Programming with Python and Java
(CS 29008)

School of Electronics Engineering, KIIT DU

Chapter 1

Lab 1: Introduction to Java

1.1 High-level programming

High-level programming languages are third generation programming language and are classified into two categories: (i) *procedure-oriented* programming and (i) *object-oriented* programming.

1.1.1 Procedure-oriented programming

A program in procedure-oriented language is nothing but a set of functions. Each function has a well-defined purpose and well-defined interface to other functions. A function can have its own data known as *local data* and multiple functions can access the same data known as *global data*. Figure 1.1 shows the structure of a procedure-oriented program.

Some of the drawbacks of procedure-oriented languages are:

1. Importance is on functions rather than data.

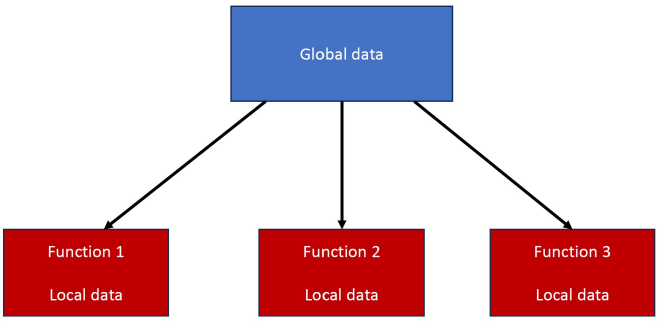

Figure 1.1: Basic structure of a procedure-oriented program

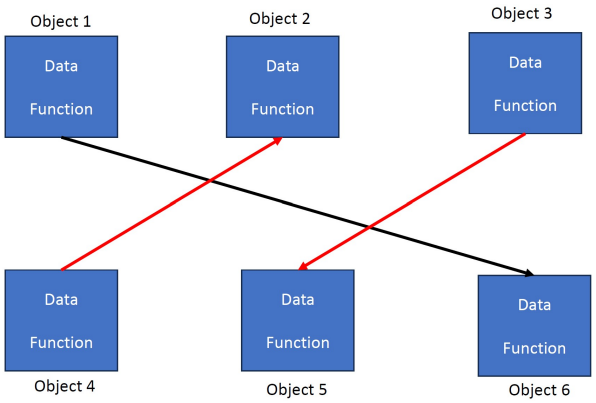

Figure 1.2: Basic structure of an object-oriented program

2. Global data is not secured in a procedure-oriented language.

3. They do not represent real-world problems efficiently.

1.1.2 Object-oriented programming

In object-oriented language, a program is written as a set of different objects. Objects communicate with each other through functions. Figure 1.2 shows the structure of an object-oriented program, where, each object has its unique behaviour (i.e. each object contains localized data and functions and hence data is protected as it is not global).

Object-oriented programming languages have following features:

- Data abstraction
- Encapsulation
- Inheritance
- Polymorphism

1.1.3 Procedure-oriented versus object-oriented

Table 1.1 lists the common differences that exist between procedure-oriented and object-oriented programming languages.

Table 1.1: Comparison between procedure- and object-oriented programs

Parameters	Procedure-oriented	Object-oriented
Program structure	Program is divided into functions.	Program is divided into objects.
Importance	Importance is given to functions.	Importance is given to data.
Approach	Top-down	Bottom-up
Access specifiers	No access specifiers	Has three access specifiers (namely, *public, private, protected*)
Data hiding	No data hiding is possible, hence security is not possible .	Allows data hiding, hence secured programming is possible.
Example	C, FORTRAN	C++, JAVA

1.2 Java programming language

In June 1991, **James Gosling, Mike Sheridan, Patrick Naughton** initiated the Java language project. Java was originally designed for small embedded systems in electronics appliances, for example, set-top boxes, washing machines and so on. Java technology is also incorporated in computer networking. The Java programming language has the following features:

- Simple
- Portable
- Platform independent
- Multi-threaded
- Architecture-neutral
- Object-oriented

1.3 Keywords and Identifiers

The *keywords* are reserved words which have pre-defined meaning assigned by the Java language. The keywords must be used exclusively for their intended purposes and must not be used for any other purposes. Following (Table 1.2 is the list of keywords supported by Java:

Table 1.2: Java keywords

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

Identifiers

- Names given to various program elements (variables, constants, class, methods, etc.)
- May consist of letters, digits, and the underscore (_) character
- Blank space and commas are not allowed.
- First character must be an alphabet or underscore.
- It should not be a keyword.
- An identifier can have any number of characters (however, it is a good practice to keep the name short).
- Java is case-sensitive. For example, area, Area and AREA are all different variable names.

1.4 Your first Java program

Figure 1.3 shows a basic Java program. Here,

- *class Example1* uses the keyword **class** to declare a new class is being defined. **Example1** is an identifier (i.e. the name of the class).
- In the second line, *public* keyword is an access modifier, *static* keyword allows **main()** to be called and the keyword **void** simply tells the compiler that **main()** does not return a value.
- **String args**[] declares a parameter named **args** which is an array of type *String*. In this case, **args** receives any command-line arguments when the program is executed.
- The built-in **println()** method displays the output passed to it. This method is defined within a pre-defined class **System**. **out** is the output stream that is connected to the console.

```
class Example1 {
    public static void main(String args[]) {
        System.out.println("Programming with Python and Java!");
    }
}
```

Figure 1.3: A sample Java program

1.5 Java program editing, compilation and execution

1. Any text editor can be used to write Java programs. For example,
   - In Windows OS, *Notepad, Notepad++*, etc. can be used.
   - In Linux, *vi, emacs, gedit*, etc. can be used.
2. Save the program with a filename, say, **Example1.java** (for the above program).
3. To compile the program,
   - Open a command prompt (in Windows) or terminal (in Linux).
   - Move to the directory where you have saved your Java program.
   - Enter the command to compile the above program:
     javac Example1.java
4. To execute the Java program, type the command in the command prompt

   java Example1

1.6 Java arithmetic operators

The arithmetic operators are used to perform arithmetic operations like addition, subtraction, multiplication, division over numeric values by formulating appropriate arithmetic expressions. Figure 1.5 illustrates the usage of arithmetic operators $[+, -, *, /, \%]$. In this program,

- we accept the inputs from the keyboard. To do this, we use the built-in class **Scanner** which is available in **java.util.** package.
- In order to use the class **Scanner**, we need to import the class with the help of the statement **import java.util.*** mentioned at the first line of the program.
- We should declare a reference to an object of **Scanner** type as **Scanner sc = new Scanner(System.in);** (refer to line 4 in the program).
- Table 1.3 shows how to read a value for the variable.

Table 1.3: Commands for reading a value to different data types

Data type	Command
int a	a = sc.nextInt();
float a	a = sc.nextFloat();
double a	a = sc.nextDouble();
string str	str = sc.next();

```
D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>javac Arithmetic.java

D:\KIIT\Programming_With_Python_Java\My_Codes\Java_Program>java Arithmetic
Enter the value of a
5
Enter the value of b
10

Integer Arithmetic

Sum = 15
Difference = -5
Product = 50
Quotient = 0
Remainder = 5
```

Figure 1.4: Input and output

1.7 Lab 1 Exercises

Objectives:

- To learn writing, executing and debugging programs related to basic I/O functions.

```
import java.util.*;

class Arithmetic {
        public static void main(String args[]) {
                int a, b;
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter the value of a");
                a = sc.nextInt();
                System.out.println("Enter the value of b");
                b = sc.nextInt();
                System.out.println();
                System.out.println("Integer Arithmetic");
                System.out.println();
                System.out.println("Sum = " + (a + b));
                System.out.println("Difference = " + (a - b));
                System.out.println("Product = " + (a * b));
                System.out.println("Quotient = " + (a / b));
                System.out.println("Remainder = " + (a % b));
        }
}
```

Figure 1.5: Usage of arithmetic operators

- To learn writing, executing and debugging programs related to arithmetic operators.

Outcomes:

- After completing this, the students would be able to develop basic Java programs.

Lab Assignments

1. Write a Java program to print the following on the console: "Name"
   "Roll number"
   "Branch"
   "Department"
   "University"

2. Write a Java program to find the area and circumference of a circle, given its radius, r.

3. Write a Java program to accept the length and breadth of a rectangle and display its area and perimeter.

4. Write a Java program to accept the number of seconds and display its equivalent number of hours, number of minutes and number of seconds (Hint: Use / and % operators).