CHAPTER 1

Introduction to java programming language

Objectives

- About the Java Technology (History, a high-level language)
- What can Java Technology do?
- How can Java support platform-independence? (Java Virtual Machine, Java Platform)
- What is Java IDE? (Set up Environment Variables)
- The first Java program in the NetBeans (Structure of a Java program, End users run Java Programs, The first Java program in the NetBeans)
- Java - Basic Datatypes (Primitive Data Types, Reference/Object Data Types)
- Basic Constructs (Selection, Loops, Jump)
- Standard Input and Output
- Case study

About the Java Technology(1)

- History
  - 1990, James Gosling, Bill Joy, Patrick Naughton(Sun Microsystem) developed the Oak language for embedding programs to devices such as VCR, PDA (personal data assistant). The Oak programs require:
  - Platform independent/- Extremely reliable/ - Compact
  - 1993, interactive TV and PDA failed, Internet and Web were introduced, Sun change the Oak to an internet-development environment with a new project, named Java.
  - 1994, the Sun's HotJava Browser was introduced (written using Java). It showed the strength of Java applets and abilities to develop Java application.
  - Embedded Systems (1991 – 1994)
  - A client – side Wonder (1995 – 1997)
  - Moved into the Middle – tier (1997 – to present)
  - Future: may gain more success

About the Java Technology(2)

- The Java Programming Language is a high-level language. It's characteristics:
  - Simple
  - Object oriented
  - Distributed
  - Multithreaded
  - Dynamic linking
  - Architecture neutral
  - Portable
  - High performance
  - Robust
  - Secure

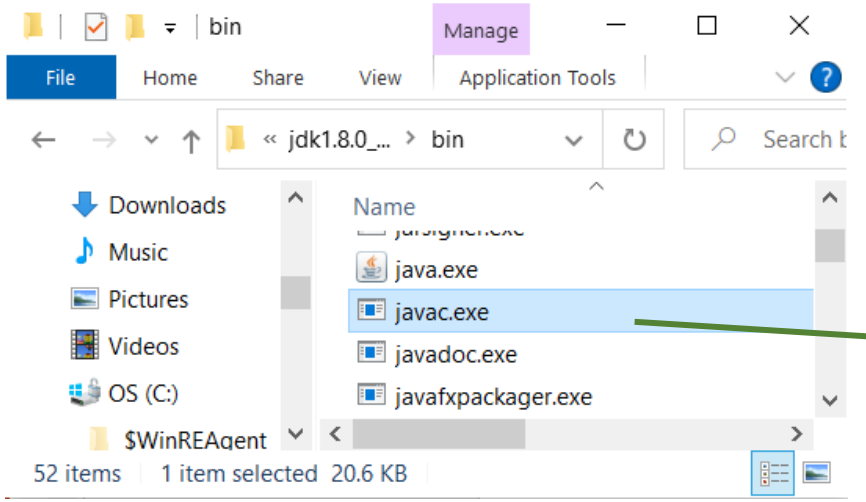
What can Java Technology do?

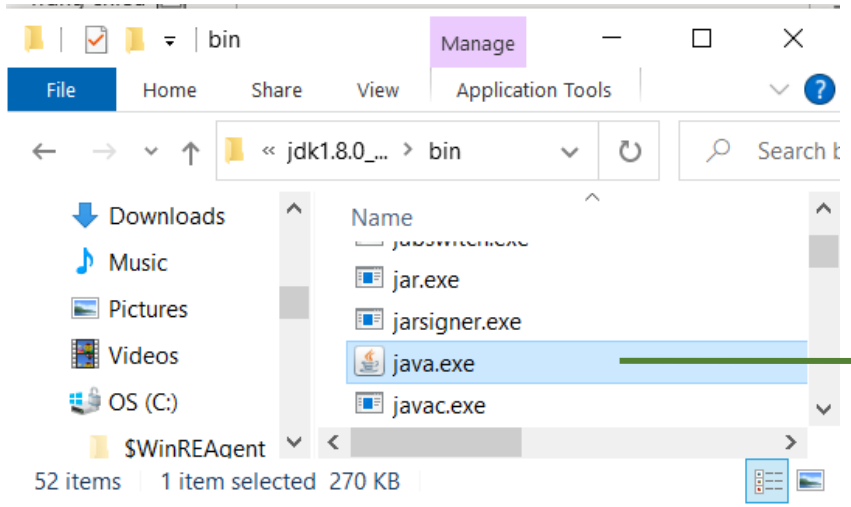
Using Java, we can:

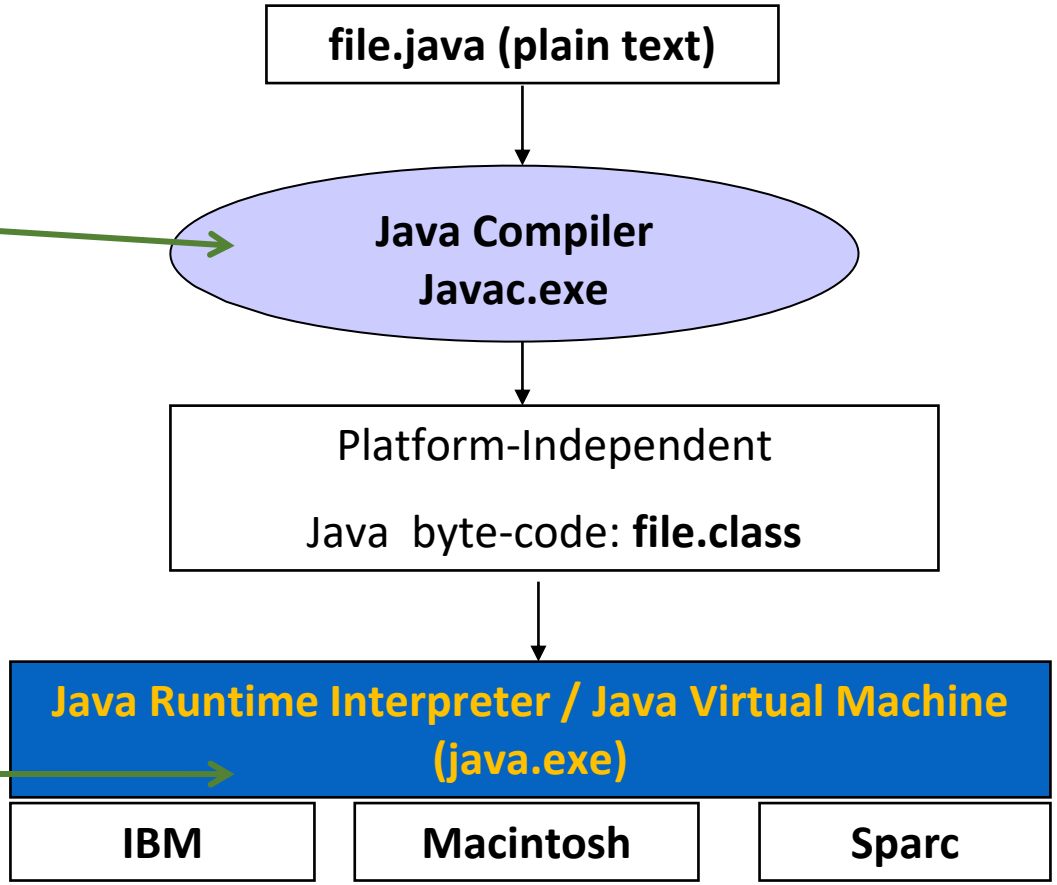
- Development Tools.
- Application Programming Interface (API).
- Deployment Technologies.
- User Interface Toolkits.
- Integration Libraries.

➔Desktop Application (Console App, GUI Apps)
➔Web-based Applications
➔ Network-based Applications
➔ Game
➔ Distributed Applications
➔ Embedding Application (Apps on Devices)

How can Java support platform-independence?

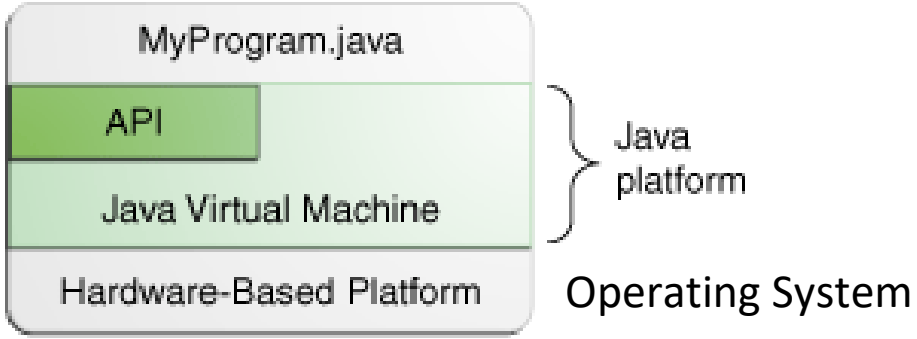






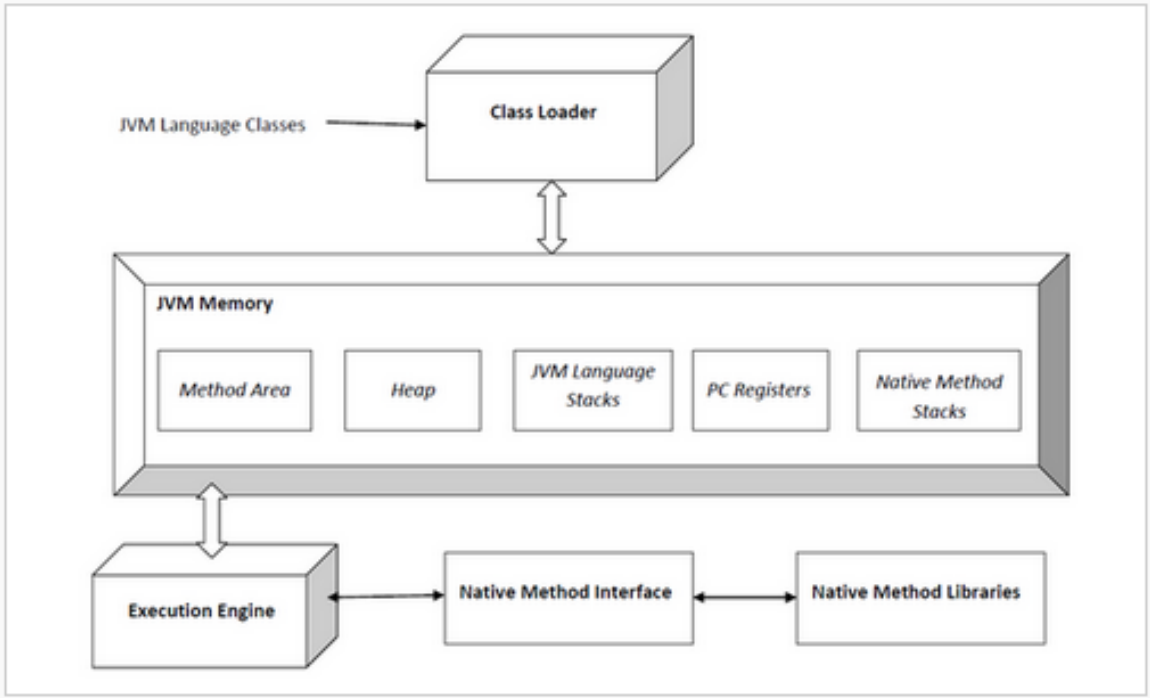
Java Platform

- A platform is the hardware or software environment in which a program runs.
- The Java platform has two components:
  - The Java Virtual Machine
  - The Java **A**pplication **P**rogramming **I**nterface (API)



Java Virtual Machine

- The Java Virtual Machine is an abstract computing machine. Like a real computing machine, it has an instruction set and manipulates various memory areas at run time. It is reasonably common to implement a programming language using a virtual machine; the best-known virtual machine may be the P-Code machine of UCSD Pascal



Overview of a Java virtual machine (JVM) architecture based on The Java Virtual Machine Specification Java SE 7 Edition

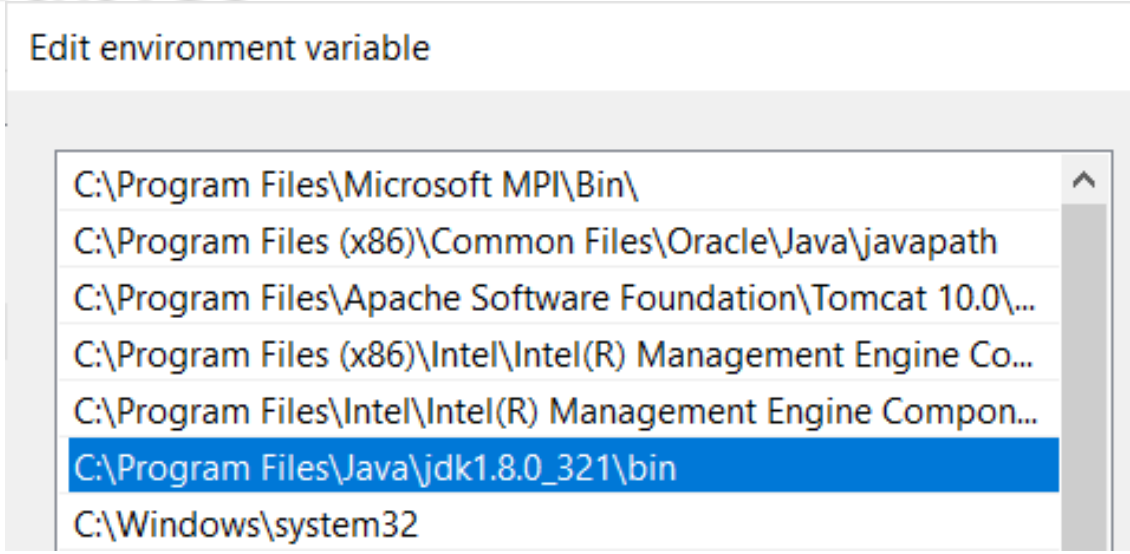
http://en.wikipedia.org/wiki/Java_virtual_machine

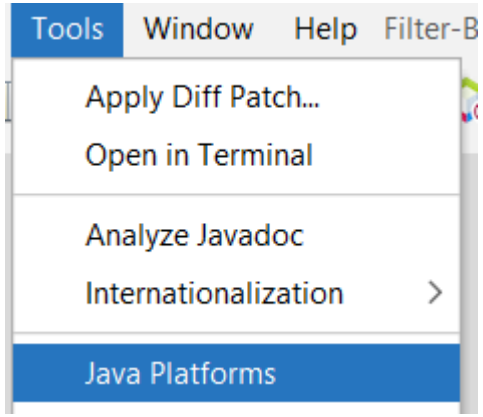
What is Java IDE?

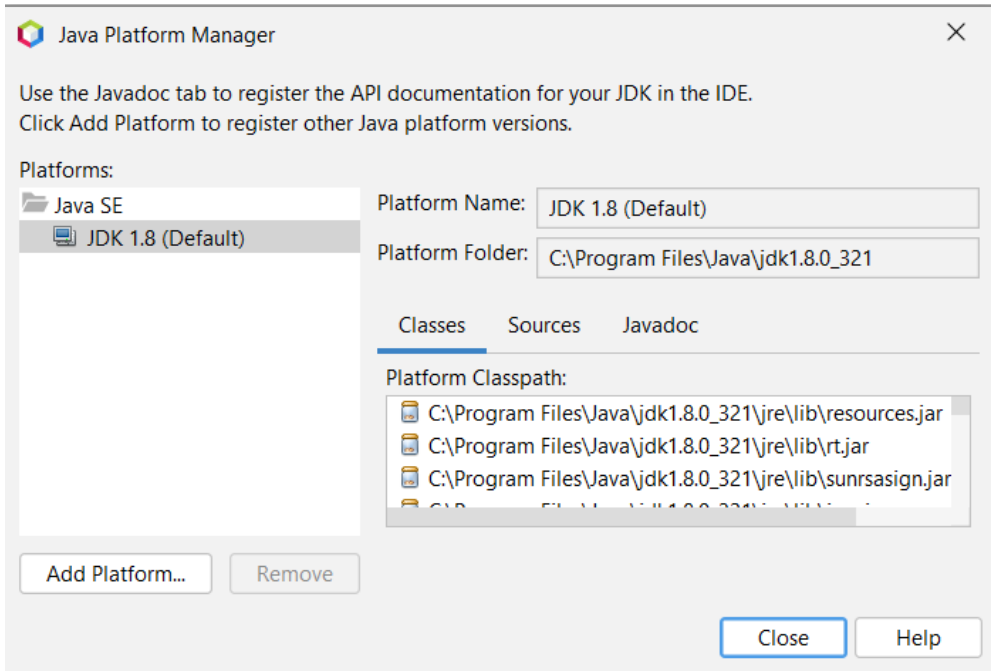
- IDE (Integrated Developement Environment). Top 10 Java IDEs
  - Eclipse (http://www.eclipse.org)
  - NetBeans (http://netbeans.org)
  - IntelliJ IDEA
  - BlueJ
  - ……

Set up Environment Variables

- After installing JavaSE (Java Development Kit Standard Edition), environment variables should be setup to point to the folder in which JavaSE is installed.
- Steps: My Computer/ Properties/ Advanced/Environment Variables/System Variables/ Path/ Edit



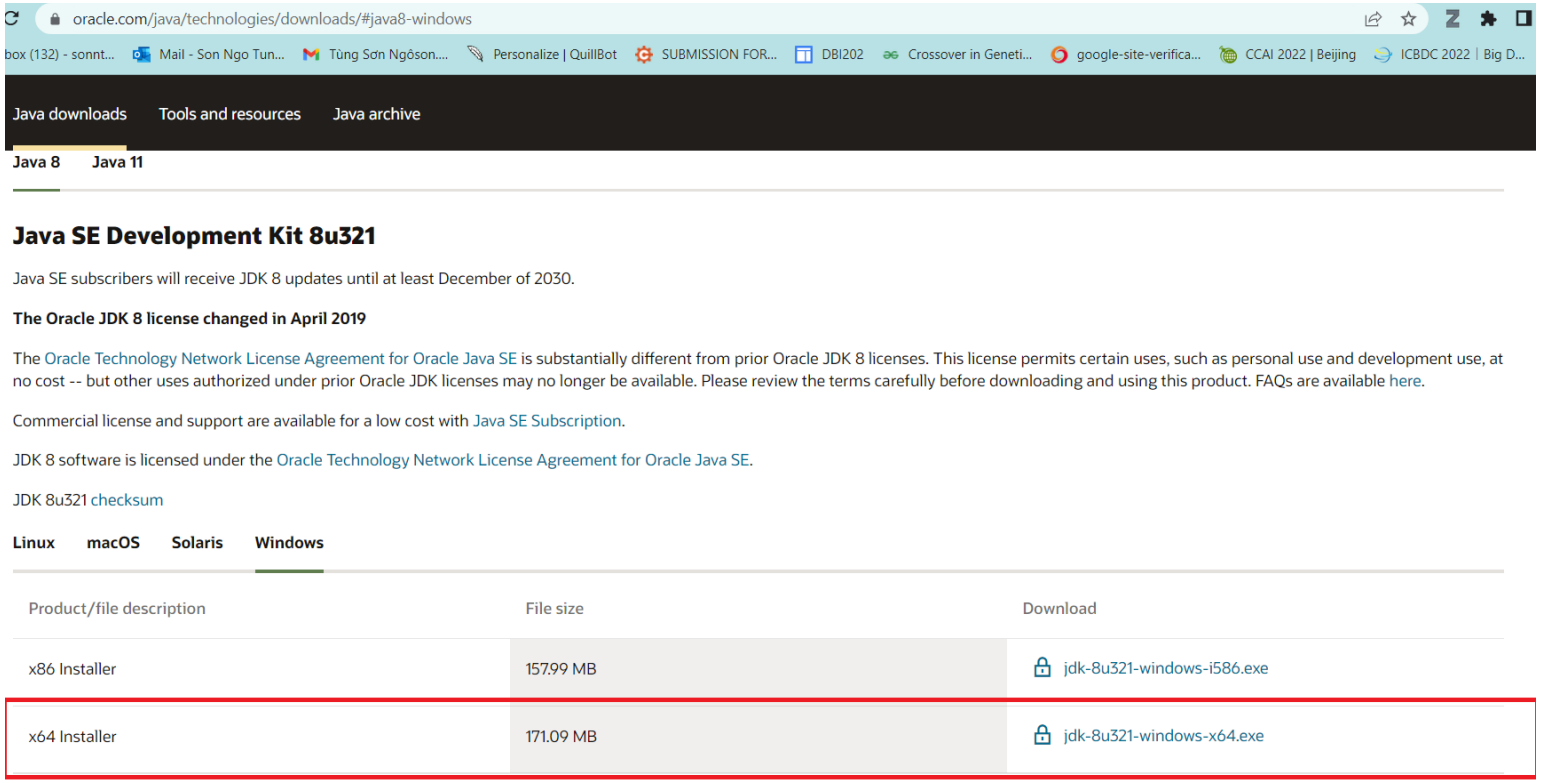




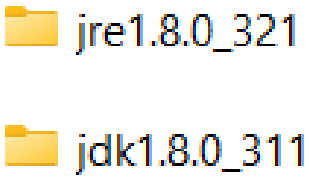
Instructions for setting up the environment for java

- Installing JDK 8u321
- Page: https://www.oracle.com/java/technologies/downloads/

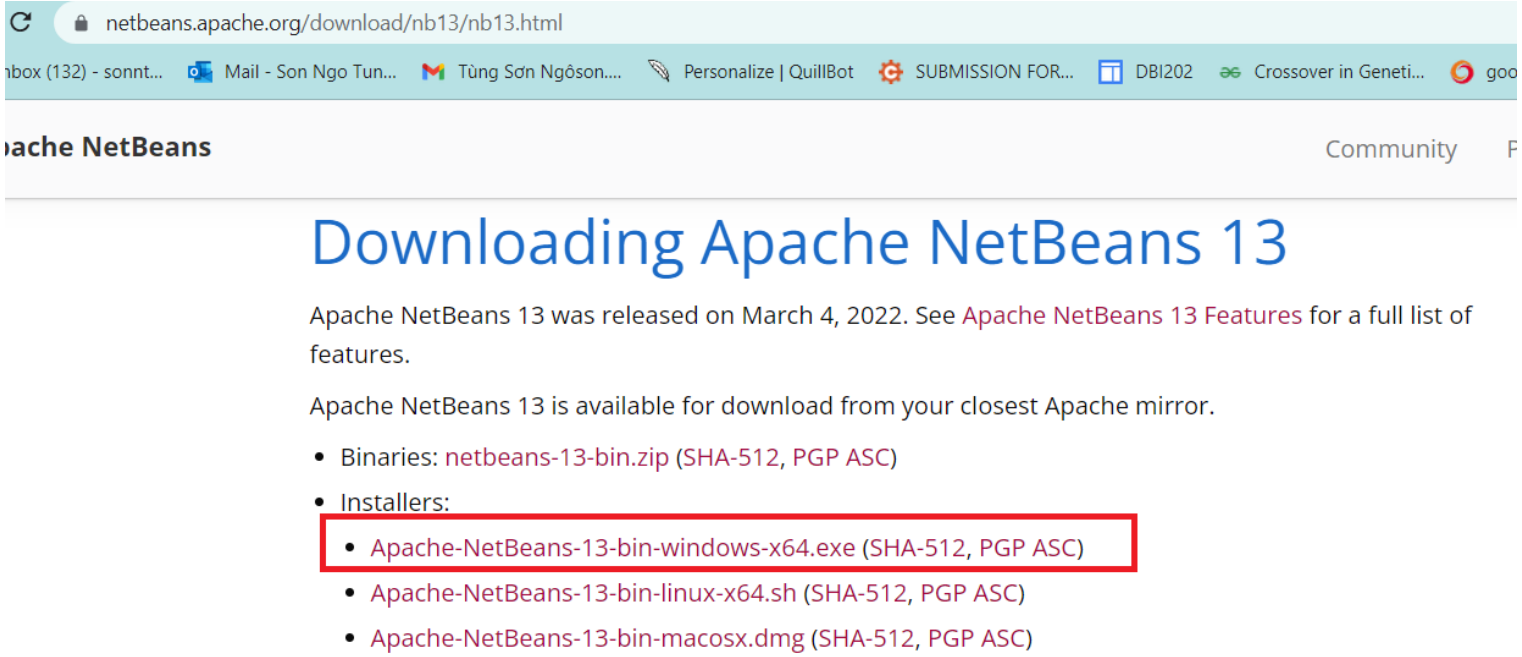
  Scroll down to find the version you need.



- After installing, let check the installation folder:
  - Normally it will be located at: C:\Program Files\Java. It contains 2 folders JDK and JRE.



- Installing Apache NetBeans IDE 13.
- Download: https://netbeans.apache.org/download/nb13/nb13.html



- During your installation, remember to select JDK 1.8 (just in case, you have more than one version of Java).
- After installation, run netbeans 13, File > New Project > Java with Ant > Java Application

The first Java program in the NetBeans

This program will show the string “Hello World” to the screen.

Steps

1- Create a new Java NetBeans project

2- Add a Java class

3- Write code

4- Compile/Run the program

JDK Editions

- Java Standard Edition (J2SE)
- Java Enterprise Edition (J2EE)
- Java Micro Edition (J2ME)
- Links for reading
  - Java tutorial: https://docs.oracle.com/javase/tutorial/
  - data type and java platform: http://www.tutorialspoint.com/java/java_basic_datatypes.htm
  - https://www.javatpoint.com/java-tutorial

Keywords and Identifiers

- Keywords: Almost of them are similar to those in C language
- Java is a case-sensitive language
- Identifiers must be different to keywords
- Naming Convention:
  - All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
  - After the first character, identifiers can have any combination of characters.
  - Class and Interface should start with the uppercase letter. They should be a noun for Class and an adjective for Interface.
  - Method should start with lowercase letter, it should be a verb. If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter.
  - Variable should start with a lowercase letter, it should not start with the special characters like & (ampersand), $ (dollar), _ (underscore). If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter.
  - Package should be a lowercase letter.
  - Constant should be in uppercase letters. If the name contains multiple words, it should be separated by an underscore(_).

Primitive Data Types - Variables

- A primitive is a simple non-object data type that represents a single value.
- Java's primitive data types are:

Type var [=Initial value] ;

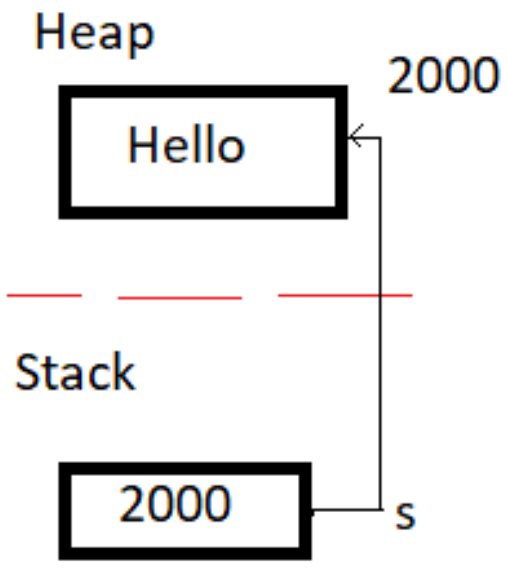
Type	Bytes	Minimum	Maximum
char	2	\u0000	\uFFFF
byte	1	-2^7	$2^7 - 1$
short	2	-2^{15}	$2^{15} – 1$
int	4	-2^{31}	$2^{31} – 1$
long	8	-2^{63}	$2^{63} - 1$
float	4		
double	8		
boolean	true/false		

Reference Data Types - Variables

- A reference data type contains reference/address of dynamically created objects.
- reference types in Java:
  - Class types.
  - Array types
  - Interface types
- Default value of any reference variable is null

for example:

```
String s=new String("Hello");
```



Operators

Category (Descending Precedence)	Operators
Unary	++ -- + - ! ~ (type)
Arithmetic	* / % + -
Shift	<< >> >>>
Comparison	< <= > >= instanceof == !=
Bitwise	& ^ |
Short-circuit	&& ||
Conditional	? :
Assignment	= op=

They are the same with those in C language

Using Operators Demonstration (1)

UseOps.java

```java
public class UseOps {
  public static void main(String[] args)
  {  int x=-1;
     System.out.println("-1<<1: " + (x<<1) );
     System.out.println("-1>>1: " + (x>>1) );
     System.out.println("-1>>>1: " + (x>>>1));
     System.out.println("3|4: " + (3|4));
     System.out.println("3&4: " + (3&4));
     System.out.println("3^4: " + (3^4));
     String S="Hello";
     boolean result= S instanceof String;
     System.out.println("Hello is an instance of String: " + result);
  }
}
```

Output - Chapter01 (run)

```
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Using Operators Demonstration (2)

Use 2 bytes to store value

```
Output - Chapter01 (run)
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

1: → 0000 0000 0000 0001
1111 1111 1111 1110 (1-complement)
-1 → 1111 1111 1111 1111 (2-complement)
-1 <<1 → 1111 1111 1111 1110 (-2)

-1 → 1111 1111 1111 1111
-1 >>1 → 1111 1111 1111 1111

-1 → 1111 1111 1111 1111
-1 >>>1 → 0111 1111 1111 1111 (2147483647)

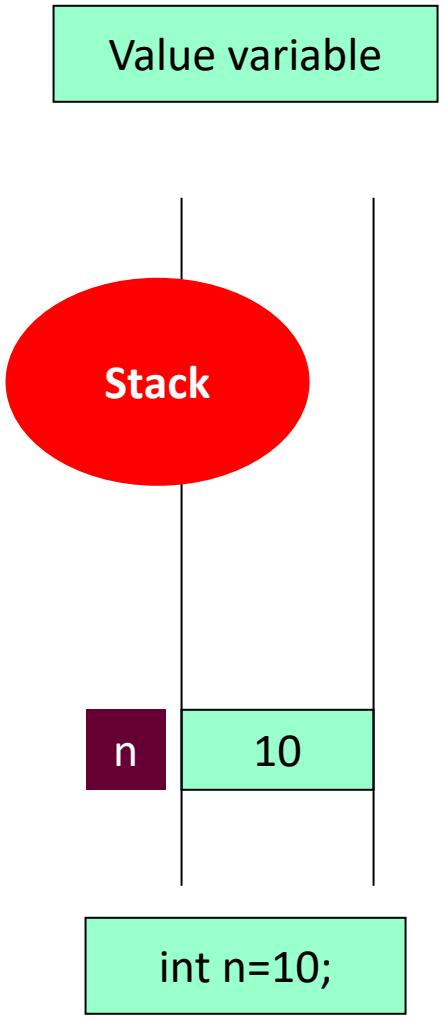
3 → 0000 0000 0000 0011
4 → 0000 0000 0000 0100
3|4 → 0000 0000 0000 0111 (7)

3 → 0000 0000 0000 0011
4 → 0000 0000 0000 0100
3&4 → 0000 0000 0000 0000 (0)

3 → 0000 0000 0000 0011
4 → 0000 0000 0000 0100
3^4 → 0000 0000 0000 0111 (7): XOR BIT

Literals and Value Variables

- Character: ‘a’
- String: String S=“Hello”;
- Escape sequences: see the page 10
- Integral literals: 28, 0x1c, 0X1A (default: int).
  123l, 123L (long)
- Floating point:
  - 1.234 (default: double)
  - 1.3f 1.3F
  - 1.3E+21
  - 1.3d 1.3D



Type Casting

- Assigning a value of one type to a variable of another type is known as **Type Casting**. In Java, type casting is classified into **two** types,
- Widening Casting(Implicit)

- Narrowing Casting(Explicitly done)

Example casting

- double d = 100.04;
- long l = (long)d; //explicit type casting required int i = (int)l; //explicit type casting required

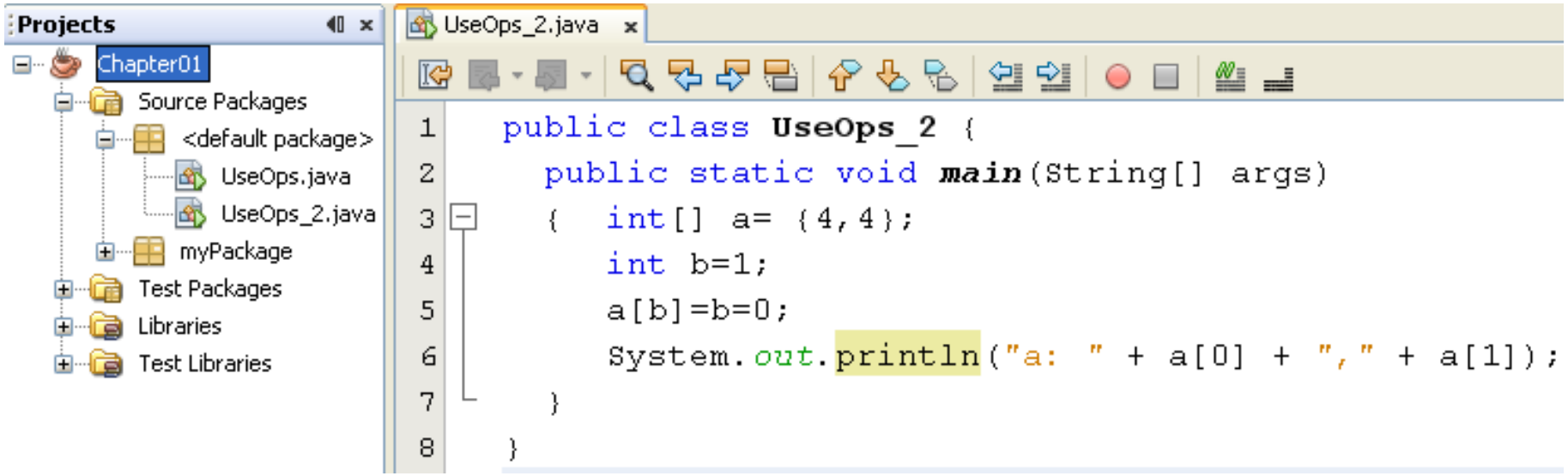
int a, b;
short c;
a = b + (int)c;
	int d;
	short e;
	e = (short)d;
		double f;
		long g;
		f = g;
		g = f; //error

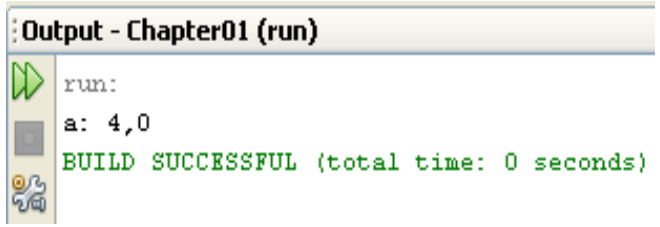
Evaluating Expressions and Operator Precedence

- The compiler generally evaluates such expressions from the innermost to outermost parentheses, left to right.

```
int x = 1; int y = 2; int z = 3;
int answer = ((8 * (y + z)) + y) * x;
would be evaluated piece by piece as follows:
((8 * (y + z) ) + y) * x
((8 * 5) + y) * x
(40 + y) * x
42 * x
42
```

Operator Precedence- Evaluation Order





Order:

(1) [ ] ➔ a[b] ➔ a[1]

(2) = (from the right) ➔ b=0 ➔ return 0

➔ a[1] = 0

Basic Constructs

- They are taken from C-language
- Selection
  - if, if … else
  - switch (char/int exp)… case … default…
- Loops
  - for
  - do… while
  - while

The *while()* Loop and the do Loop

```
while
  (boolean_condition)
{
    statement(s);
}
```

```
do {
do_something
do_more
} while
(boolean_condition);
```

```
int number = 1;
 while (number <= 200) {
    System.out.print(number + " ");
      number *= 2;
 }
```

Output:

```
1 2 4 8 16 32 64 128
```

```
Random rand = new Random();
int die;
do {
    die = rand.nextInt();
} while (die == 3);
```

The *for()* Loop

```
for (start_expr;
 test_expr;increment_expr){
  // code to execute repeatedly
  }
```

- ```
  for (int index = 0; index < 10;
  index++) {
   System.out.println(index);
   }
  ```
- ```
  for (int i = -3; i <= 2; i++) {
      System.out.println(i);
    }
  ```
- ```
  for (int i = 3; i >= 1; i--) {
      System.out.println(i);
   }
  ```
  ```

Enhanced *for* Loops

- Java's for loops were enhanced in release 1.5 to work more easily with **arrays** and **collections**.
- Syntax:

  for (type variable_name:array)

```
int sumOfLengths(String[] strings) {
    int totalLength = 0;
    for (String s:strings)
        totalLength += s.length();
    return totalLength;
}
```

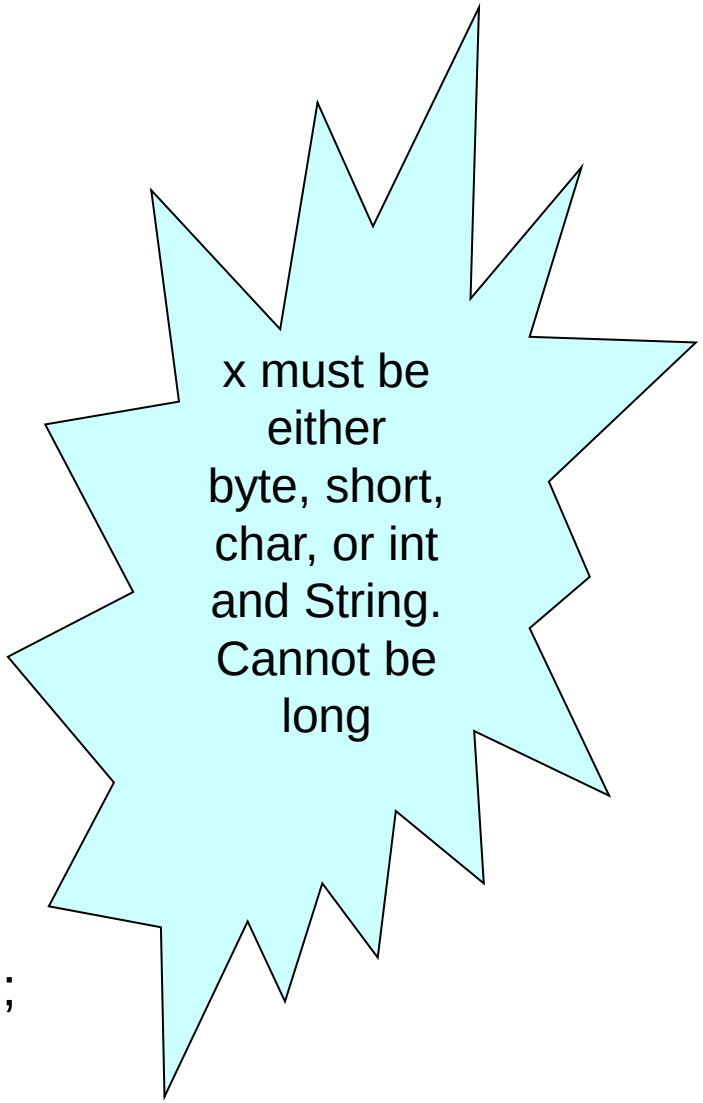
Example

```
public class Example{
  public static void main(String args[]){
     int [] numbers = {10, 20, 30, 40, 50};
     for(int x : numbers ){
         System.out.print( x );// numbers[i]
         System.out.print(",");
     }
     System.out.print("\n");
    String [] names ={"James", "Larry", "Tom",
 "Lacy"};
     for( String name : names ) {
         System.out.print( name );
         System.out.print(",");
     } } }
```

The Selection Statements

- **The *if()/else* Construct**
- **The *switch()* Construct**

```
switch (x) {
        case 1:
                System.out.println("Got a 1");
                break;
        case 2:
        case 3:
                System.out.println("Got 2 or 3");
                break;
        default:
                System.out.println("Not a 1, 2, or 3");
                break;
}
```



The *continue* Statements in Loops (for, while, do)

```
for( .;.;.)
{
    //process part 1
    if(condition)
      continue;
    //process part 2
}
```

```
mainloop:for(.;.;.){
   for( .;.;.)
   {
      //process part 1
      if(boolean_exp)
        continue mainloop;
      //process part 2
   }
}
```

continue

```
for (i=0; i<=5; ++i) {
        if (i % 2 == 0)
                continue;
        System.out.println("This is a " + i + " iteration");
}
```

```
i = 0;
 while (i <= 5)  {
        ++i;
        if (i % 2) == 0)
                continue;
        System.out.println("This is a odd iteration - " + i);
 }
```

The *break* Statements in Loops (for, while, do)

```
for( .;.;.)
{
    //process part 1
    if(condition)
      break;
    //process part 2
}
```

```
int i = 1;
 while (true) {
        if (i = = 3)
                break;
        System.out.println("This is a " + i
             + " iteration");
        ++i;
 }
```

The String type

- A String represents a sequence of zero or more Unicode characters.
- String name = "Steve";
- String s = “”;
- String s = null;
- String concatenation.
- String x = "foo“ + "bar“ + "!";
- Java is a case-sensitive language.

Standard Input and Output

- import java.util.Scanner;
- Scanner input = new Scanner(System.in);
- public String next()
- public String nextLine()
- public byte nextByte()
- public short nextShort()
- public int nextInt()
- public long nextLong()
- public float nextFloat()
- public double nextDouble()

Problem with .nextLine() in Java

- System.out.println("Enter numerical value");

```
int option;
option = input.nextInt();
System.out.println("Enter 1st string");
String string1 = input.nextLine();
System.out.println("Enter 2nd string");
String string2 = input.nextLine();
```

- Solution:

```
int option = input.nextInt();
input.nextLine(); // Consume newline left-over
String str1 = input.nextLine();
int option = Integer.parseInt(input.nextLine());
```

Example

InputOutputDemo.java

```java
/* Write a program that will accept an array of intergers then
   print out entered value and the sum of values
 */
import java.util.Scanner;
public class InputOutputDemo {
   public static void main (String args[])
   {  int a[]; // array of integers
      int n ;  // number of elements of the array
      int i;   // variable for traversing the array
      Scanner sc= new Scanner(System.in); // object for the keyboard
      System.out.print("Enter number of elements: ");
      n = Integer.parseInt(sc.nextLine());
      a = new int[n]; // mem. allocating for elements of the array
      for (i=0;i<n;i++)
      {  System.out.print("Enter the " + (i+1) + "/" + n + " element: ");
         a[i]=Integer.parseInt(sc.nextLine());
      }
      System.out.print("Entered values: ");
      for (i=0;i<n;i++) System.out.format("%5d", a[i]);
      int S=0;
      for (int x: a) S+=x;
      System.out.println("\nSum of values: " + S);
   }
}
```

Refer to Java documentation:
java.lang.String class,
- the **format** method,
  - format string

for more details

n= sc.nextInt();

Output - Chapter01 (run) #2

```
run:
Enter number of elements: 5
Enter the 1/5 element: 1
Enter the 2/5 element: 4
Enter the 3/5 element: 2
Enter the 4/5 element: 0
Enter the 5/5 element: 7
Entered values:     1    4    2    0    7
Sum of values: 14
BUILD SUCCESSFUL (total time: 11 seconds)
```

Summary

- An overview of Java technology as a whole.
- What to download, what to install, and what to type, for creating a simple "Hello World!" application.
- Study some fundamentals of Java languages: Data types, variables, arrays, operators, logic constructs.
- The traditional features of the language, including: variables, data types, operators, and control flow.
- Standard Input and Output

Case study

- Using simple menu
- https://code.ptit.edu.vn/student/question (from 1 to 25)
- More: https://codelearn.io/learning/java-fundamentals