OBJECT-ORIENTED PROGRAMMING
LAB 1: JAVA, HOW TO PROGRAM

I. Objective

After completing this first lab tutorial, you can:

- Set up the Java Environment, compile, and run the Java program.
- Know the primitive types and basic operators in Java.
- Use the basic statements: conditional statements, loop statements, and declare the method in Java.
- Use input and output syntax in Java.

II. Overview of Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of computer architecture.

As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

III. Environment Setup

This section will help you to set up the environment for the Java programming language. Follow the steps below:

- Download and install the Java SE Development Kit. You are recommended to use Java version 11. If you use another version, please be careful with the syntax. But all of your work when you are doing the exams or the exercises will be assigned with Java 11.
- Setting up the Path for Windows:
    - Right-click on This PC and select **Properties**.
    - Click **Advanced system settings**.
    - Select tab **Advanced**, click **Environment Variables** button.

- Click the **Path** row in **System Variables** area to include the directory path of the installed JDK (as in FIG). Assuming you have installed it in *C:\Program Files\Java\jdk-11.0.13\bin*, then add it to the Path.
- Then, open the command prompt (CMD) window and type `java -version` and `javac -version`. If you get the message that provides the information about the installed Java version, that means you have successfully set up the JDK and Java Path.

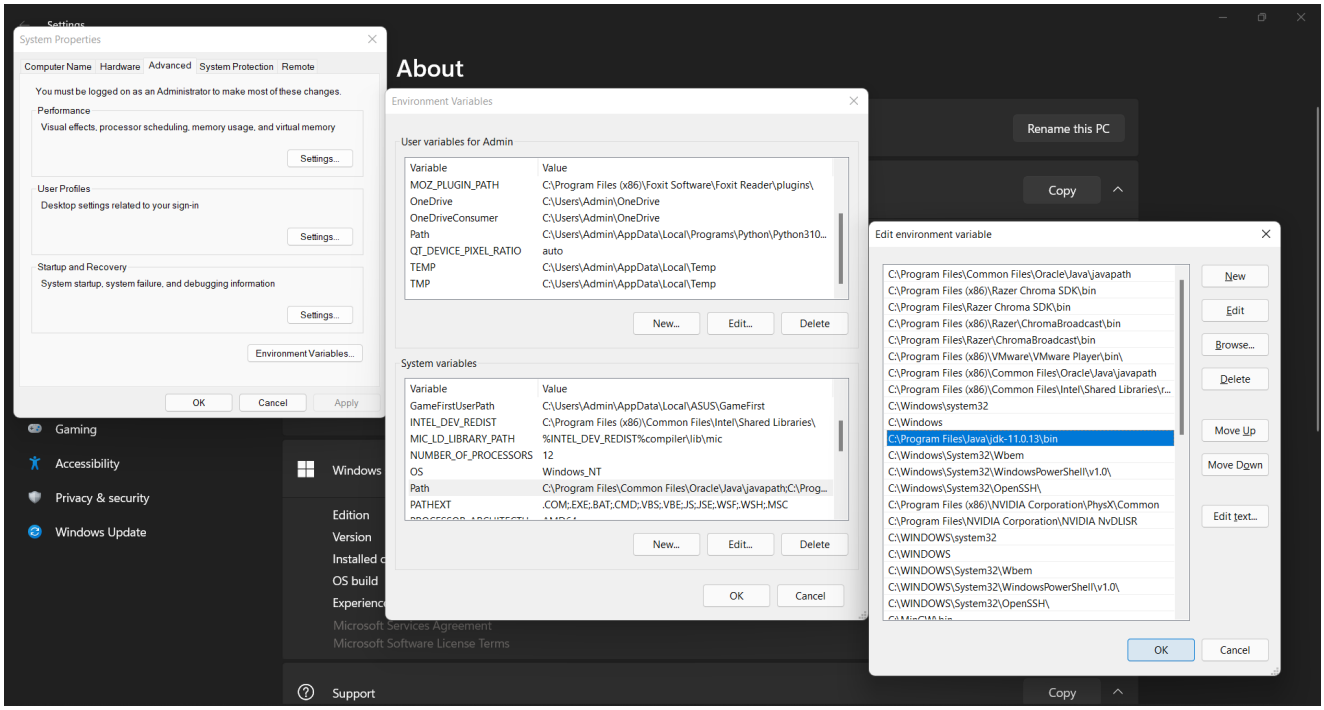


Figure 1. Environment Variables window

- For Java code editor, we strongly recommend Visual Studio Code for your very beginning course in Java. Besides that, you can use Notepad++ or Sublime Text. After completing this course, you can use other Java IDE, e.g., NetBeans, Eclipse, and IntelliJ IDEA.

IV. First Java Program

Let's look at and run your first Java program, which will print the "Hello World" string on the screen.

```
public class MyFirstProgram {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Do the following steps to save the file, compile and run the program (as in Figure 2):

- Open the text editor program (Visual Studio Code, Notepad++, …) and type the above code.

- Save as *MyFirstProgram.java*.
- Open the command prompt window (CMD) and navigate to the directory where you save the file.
- Type `javac MyFirstProgram.java` and press Enter to compile the source code.
- Type `java MyFirstProgram` to run the program.
- Now, you will see the *"Hello World"* string on the screen.

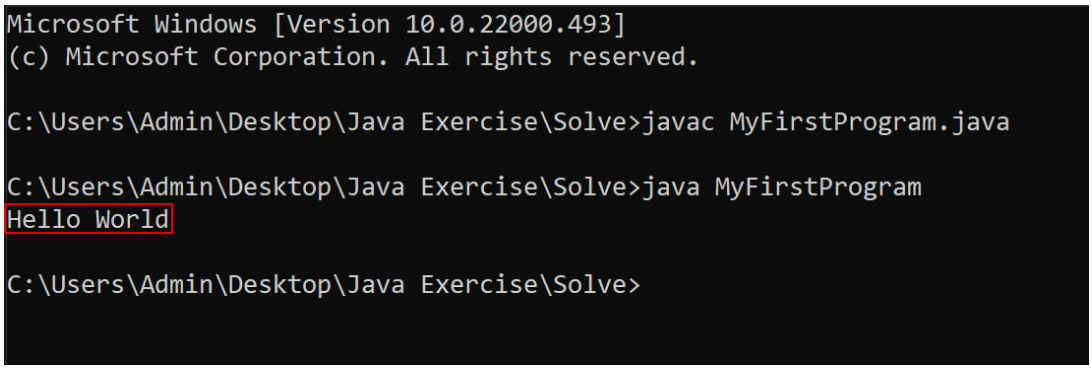


Figure 2. Compile and run Java program

1. Basic syntax

In java, you should keep in mind the following points:

- **Case Sensitivity**: Java is case sensitive, which means the identifier "Hello" and "hello" would have different meanings in Java.
- **Class Names**: For all class names, the first letter should be in Upper-Case. If several words are used to form the name of the class, each inner word's first letters should be in Upper-Case. Example: class `MyFirstProgram`
- **Method Names**: All method names should start with a Lower-Case letter. If several words are used to form the names of the method, then each inner word's first letter should be in Upper-Case.
  Example: `public void methodName()`
- **Program File Name**: The name of the program file has to exactly match the public class name. Example: *public class MyFirstProgram* → *MyFirstProgram.java*
- `public static void main(String args[])`: Java program processing starts from the **main**() method which is a mandatory part of every Java program.

2. Java Identifiers

The identifier is the name used for classes, variables, and methods. To name an identifier in Java, you should remember the following points:

- All identifiers should begin with a letter (A to Z or a to z), a currency character ($), or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A keyword cannot be used as an identifier.
- Most importantly, identifiers are "case sensitive".
- Examples of legal identifiers: `age`, `$salary`, `_value`, `__1_values`.
- Examples of illegal identifiers: `123abc`, `-salary`.

3. Java Modifiers

There are two categories of modifiers in Java:

- Access Modifier: `default`, `public`, `protected`, `private`.
- Non-access Modifiers: `final`, `abstract`, `static`…

4. Comments in Java

Java supports both single-line and multiple-line comments, and they are similar to C and C++. All characters available inside any comments are ignored by the Java compiler.

```
// This is a single-line comment
public class MyFirstProgram {
  public static void main(String[] args) {
    System.out.println("Hello World!");
    /*
    This is the multiple-line comments
    */
  }
}
```

V. Data Types

In this first lab tutorial, we only focus on **primitive data types**. For the user-defined data types, we will discuss later in this course.

Type	Default	Size
boolean	false	1 bit
byte	0	8 bits
char	\u0000	16 bits

short	0	16 bits
int	0	32 bits
long	0	64 bits
float	0.0f	32 bits
double	0.0	64 bits

Let's look at the example below:

```
// MyFirstProgram.java
public class MyFirstProgram {
  public static void main(String[] args) {
    int a = 5, b = 10;
    int sum = a + b;
    System.out.println("a + b = " + sum);
  }
}
```

VI. Basic Operators

Java provides a rich set of operators to manipulate variables, including:

- Arithmetic operators
- Relational operators
- Bitwise operators
- Logical operators
- Assignments operators
- Misc. operators

In this lab tutorial, we discuss arithmetic operators, relational operators, logical operators, and assignment operators. For the others, you can read in the textbook.

1. Arithmetic operators

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra.

Initial two variables a = 20 and b = 10.

Operators	Example
+ (addition)	a + b will give 30
- (subtraction)	a – b will give 10
* (multiplication)	a * b will give 200
/ (division)	a / b will give 2
% (modulus)	a % b will give 0
++ (increment)	b++ will give 21
-- (decrement)	b-- will give 19

2. Relational operators

The followings are the relational operations supported by Java language.

Initial two variables a = 10 and b = 20.

Operators	Example
== (equal to)	(a == b) is false
!= (not equal to)	(a != b) is true
> (greater than)	(a > b) is false
< (less than)	(a < b) is true
>= (greater than or equal to)	(a >= b) is false
<= (less than or equal to)	(a <= b) is true

3. Logical operators

The followings are the logical operators supported by Java language.

Initial two variables A = true and B = false.

Operators	Example
&& (logical "and")	(A && B) is false
|| (logical "or")	(A || B) is true
! (logical "not")	!(A && B) is true

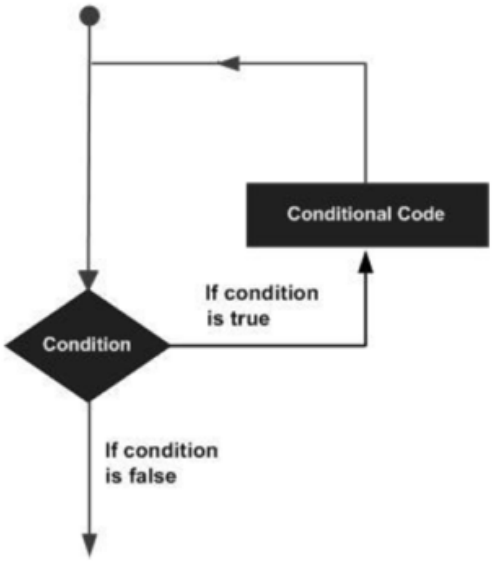
4. Assignment operators

Operators	Example
=	C = A + B will assign the value of A + B to C
+=	C += A is equivalent to C = C + A
-=	C -= A is equivalent to C = C - A
*=	C *= A is equivalent to C = C * A
/=	C /= A is equivalent to C = C / A
%=	C %= A is equivalent to C = C % A

VII. Loop Control

A loop statement allows us to execute a statement or group of statements multiple times. Java programming language provides the following types of iterating to handle looping requirements:

- `while` loop
- `for` loop, and enhanced for loop
- `do ... while` loop



1. *while* loop

A *while* loop statement in Java programming language repeatedly executes the target statement as long as a given condition is true.

Example:

```
public class MyFirstProgram {
  public static void main(String[] args) {
    int i = 1, sum = 0;

    while (i <= 10) {
      sum = sum + i;
      i++;
    }

    System.out.println("sum = " + sum);
  }
}
```

2. *for* loop

A *for* loop is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times. A *for* loop is useful when you know how many times a task will be repeated.

Example:

```
public class MyFirstProgram {
  public static void main(String[] args) {
    int sum = 0;

    for (int i = 1; i <= 10; i++) {
      sum += i; // Same sum = sum + i
    }

    System.out.println("sum = " + sum);
  }
}
```

3. *do ... while* loop

A *do ... while* loop is similar to a while loop, except that, the *do ... while* loop is guaranteed to execute at least one time.

Example:

```
public class MyFirstProgram {
  public static void main(String[] args) {
    int i = 0, sum = 0;
    do {
      sum = sum + i;
      i++;
    } while (i <= 10);
    System.out.println("sum = " + sum);
  }
}
```

VIII. Conditional Statements

Decision-making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be *true*, and optionally, other statements to be executed if the condition is determined to be *false*.

Java provides two main types of conditional statements and one operator:

- `if ... else` statement
- `switch ... case` statement
- `? ... : ...` operator

Example 1:

```
public class MyFirstProgram {
  public static void main(String[] args) {
    int x = 11;
    if (x % 2 == 0) {
      System.out.println("x is even");
    } else {
      System.out.println("x is odd");
    }
  }
}
```

Example 2:

```
public class MyFirstProgram {
  public static void main(String[] args) {
    int x = 11, y = 12;
    if (x < y && x + y >= 10) {
      System.out.println("True");
    } else {
      System.out.println("False");
    }
  }
}
```

IX. Methods

A Java method is a collection of statements that are grouped to operate. The syntax is as follows:

```
modifier returnType nameOfMethod (Parameter List) {
  // statements
}
```

Example:

```
public class MyFirstProgram {
  public static int findMax(int a, int b) {
    return (a > b) ? a : b;
  }

  public static void main(String[] args) {
    int x = 5, y = 6;
    System.out.println("Max is " + findMax(x, y));
  }
}
```

According to the above example, *findMax* is the **static** method. So in the *main* function, you can call directly without creating a new object. You will learn more about the **method** in Lab 4.

X. Text Output

Here is a list of the various print functions that we use to output statements:

- `print()`: This method in Java is used to display text on the console. This text is passed as the parameter to this method in the form of a String. This method prints the text on the console and the cursor remains at the end of the text at the console. The next printing takes place from just here.

Example:

```
class Demo {
  public static void main(String[] args) {
    System.out.print("Hello! ");
    System.out.print("Hello! ");
    System.out.print("Hello! ");
  }
}
```

Output:

```
Hello! Hello! Hello!
```

- `println()`: This method in Java is also used to display text on the console. It prints the text on the console and the cursor moves to the start of the next line at the console. The next printing takes place from the next line.

Example:

```
class Demo {
  public static void main(String[] args) {
    System.out.println("Hello! ");
    System.out.println("Hello! ");
    System.out.println("Hello! ");
  }
}
```

Output:

```
Hello!
Hello!
Hello!
```

- `printf()`: This is the easiest of all methods as this is similar to *printf* in C. Note that *System.out.print*() and *System.out.println*() take a single argument, but *printf*() may take multiple arguments. This is used to format the output in Java.

Example:

```
class Demo {
  public static void main(String args[]) {
    int x = 100;
    System.out.printf("Printing simple" + " integer: x = %d\n",x);
    System.out.printf("Formatted with" + " precision: PI = %.2f\n", Math.PI);

    float n = 5.2f;
    System.out.printf("Formatted to " + "specific width: n = %.4f\n", n);

    n = 2324435.3f;
    System.out.printf("Formatted to " + "right margin: n = %20.4f\n", n);
  }
}
```

Output:

```
Printing simple integer: x = 100
Formatted with precision: PI = 3.14
Formatted to specific width: n = 5.2000
Formatted to right margin: n =         2324435.2500
```

XI. Text Input from Keyboard

Java provides a `Scanner` class that is used to get user input and it is found in `java.util` package.

To use the Scanner class, create an object of the class and use any of the available methods found in the documentation.

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads an integer value from the user
nextLine()	Read a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

Example:

```
import java.util.Scanner;

class MyFirstProgram {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in); // Create a Scanner object

    String name = sc.nextLine(); // Read user input
    int age = sc.nextInt();

    System.out.println("Name: " + name); // Output user input
    System.out.println("Age: " + age);
  }
}
```

XII. Exercises

1. Write a program to print your name, date of birth, and student ID from the keyboard.
2. Write a program to compute the area of a triangle with a height and base provided by the user. ($area = \frac{1}{2} \times base \times height$)

3. Write a function to return the remainder of a division.
4. Write a function to convert the temperature from Fahrenheit to Celsius and a function to convert the temperature from Celsius to Fahrenheit.
5. Write a function to check whether a year is a leap year or not.
6. Write a function to find the minimum between three numbers.
7. Write a function that receives a character and check whether it is alphanumeric or not (*Hint: you should use ASCII code*).
8. Write functions to calculate the below formulas with *n* provided by the user (each formula is one function):
   a. $S = 1 + 2 + 3 + \cdots + n$
   b. $P = 1 \times 2 \times 3 \times \ldots \times n$
   c. $S = 1 + 2^1 + 2^2 + \cdots + 2^n$
   d. $S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \cdots + \frac{1}{2n}$
   e. $\sum_{i=1}^{n} i^2$
9. Write a function that displays the Hailstone sequence:
   - With some positive number ($n > 0$):
     1. If *n* is an even number, divide by 2.
     2. If *n* is an odd number, multiply it by 3 and add 1.
     3. Repeat the two steps above until *n* equals 1.
   - For example, choose n = 5

```
C:\Users\Admin\Desktop\Java Exercise\Solve>java Hailstone
5
5 is odd, so we take 3*n+1: 16
16 is even, so we take n/2: 8
8 is even, so we take n/2: 4
4 is even, so we take n/2: 2
2 is even, so we take n/2: 1
```

10. Write a function to sum the first digit and the last digit of a number.
11. Write a function to count the number of digits in a number.
12. Write a function to reverse the input integer number.
13. Write a function to check whether a number is a palindrome or not.

14. Write a program to simulate a vending machine. This is the output of the demo program, you can use this output to build your program *(Hint: you should use **while** statement and **switch…case…** statement)*:

```
PS D:\Work\Teaching\HK2.2021\LTHDT\TestBuildLab\Lab01\VendingMachine> java VendingMachine
----Menu----
1. Coca
2. Pepsi
3. Sprite
4. Snack
5. Shutdown Machine
Please enter the number:
1
The price of Coca is: 2$, please enter the amount of money:
5
Your change is 3.0$.
----Menu----
1. Coca
2. Pepsi
3. Sprite
4. Snack
5. Shutdown Machine
Please enter the number:
1
The price of Coca is: 2$, please enter the amount of money:
1
Not enough money to buy this item. Please select again.
----Menu----
1. Coca
2. Pepsi
3. Sprite
4. Snack
5. Shutdown Machine
Please enter the number:
6
Please enter the valid number.
----Menu----
1. Coca
2. Pepsi
3. Sprite
4. Snack
5. Shutdown Machine
Please enter the number:
5
Machine is shutting down.
PS D:\Work\Teaching\HK2.2021\LTHDT\TestBuildLab\Lab01\VendingMachine>
```

-- THE END --