CHAPTER 2

INTRODUCTION TO JAVA

Contents

Key Objectives

In this chapter, we will briefly present some basic syntactic concepts used for the practical work activities illustrated by examples. In particular, data types, arrays, strings, conditional and repetitive structures, functions, etc. The chapter concludes with a series of practical exercises designed to reinforce student understanding and allow him to apply what he has learned.

2.1 What is Java ?

Java is a programming language which helps us to write instructions, called code, to make the computer do a particular task. Java is one of several programming languages we use to instruct the computer. Among these languages, we can cite C, C++, Python, etc. Thus, Java is a mediator between human and computer. Java can be used everywhere and the applications where it can be used are endless. It can be used to instruct robots to carry out a particular task, to create Android apps (almost every anroid phone application uses java), to create desktop applications, and many more.

2.2 How does Java work ?

A java program follows the following steps:

① First we write a list of java instructions (using any Java editors and Integrated Development Environments (IDEs), such as IntelliJ, NteBeans or Eclipse)

② Then, the program is converted into intermediate version called *byte-code* which is not readable by humans.

③ The *byte-code* is then read by a special software called a Java interpreter, which translates it to the *machine language*.

④ The computer reads the translated code and executes the tasks requested.

2.3 Building your first Java program

Let's start with the first program, called Hello World. It asks the computer to display (print) "Hello World" on screen. The command used to display some text on the screen is : `System.out.print()`; and the text which being printed must be inside the curved brackets and enclosed in double quotes. So, the hello world program will look like the following code:

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

Now, you can hit the Run button and check if you get the desired output.

- Java is a case sensitive language, the uppercase letters and lowercase letters are different.
- The text is written in quotes to tell Java that it has to be cinsidered as a string.

2.4 Input/Output & Control statements

2.4.1 Primitive (Basic) Data Types

Recapitulation 2.1

Primitive (basic) types of Java correspond to those of the C language, plus the byte and boolean types. Strings are managed (differently from C) as classes. Table 2.1 lists the basic Java types.

Type	Description	Valeurs
byte	un entier signé sur 8 bits	de –128 à +127
short	un entier signé sur 16 bits	de –32 768 à +32 767
int	un entier signé sur 32 bits	de –2147483648 à 2147483647
long	un entier signé sur 64 bits	de l'ordre de (±) 9 milliards de milliards
float	un réel sur 32 bits	de l'ordre de 1.4E-45 à 3.4E38
double	un réel sur 64 bits	de l'ordre de 4.9E-324 à 1.79E308
boolean	vrai ou faux	true ou false
char	un caractère Unicode entier positif sur 16 bits	entre 0 et 65535

Table 2.1: Primitive data types of Java

2.4.2 Implicit and explicit casting

Recapitulation 2.2

Arithmetic operators are only defined when both of their operands are of the same type. But we can write mixed expressions in which operands of different types are involved. Implicit (automatic) type conversions allowed in Java with examples are illustrated by Listing 2.1. However, the reverse is not allowed and we must explicitly force the type conversion. To force the conversion of any expression into a type of choice, we use a special operator called cast.

Its denotation consists of placing in parentheses, before the value to be converted, the type into which it must be converted. Explicit (manual) type conversions in Java with examples are illustrated by Listing 2.2.

```
//converting a smaller type to a larger type size
byte -> short -> char -> int -> long -> float -> double
//Example:
int n = 1, p = 5;
float x = 1.5f;
float b= n * x + p; // Automatic casting: float to double
long a=n;           // Automatic casting: int to long
double d =n+x;      // Automatic casting: int to double
char c='a';
int e=c;            // Automatic casting: char to int
```

Listing 2.1: Automatic casting in Java

```
// converting a larger size type to a smaller size type
double -> float -> long -> int -> char -> short -> byte
//Example:
double d = 9.78d;
int myInt = (int) d; // Manual casting: double to int
float f=3;
f=(float)(d+3);      // Manual casting: double to float
n= (int) f ;         // Manual casting: float to int
char c='a'; c=(char) e;   // Manual casting: int to char
```

Listing 2.2: Manual casting in Java

2.4.3 Displaying data

Recapitulation 2.3

Displaying data on screen in Java is done using the `System.out.print` or `System.out.println` functions. This function allows to display literal strings in quotes (e.g., `System.out.println ("Hello world!");`) or simple or complex Java expressions (e.g., `System.out.println(a+b);`).

2.4.4 Reading data from keyboard

Recapitulation 2.4

Regarding reading data from the keyboard, we need first to import the Scanner class, for this we write at the beginning of the program: `import java.util.Scanner;`, *then we can create a variable of type Scanner as follows:* `Scanner read = new Scanner (System.in);`, *then, the read variable can be used as many times as necessary to request values from the keyboard, and depending on the desired data type the appropriate method is used, as illustrated by Listing 2.3:*

```
1 //import java.util.Scanner; in the preambule of the program
2  Scanner read = new Scanner(System.in);
3  int n=read.nextInt();
4  double d=read.nextDouble();
5  float f=read.nextFloat();
6  String s=read.nextLine();
7  char c=read.next().charAt(0);
8  //and so on ...
```

Listing 2.3: Reading data from keyboard

2.4.5 Control statements

Recapitulation 2.5

Java provides several control statements to manage program flow, including:

- *Conditional Statements: if, if-else, nested-if, if-else-if*
- *Switch-Case: For multiple fixed-value checks*
- *Jump Statements: break, continue, return*

Java's control statements are largely analogous to those found in C. The Listing 2.4 illustrates the syntax of the Java's control statements

```
1  //if without else
2  if(test) //test is a logic expression
3  {  block_of_instructions }
4
5  //if with one instruction
6  if(test) one_intruction;
```

```java
//if with else
if(test) {  block_of_instructions }
else
{  block_of_instructions }

//if with else if
if(test)
{  block_of_instructions  }
else if {  block_of_instructions }

// switch statement
switch (expression) {
  case value1:// Statements
  break; // Exits the switch block
  case value2:// Statements
  break;

  // ... any number of cases

  default: // Statements executed if no case matches (optional)
}
```

Listing 2.4: Control statements in Java

2.5 Array & ArrayList

2.5.1 Array

Recapitulation 2.6

- *In Java, an array is a data structure designed to store a fixed-size sequence of elements of the same data type, that are accessed using their index, which starts from 0.*
- *In Java, an array can be declared in several ways (see Listing 2.5 for single-dimensional arrays and Listing 2.6 for multi-dimensional arrays).*
- *Using a traditional for loop with an index for iterating through Arrays, as follows:*

```java
    for (int i = 0; i < t.length; i++) {
```

```
        System.out.println(t[i]);
    }
```

- *The length property provides the number of elements in the array (`int s=t.length`).*
- *Attempting to access an index outside the valid range (0 to length - 1) will result in an ArrayIndexOutOfBoundsException.*
- *The elements of an array can be of any type, not just a primitive type.*

```
int tab1[] = {1, 2, 3};// the same as language C
int[] tab2 = {1, 2, 3};
int[] tab3 ;
tab3=new int[6];/tab3 refers to an array of 6 integers
int []t1,t2;//t1 et t2 are references to arrays of integers
int t[10] ; // error: we cannot indicate the size like this
```

Listing 2.5: Declaration of simple arrays in Java

```
// these three statements are equivalent
int t [] [] ;
int [] t [] ;
int [] [] t ;

//an array of two rows and a varying number of columns
int t [] [] = {new int [3], new int [2]};
```

Listing 2.6: Declaration of multi-dimensional arrays in Java

2.5.2 Dynamic array (List)

Recapitulation 2.7

Dynamic arrays (lists) have the particularity of being able to change size during program execution. In Java, they are defined using the ArrayList class from the `java.util.ArrayList` library, as follows:

```
    ArrayList <Type> myList;
    myList = new <Type> ArrayList();
```

The type of the elements of an arraylist cannot be a simple type (int or double for example). It

must be one of the advanced types corresponding to simple data types. For example: Integer for int, Float for float, Double for double, etc. As it can be a specific class (Person for example). The following example defines an arraylist of integer:

```
import java.util.ArrayList; // Import the ArrayList class
ArrayList<Integer> numbers = new ArrayList<Integer>();
```

The ArrayList class offers various methods for manipulating an arraylist elements. Some of the most commonly used ArrayList methods are summarized in Table 2.2. An example of adding and getting elements of an ArrayList of String is given bellow:

```
ArrayList<String> cars = new ArrayList<String>();
cars.add("Mazda");
cars.add("Ford");
cars.add("BMW");
cars.add("Volvo");
for (int i = 0; i < cars.size(); i++) {
System.out.println(cars.get(i));
}
```

R To loop through the elements of an ArrayList we use a for loop with the the size() method to specify how many times the loop should run. If an attempt to access an element at an index that is outside the valid range when looping through an ArrayList in Java, an IndexOutOfBoundsException occurs. The valid indices for an ArrayList are 0 to size() - 1.

Method	Returned result
$size()$	Returns the number of elements in this list.
$add(element)$	Appends the specified element to the end of this list.
$add(int\ index, element)$	Inserts the specified element at the specified position in this list.
$get(int\ index)$	Returns the element at the specified position in this list
$remove(int\ index)$	Removes the element at the specified position in this list
$set(int\ index, element)$	Replaces the element at the specified position in this list with the specified element
$isEmpty()$	Returns true if this list contains no elements
$clear()$	Removes all of the elements from this list
$indexOf(element)$	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element
$contains(element)$	Returns true if this list contains the specified element

Table 2.2: Some of the most commonly used methods of the class ArrayList

2.6 Characters & String

Recapitulation 2.8

- *char is a primitive data type in Java, which directly stores the character value in memory.*
- *A char variable can hold only a single character, enclosed in single quotes (* `char c='B';`*).*
- *The Character class provides utility methods for character manipulation, for example:* `Character.isDigit('5'); Character.toUpperCase('r'); Character.isLowercase ('Z'); Character.isLetter('a').`
- *In Java, converting a char to its numerical representation can be achieved in several ways, depending on whether you want the character's Unicode/ASCII value or its numeric digit value if the character represents a digit.*
- *We can use the Unicode/ASCII Value (Implicit Type Casting):*

```
char c = 'A';
int Value = c; // Value will be 65 (ASCII/Unicode value of 'A')
char d = '5';
```

```
int digiteValue = d; // digiteValue will be 53 (ASCII/Unicode value of '5'
```

- *If the char represents a digit ('0'-'9') and we want its actual numeric value, we can use* `Character.getNumericValue()` *method: this method returns the integer value represented by the character:*

```
char c = '2';
int n = Character.getNumericValue(c); // n will be 2
System.out.println("Numeric value of '2': " + n);

```

Recapitulation 2.9

- *String is a class in Java, and String variables are objects that stores sequences of characters, that can hold zero or more characters, enclosed in double quotes ("", "Java", "Java Programming", ...).*
- *String in Java are immutable; their content cannot be changed after creation. Any operation that attempts to modify a String actually creates a new String.*
- *The String class provides a variety of methods for manipulating and working with strings, such as concatenation, comparison, searching, and substring extraction. Table 2.3 summarizes some of the most commonly used functions of the String class.*

Method	Returned result
$length()$	Returns the number of characters in the string
$charAt$	Returns the character at the specified index
$compareTo$	Compares two strings lexicographically
$startsWith$	Checks if the string begins with the specified prefix. Returns $true$ ou $false$
$endsWith$	Checks if the string ends with the specified suffix. Returns $true$ ou $false$
$equals$	Compares two strings for equality (case-sensitive)
equalsIgnoreCase	Compares two strings for equality, ignoring case differences. Returns $true$ or $false$
$contains$	Checks if the string contains the specified sequence of characters, returns $true$ or $false$
$indexOf$	Returns the index of the first occurrence of the specified substring
$isEmpty$	Checks if the string is empty (has a length of 0)
$toLowerCase$	Converts all characters in the string to lowercase
$toUpperCase$	Converts all characters in the string to uppercase

Table 2.3: Some of the most commonly used methods of the class String

2.7 Static functions

Recapitulation 2.10

A static method in Java is a part of the class definition. We can define a static method by adding the `static` *keyword to a method. Static methods can be called directly using the class name, without the need to create an object of that class.*

```
public class MathOperations {
      // Static method to add two numbers
  public static int add(int a, int b) {return a + b;}
      // Static method to multiply two numbers
  public static int multiply(int a, int b) {  return a * b;}
  public static void main(String[] args) {
        // Calling static methods:
int sum = add(4, 2);
System.out.println("Sum: " + sum); // Output: Sum: 6
```

```
int product = multiply(5, 3);
System.out.println("Product: " + product); // Output: Product: 15
    }}
```

R We will talk in detail about methods in the next chapter.

2.8 Quick Check

Quick Check 2.1 *Which one of these instructions is correct for printing text on screen ?*

① `System.Out.print`

② `System.out.Print`

③ `System.out.print`

④ `system.Out.print`

Quick Check 2.2 *Which of these instructions contain errors?*

① `System.out.print(Hello World);`

② `System.out.print("Hello World")`

③ `System.out.Print("Hello World");`

④ `System.out.print("Hello World");`

Quick Check 2.3 *What will be the value of* sum *after the following nested for loops are executed?*

①
```
    int sum = 0;
    for (int i = 0; i < 5; i++) {
      sum = sum + i;
      for (int j = 0; j < 5; j++) {
        sum = sum + j;
      }
  }
```

②
```
    int sum = 0;
    for (int i = 0; i < 5; i++) {
```

```
        sum = sum + i;
        for (int j = i; j < 5; j++) {
          sum = sum + j;
        }
    }
```

Quick Check 2.4 *Which of these statements are invalid?*

① `Person[25] person;`

② `Person[ ] person;`

③ `Person person[] = new Person[25];`

④ `Person person[25] = new Person[25];`

Quick Check 2.5 *What is the output of this code?*

```
int[][] table = new int[10][5];
System.out.println(table.length);
System.out.println(table[4].length);
```

Quick Check 2.6 *What is the output from the following code?*

```
List<String> list = new ArrayList<String>();
for(int i = 0; i < 6; i++) {
  list.add("element " + i);
}
list.remove(1);
list.remove(3);
System.out.println(list.get(2));
```

Quick Check 2.7 *What will be the output of the following codes:*

①

```
    char []C=new char[10]
```

```
    for(int i=0;i<10;i++)
    {
      C[i]='i';  System.out.print(C[i]);
    }
```

②

```
    int T[]={0,1,2,3,4,5,6,7,8,9};
    int n=6;
    n=T[T[n]/2];
    System.out.println(T[n]/2);
```

Quick Check 2.8 *Which data type is used to create a variable that should store text?*

① *text*

② *String*

③ *string*

④ *myString*

Quick Check 2.9 *Which method can be used to find the length of a string?*

① *size()*

② *len()*

③ *getLength*

④ *length()*

Quick Check 2.10 *Which method can be used to find the length of an ArrayList?*

① *size()*

② *len()*

③ *getLength*

④ *length()*

2.9 Activities

Exercise 2.1 *Rewrite the following program in Java:*

```c
#include <stdio.h>
int main() {
  int a, b, sum;
  printf("Enter two numbers: ");
  scanf("%d %d", &a, &b);

  sum = a + b;

  printf("Sum = %d\n", sum);
  return 0;
}
```

Exercise 2.2 *Write a Java program that asks the user to enter his name, family name, and age, and then, these information will be displayed by the computer.*

Test Data:

- *Name : Moustapha*
- *Family name : Mohammed*
- *Age : 21*

Expected Output: ***You are Moustapha Mohammed and you are 21 years old.***

Exercise 2.3 *Write a Java program to print the sum, multiplication, division, and the remainder of the division of two numbers*

Exercise 2.4 *Write a Java program to compute the area and perimeter of a circle.*

Test Data: Radius = 5

Expected Output: Area is 78,539816

Perimeter is 31,415926

R *To get π you can use the predefined function $Math.PI$*

Exercise 2.5 *Write a Java program to compute the distance between two points P, W. The two points are defined, in a two-dimensional space, by their coordinates as follow:*

- *P : (x_1, y_1) and,*
- *W : (x_2, y_2).*

R

- *All data are real numbers.*
- *Distance(P, W)=$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$*
- *You can use predefined functions: $Math.sqrt()$ et $Math.pow()$*

Exercise 2.6 *Write a program that inputs temperature in degrees Celsius and prints out the temperature in degrees Fahrenheit. The formula to convert degrees Celsius to equivalent degrees Fahrenheit is:*

*Fahrenheit = 1.8 * Celsius + 32*

Exercise 2.7 *Write a program that does the reverse of Exercise 2.6, that is, input degrees Fahrenheit and prints out the temperature in degrees Celsius. The formula to convert degrees Fahrenheit to equivalent degrees Celsius is Celsius =9/5 * (Fahrenheit + 32)*

Exercise 2.8 *Write a program that inputs the year a person is born and outputs the age of the person in the following format:*

You were born in 1990 and will be (are) 18 this year.

Exercise 2.9 *If you invest P dollars at R percent interest rate compounded annually, in N years, your investment will grow to $P(1 + \frac{R}{100})^N$ dollars. Write an application that accepts P, R, and N and computes the amount of money earned after N years.*

Exercise 2.10 *Your weight is actually the amount of gravitational attraction exerted on you by the Earth. Since the Moon's gravity is only one-sixth of the Earth's gravity, on the Moon you would weigh only one-sixth of what you weigh on Earth.Write a program that inputs the user's Earth weight and outputs her or his weight on Mercury, Venus, Jupiter, and Saturn. Use the values in the following table:*

Planet	*Multiply the Earth Weight by*
Mercury	*0.4*
Venus	*0.9*
Jupiter	*2.5*
Saturn	*1.1*

Exercise 2.11 *When you say you are 18 years old, you are really saying that the Earth has circled the Sun 18 times. Since other planets take fewer or more days than Earth to travel around the Sun, your age would be different on other planets. You can compute how old you are on other planets by the formula:*

$$y = \frac{x \times 365}{d}$$

where x is the age on Earth, y is the age on planet Y, and d is the number of Earth days the planet Y takes to travel around the Sun. Write an application that inputs the user's Earth age and print outs his or her age on Mercury, Venus, Jupiter, and Saturn. The values for d are listed in the following table:

Planet	*d: Number of Earth Days for This Planet to Travel around the Sun*
Mercury	*88*
Venus	*225*
Jupiter	*4.380*
Saturn	*10.767*

Exercise 2.12 *Write a program that accepts a person's weight and displays the number of calories the person needs in one day. A person needs 19 calories per pound of body weight, so the formula expressed in Java is calories = bodyWeight * 19;*

Exercise 2.13 *Write a program that accepts the unit weight of a bag of coffee in pounds and the*

number of bags sold and displays the total price of the sale, computed as:

$$totalPrice = unitWeight * numberOfUnits * 5.99;$$

$$totalPriceWithTax = totalPrice + totalPrice * 0.0725;$$

where 5.99 is the cost per pound and 0.0725 is the sales tax. Display the result in the following manner:

```
Number of bags sold:  32
Weight per bag:  5 lb
Price per pound:  $ 5.99
Sales tax:  7.25Total price:  $ 1027.884
```

Exercise 2.14 *A perfect number is a positive integer that is equal to the sum of its proper divisors. A proper divisor is a positive integer other than the number itself that divides the number evenly. For example, 6 is a perfect number because the sum of its proper divisors 1, 2, and 3 is equal to 6. Eight is not a perfect number because* $1+2+4 \neq 8$. *Write a program that accepts a positive integer and determines whether the number is perfect. Also, display all proper divisors of the number. Try a number between 20 and 30 and another number between 490 and 500.*

Exercise 2.15 ***Guess the number***

"Guess the number" is a simple game where the user tries to guess a number randomly generated by the computer within a specified range. The game rules and steps are as follow:

- *The computer generates a random number* n *from a given range, let's* $n \in [1,10]$*;*
- *The user tries to guess the generated number and inter his given number using keyboard;*
- *The user has a limit number of attempts,* k*, and let's* $k = 5$*;*
- *The computer replays if the generated number* n *matches the guesses number or it is lower/higher than it.*
- *If the guessed number is equal to* n *or if the* K *attempts are reached, the program ends with an appropriate message;*
- *For each attempt, the program also displays the number of attempts remaining;*
- *You can also incorporate further details as displaying score and giving points based on the number of attempts.*

Use the following lines in order to generate a random integer:

```
1  import java.util.Random;
2  //Par exemple, générer un nombre dans l'intervalle [a,b]:
3  Random random = new Random();
4  int nb=a+random.nextInt(b-a);
5
```