, int, and long.

TABLE 1.0 JAVA PRIMITIVE DATA TYPE

TYPE	DEFINITION	MINIMUM	MAXIMUM	DEFAULT
byte	8-bit signed integer	-128	127	0
short	18-bit signed integer	32,768	32,767	0
int	32-bit signed integer	-2 ³¹	2 ³¹	0
long	64-bit signed integer	-2 ⁶³	2 ⁶³ -1	0L
float	Single-precision 32-bit IEEE 754 floating point number	1.40239846x10 ⁻⁴⁵	3.40282347x10 ³⁸	0.0f
double	Double-precision 64-bit IEEE 754 floating point number	4.9406564581246544 x10 ⁻³²⁴	1.79769313486231570 x10 ⁻³⁰⁸	0.0d
boolean	One bit of information; flag indicator	false	true	false
char	Single 16-bit Unicode character	'\u0000' (or 0)	'\uffff' (or 65,535)	'\u0000'

DECLARING AND USING CONSTANTS AND VARIABLES

A data item is **constant** when its value cannot be changed while a program is running.

A **variable** is a named memory location that can store a value. A variable can hold only one value at a time, but the value it holds can change.

Declaring Variables

A variable declaration is a statement that reserves a named memory location and includes the following:

- ✓ A data type that identifies the type of data that the variable will store
- ✓ An identifier that is the variable's name
- ✓ An optional assignment operator and assigned value, if you want a variable to contain an initial value
- ✓ An ending semicolon

The basic form of Variable declaration

data_type variable_name = value;

Example:

The data_type is one of Java's data types and variable_name is the name of the variable. To declare more than one (1) variable of the specified type, use a comma-separated list. The following are some valid examples of variable declaration and initialization in Java:

```
int a,b,c;  
int a = 10, b = 4 +5;  
int x = b + 10;
```

```
float f = 2.30f;  
double grade = 1.50;  
char letter = "E"
```

```
byte a;  
int size; size =50;
```

Declaring Named Constants

A variable is a named memory location for which the contents can change. If a named location’s value should not change during the execution of a program, you can create it to be a named constant. A named constant is also known as a symbolic constant. A named constant is similar to a variable in that it has a data type, a name, and a value. A named constant differs from a variable in several ways:

- ✓ In its declaration statement, the data type of a named constant is preceded by the keyword final.
- ✓ A named constant can be assigned a value only once, and then it cannot be changed later in the program. Usually you initialize a named constant when you declare it; if you do not initialize the constant at declaration, it is known as a blank final, and you can assign a value later. Either way, you must assign a value to a constant before it is used.
- ✓ Although it is not a requirement, named constants conventionally are given identifiers using all uppercase letters, using underscores as needed to separate words

The syntax to declared named constant:

```
final data_type variable_name = value;
```

For example, each of the following defines a conventionally named constant:

```
final int NUMBER_OF_DEPTS = 20;
final double PI = 3.14159;
final double TAX_RATE = 0.015;
final string COMPANY = "ABC Manufacturing";
```

EXERCISE 2.0

1. Using you own IDE select an object in your kitchen and create a Java console program that uses println() statements to draw that item.

Screenshot your code and output.

Rubrics:

CREATIVITY	20
COMPLETENESS	20
ACCURACY	20
CONTENT AND ORIGINALITY	40
TOTAL	100%

