LẬP TRÌNH TRONG CNTT VỚI JAVA

GV: Ths. Đỗ Hà Phương

Thời lượng: 45LT+30TH (1TK – 1 TH - 1GK – 1CK)

Case study 1:

- Computer architectures: Instruction Set Architecture, CPU, MEMORY, BUS
- OS: Single-task/multi-task, singe-user/multi-user, 32-bit application/64-bit application, serial/parallel processing
- Computer network: TCP/IP configuration, System information

Learning Outcome (LO)

STT	LO	SO/PI
1	Hiện thực các bài toán liên quan đến danh sách liên kết, cây	
2	Hiện thực các bài toán liên quan đến cây nhị phân	
3	Hiện thực các bài toán tìm kiếm tối ưu	
4	Hiện thực một số bài toán trong Trí tuệ nhân tạo	

SẢN PHẨM MÔN HỌC

- Xây dựng và triển khai Java Application lên Google Store, CH play
- Phân nhóm sinh viên, mỗi nhóm phát triển 1 sản phẩm trên 1 java framework.

Chapter 1:

Java Primer

1.1. Java Overview

1.2. Object in Java

1.3. Expressions, Operators and Precedence

1.4. Control Flow

1.5. Functions

1.6 Simple input and Output

1.7 Exception Handling

1.8 Iterations and Generators

1.9 Additional Java Convenience

1.10 Scopes and Namespace

1.11 Modules and the Import statement

Chapter 2:

Object Oriented Programming

2.1. Goals, Principles and Patterns

2.2. Testing and debugging

2.3. Class definitions

2.4. Inheritance

2.5. Name space and Object -Orientation

2.6. Shallow and Deep copying

Chapter 3:

Recursion and Array-Based Sequences

3.1. Illustrative Examples

3.2 Analyzing Recursive Algorithms

3.3 Further Examples of Recursion

3.4 Designing Recursive Algorithms

3.5 Java’s Sequence Types

3.6 Low-level Arrays

3.7 Dynamic Arrays and Amortization

3.8 Using Array-Based Sequences

3.9 Multidimensional Data Sets

Chapter 4:

Stacks, Queues and Deques

4.1. Stacks

4.2. Queues

4.3. Double-Ended Queues

Chapter 5:

Linked Lists

5.1. Simply Linked Lists

5.2. Doubly Linked Lists

5.3. The positional List ADT

5.4. Sorting a Positional List

5.5. Link-Based vs. Array-Based Sequences

Chapter 6:

Trees

6.1. General Trees

6.2 Binary Trees

6.3. Tree Traversal Algorithms

6.4. Binary Search Trees

6.5. Red-Black Trees

Chapter 7:

Sorting and Selection

7.1. Merge-Sort

7.2. Quick Sort

7.3. Java’s Built-In Sorting Functions

7.4. Selection

Chapter 8:

AI with Java

8.1. Processing the Data

8.2. Heuristic Search

8.3. Speech recognition

WHAT is JAVA?

- A Programming language by Sun Microsystem, 1995. (2010 ORACLE)
- An Object-oriented programming language
- Many advanced features

WHY use JAVA?

- Top 3 programming language rank
- 3 billion devices
- Write once run anywhere – WORA
- Free
- Supports: Community, Platform, Framework

WHERE use JAVA?

- One of the most popular program language used to create web applications and platform.

HOW use JAVA?

- JVM, JRE, JDK, IDE, APIs
- Java programming language

TIOBE
the software quality company

About us | Knowledge | News | Coding Standards | TIOBE Index | Contact

Products | Quality Models | Markets | Schedule a demo

Aug 2023	Aug 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.33%	-2.30%
2	2		C	11.41%	-3.35%
3	4	▲	C++	10.63%	+0.49%
4	3	▼	Java	10.33%	-2.14%
5	**5**		**C#**	**7.04%**	**+1.64%**
6	8	▲	JavaScript	3.29%	+0.89%

- Nguồn: https://www.tiobe.com/tiobe-index/

CODE JAVA:

- Desktop Application: J2SE (Java Application, Java Applet)
- Server Application: J2EE
- Mobile (Embedded) application: J2ME

- Online:
  - https://www.jdoodle.com/online-java-compiler/
  - https://www.w3schools.com/java/java_compiler.asp
  - https://www.programiz.com/java-programming/online-compiler/

JAVA terms:	Meaning:
JVM	Java Virtual Machine, Loading the java class file into memory and execution JVM Loads code, Verifies code, Executes code, Provides runtime environment - Interpreter
JRE	Java Runtime Environment, composed of **JVM**, **tools**, **APIs** Run / to deploy java codes or applications
JDK	Java Development Kit, composed of **JRE** and **JVM**, a platform to develop java application.
IDE	Integrated Development Environment, Platform provides a editor for writing application with detecting errors, provides proper indentation, helps in managing files etc.. Java have a lot of IDEs: **Netbeans**, **Eclipse**…
Java Platform	Composed compiler, execution engine, library.
Java Framework	The body or platform of pre-written codes used by Java developers to develop Java applications or web applications.Ex: Spring, Grails, Google Web Toolkits, …
JAVA APIs	Application Programming Interface, Ex: Beans, AWT, Swing….
SDK	Sortware Development Kit

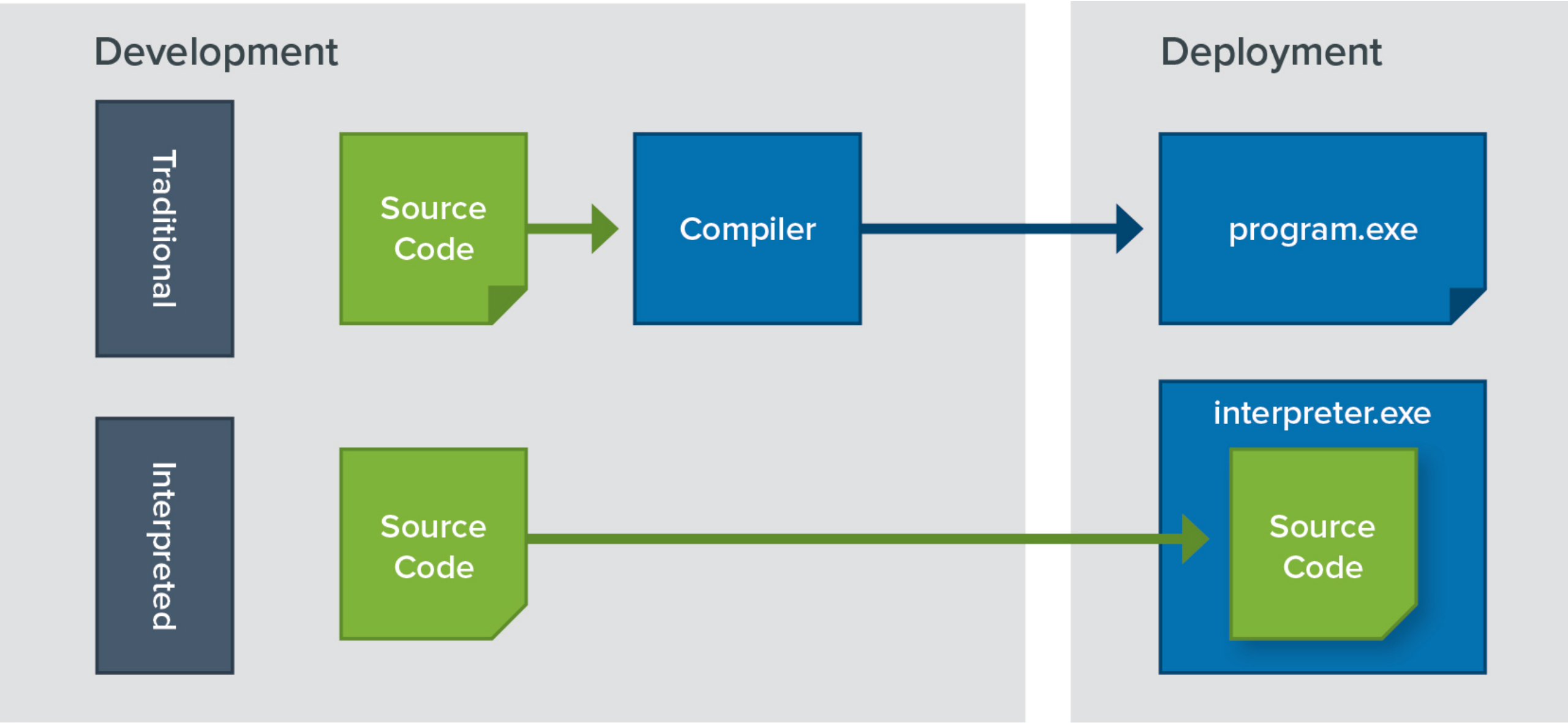

Development
Deployment
Traditional
Source Code
Compiler
program.exe
Interpreted
Source Code
interpreter.exe
Source Code

Diagram A-1

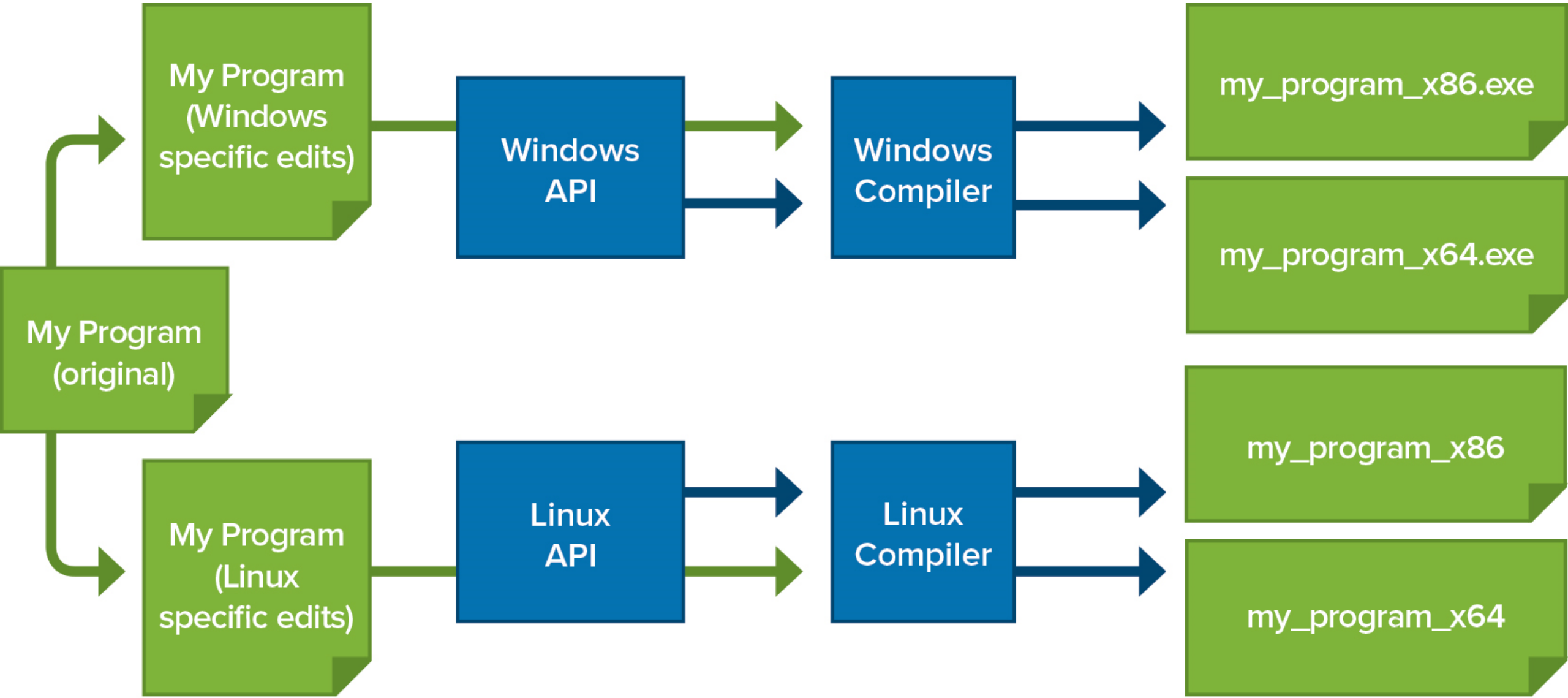
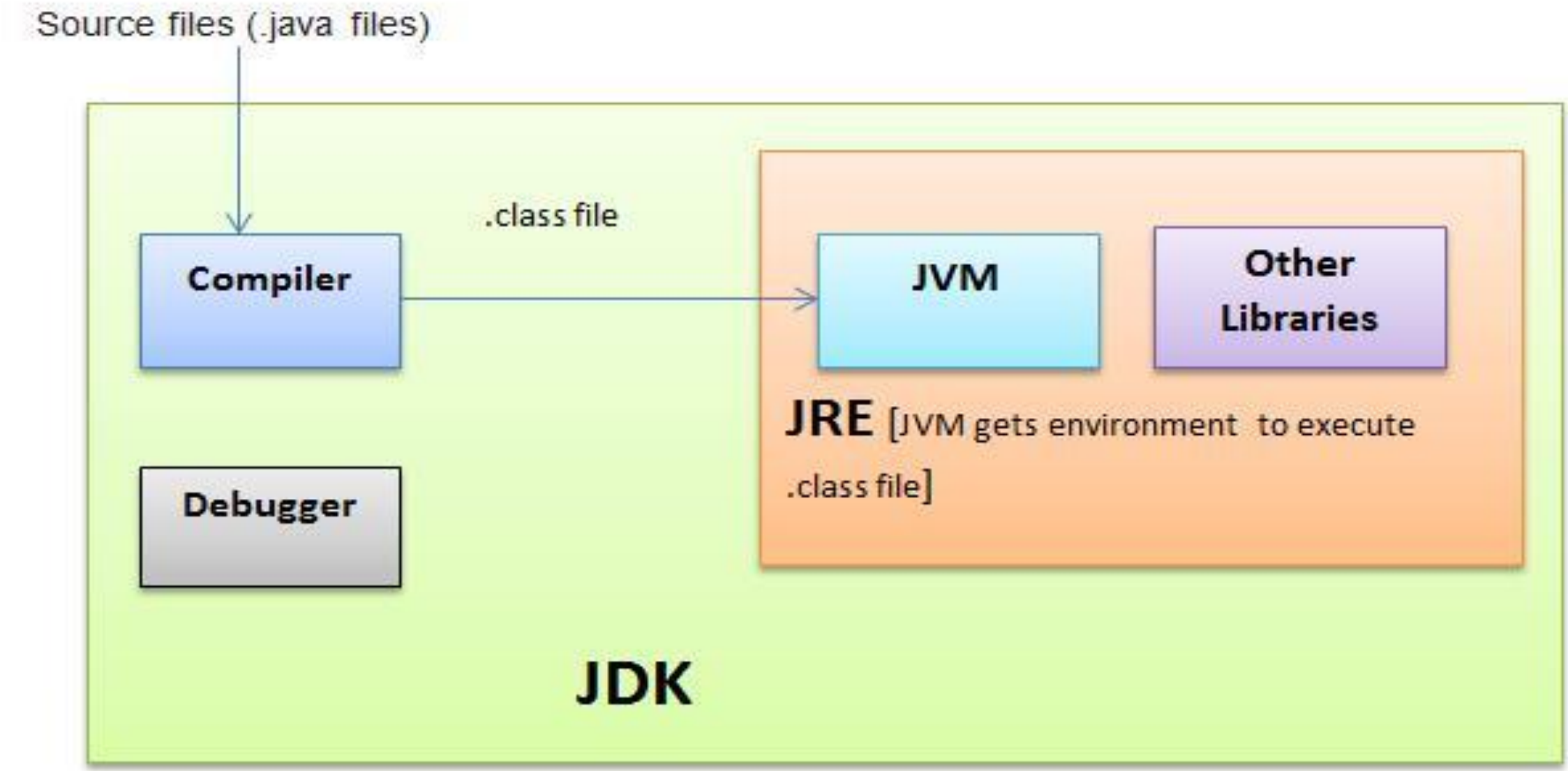
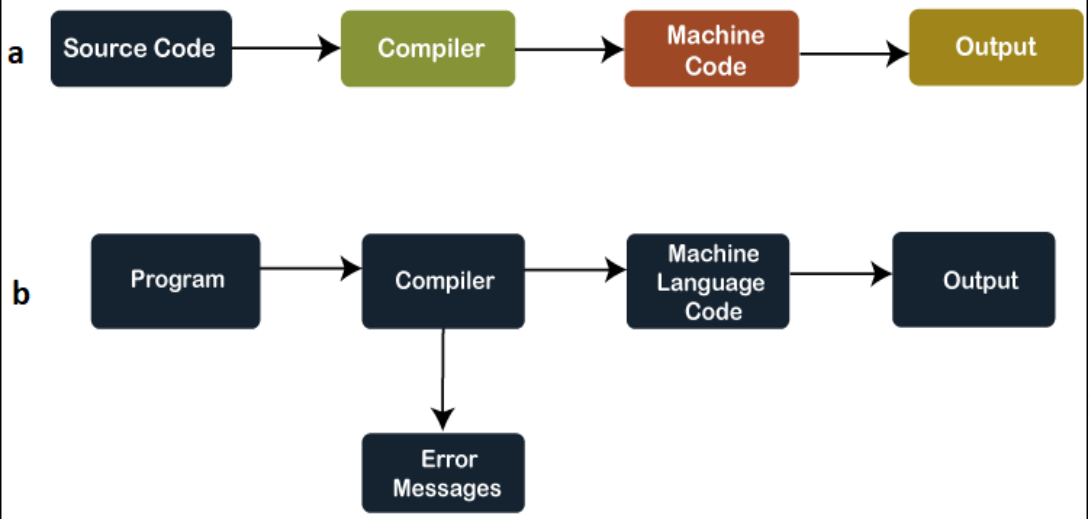

My Program (Windows specific edits)
Windows API
Windows Compiler
my_program_x86.exe
my_program_x64.exe
My Program (original)
My Program (Linux specific edits)
Linux API
Linux Compiler
my_program_x86
my_program_x64

Diagram B-1

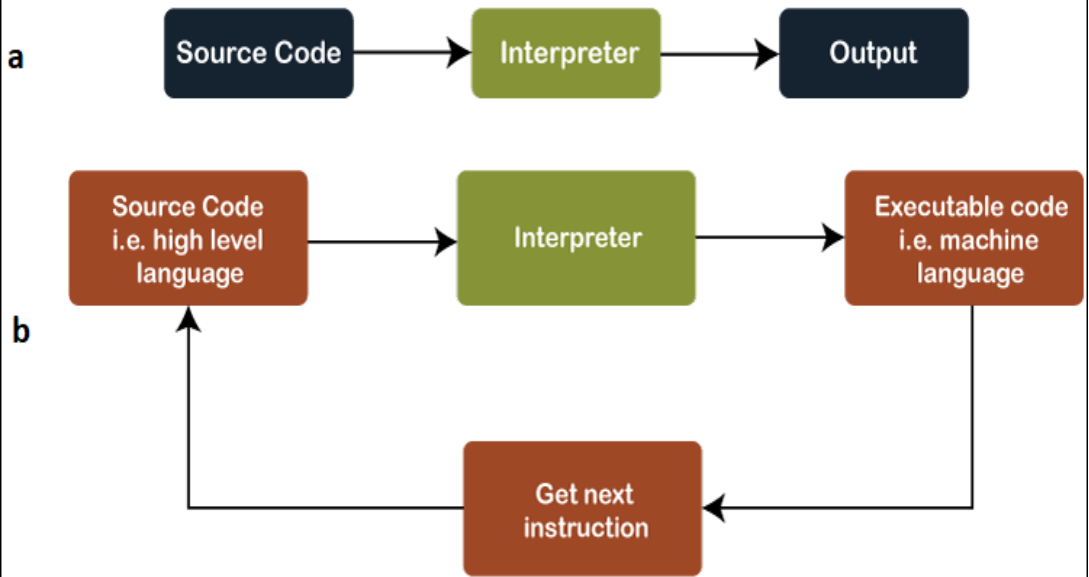


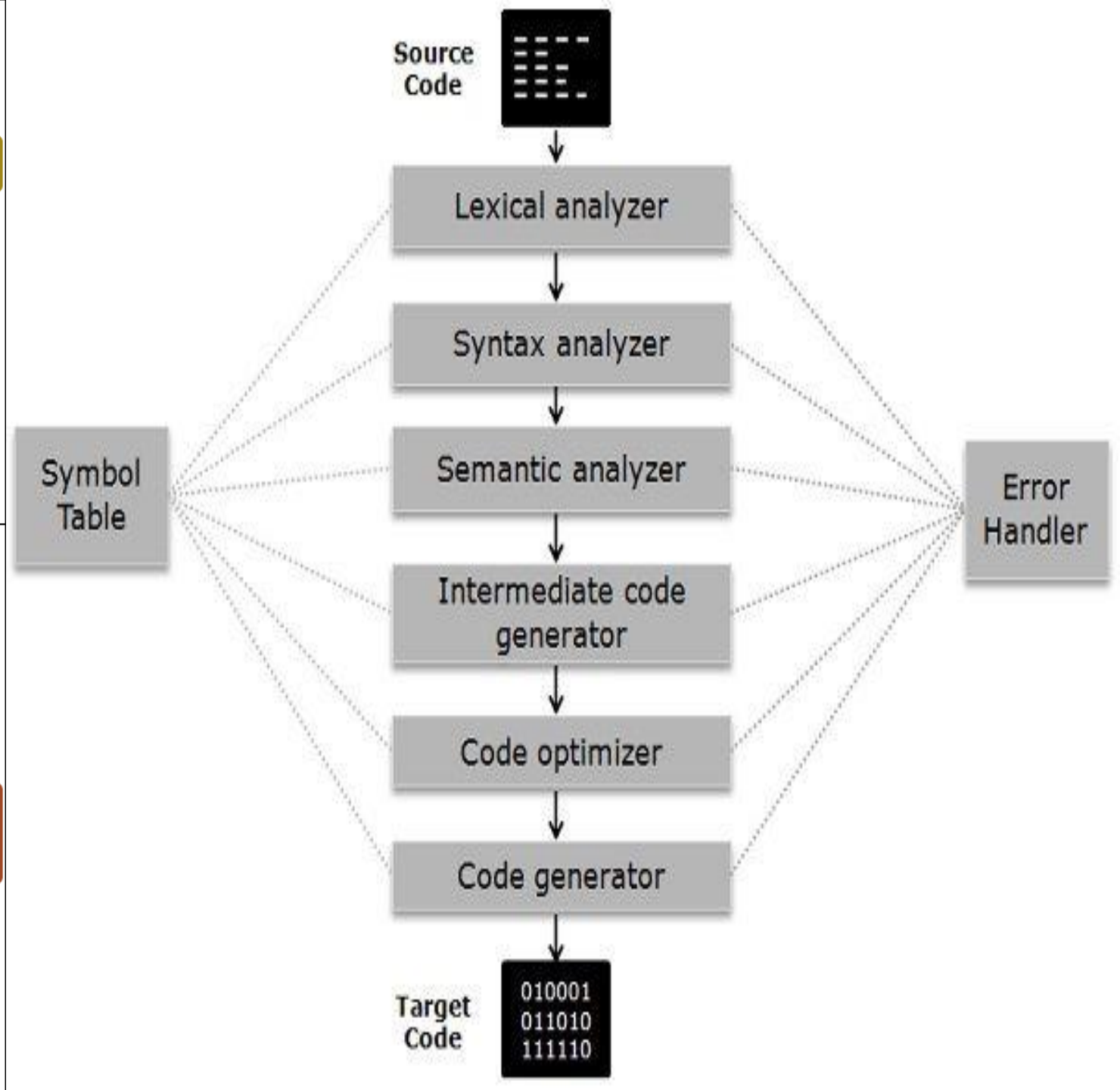
Note: **To built source code and development java application: Install IDE, JDK**
To run Java application: Install JRE

How Compiler Works



How Interpreter Works





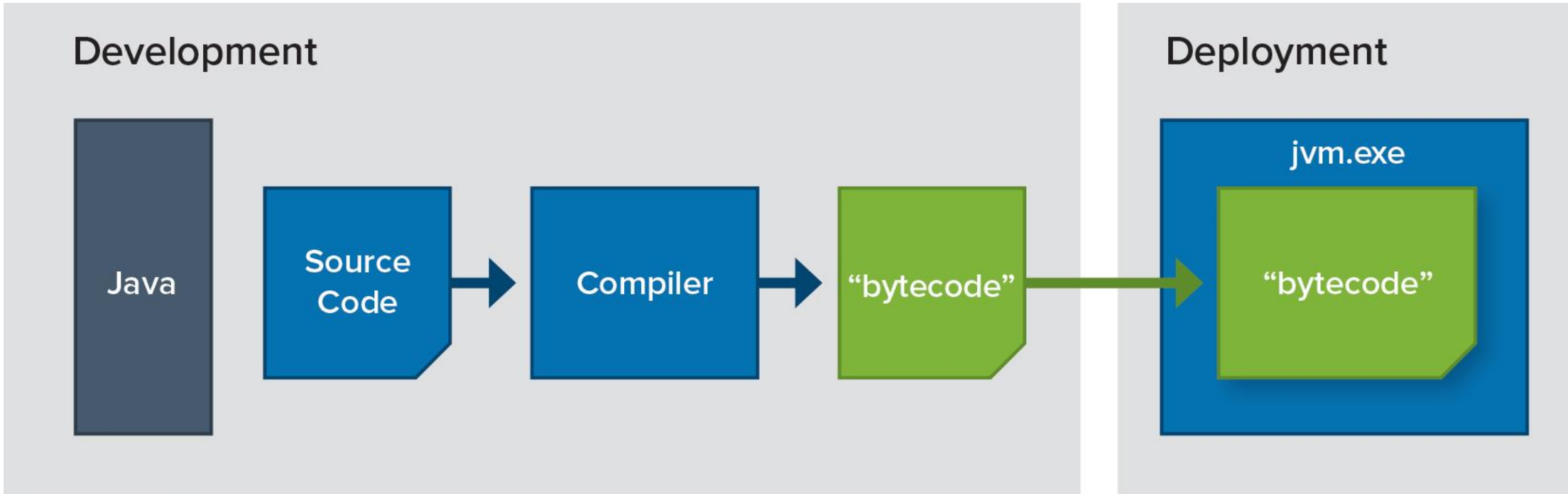

Development
Java
Source Code
Compiler
“bytecode”
Deployment
jvm.exe
“bytecode”

Diagram A-2

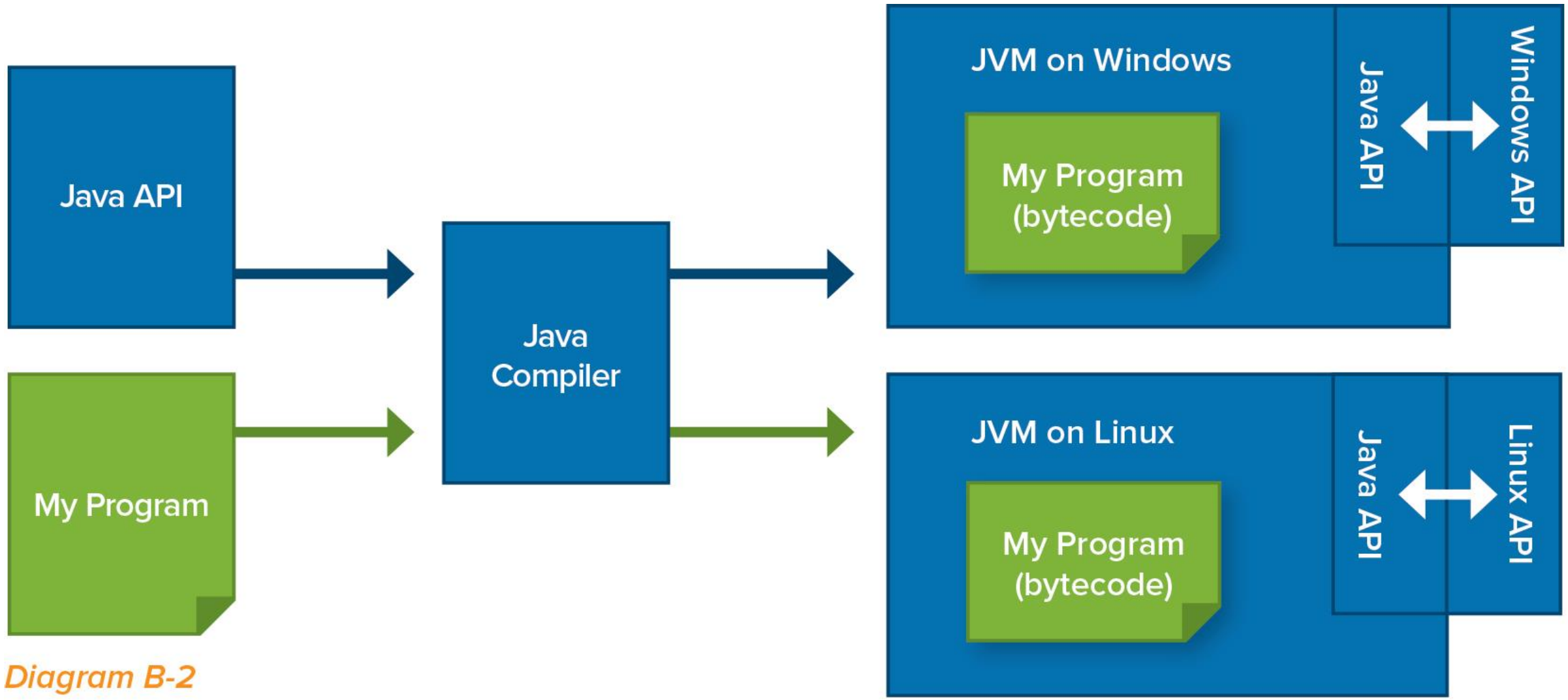
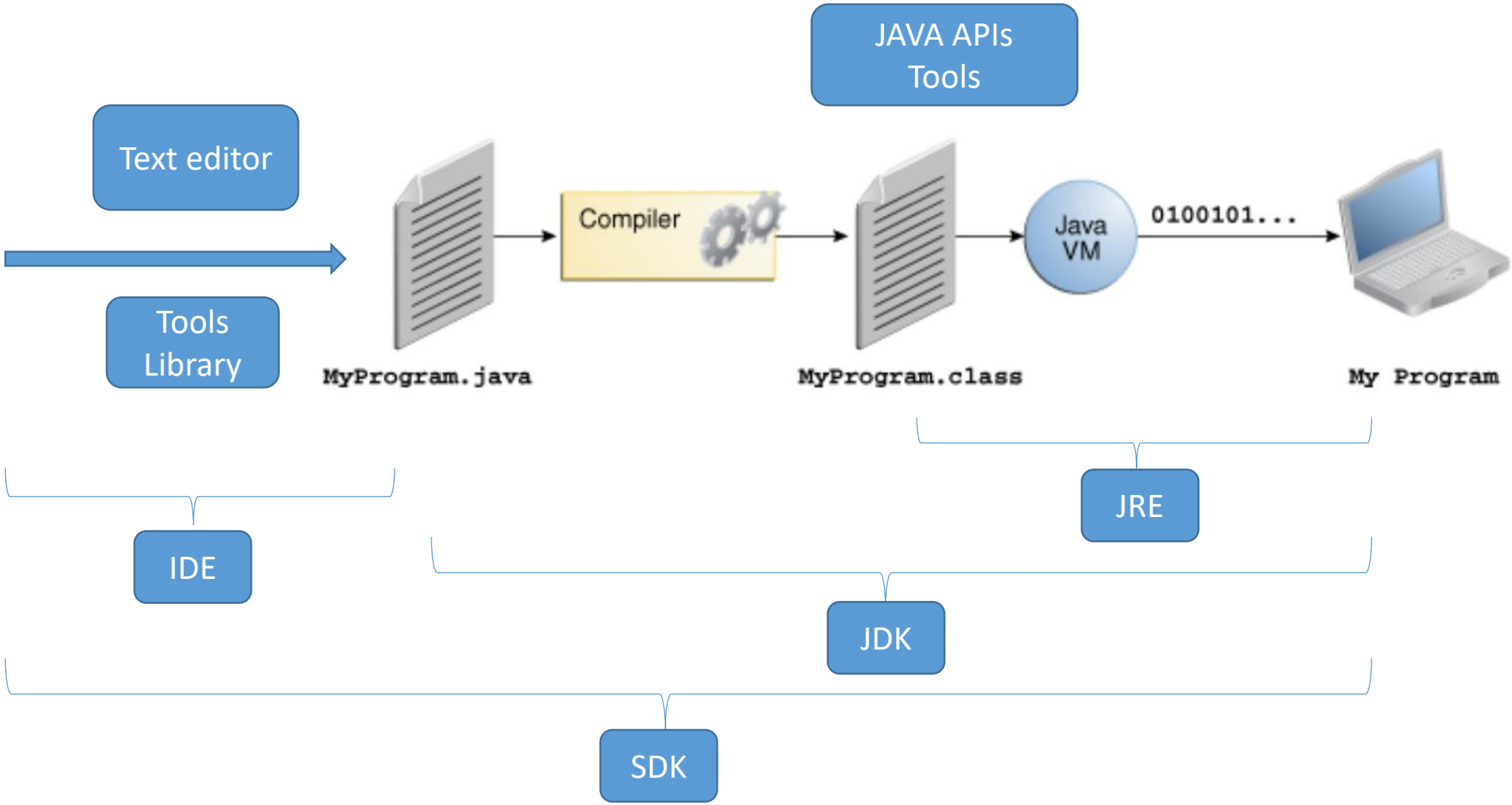
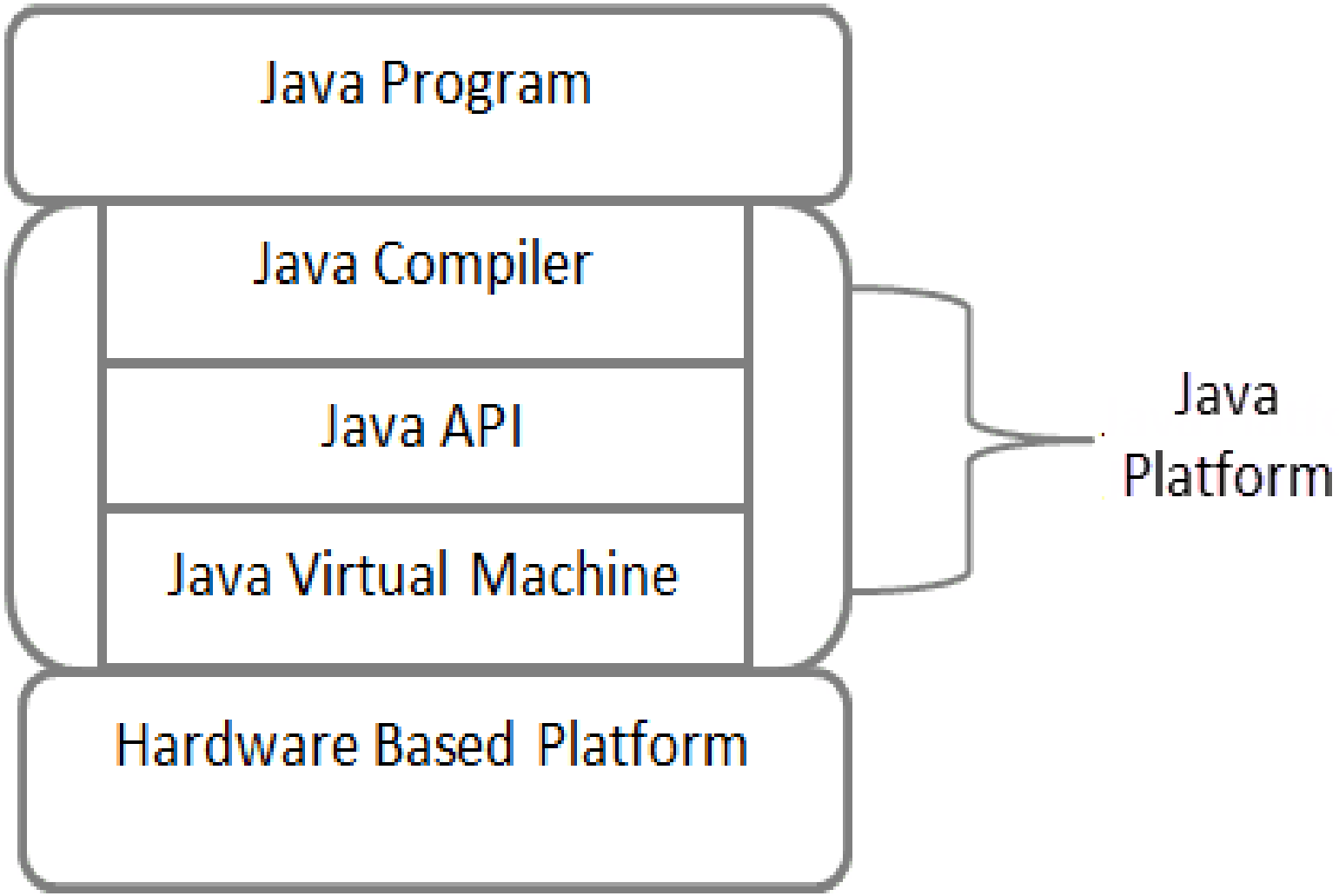

Java API
My Program
Java
Compiler
JVM on Windows
My Program
(bytecode)
Java API
Windows API
JVM on Linux
My Program
(bytecode)
Java API
Linux API

Diagram B-2


JAVA APIs
Tools
Text editor
Tools
Library
Compiler
Java
VM
0100101...
MyProgram.java
MyProgram.class
My Program
IDE
JRE
JDK
SDK


Java Program
Java Compiler
Java API
Java Virtual Machine
Hardware Based Platform
Java
Platform

Study case:

- **Library**
- **APIs**
- **Framework**

CÀI ĐẶT MÔI TRƯỜNG VÀ CÔNG CỤ CẦN THIẾT ĐỂ LẬP TRÌNH JAVA

- JDK (**J**ava SE **D**evelopment tool**K**it) – Thư viện, trình thông dịch, trình biên dịch javac, JRE chạy và thực thi các chương trình Java
  - Oracle: https://www.oracle.com/java/technologies/javase-downloads.html
  - OpenJDK: https://adoptopenjdk.net/ (cặp nhật 6 tháng 1 lần)
- IDE : để quản lý và lập trình (Eclipse, Netbeans…)
  - Tải Eclipse: https://www.eclipse.org/downloads/

JAVA PROGRAMMING LANGUAGE

Identifiers and Reserved Words in Java

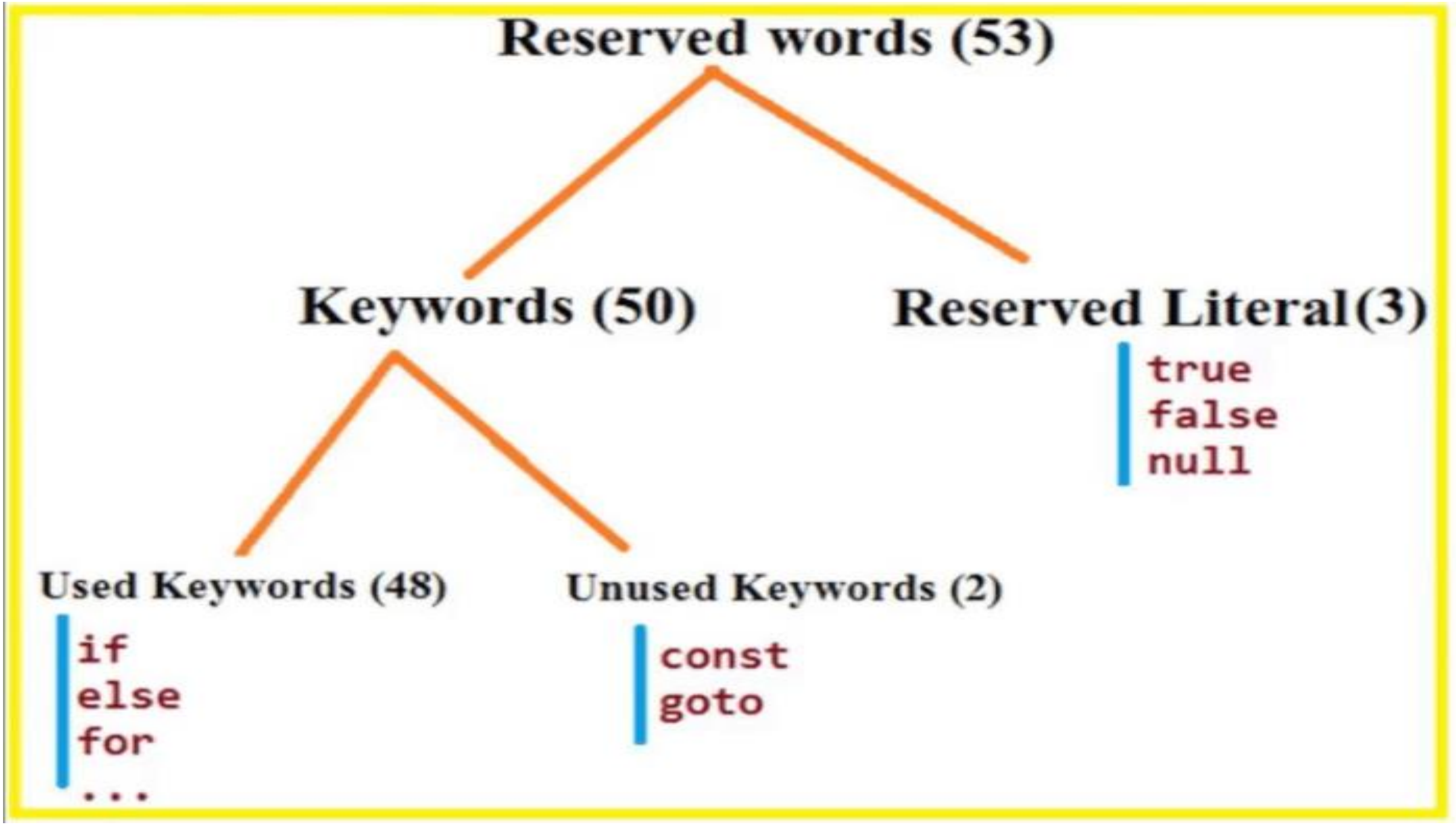
- Name used for classes, methods, interfaces, and variables are called **identifiers**

List of some valid identifiers in Java

1.MyVariable
2.myvariable
3.x
4.I
5.my_Variable
6._myvariable
7.$myvariable
8.sum_of_array
9.MYVARIABLE
10.dataflair123

List of some invalid identifiers in Java

1.My Variable	(it contains a space)
2.123gkk	(it begins with numbers)
3.a+c	(plus sign is not an alphanumeric character)
4.variable-2	(the hyphen is not allowed)
5.sum_&_difference	(ampersand is not an alphanumeric character)
6.O’Reilly	(the apostrophe is not an alphanumeric character)

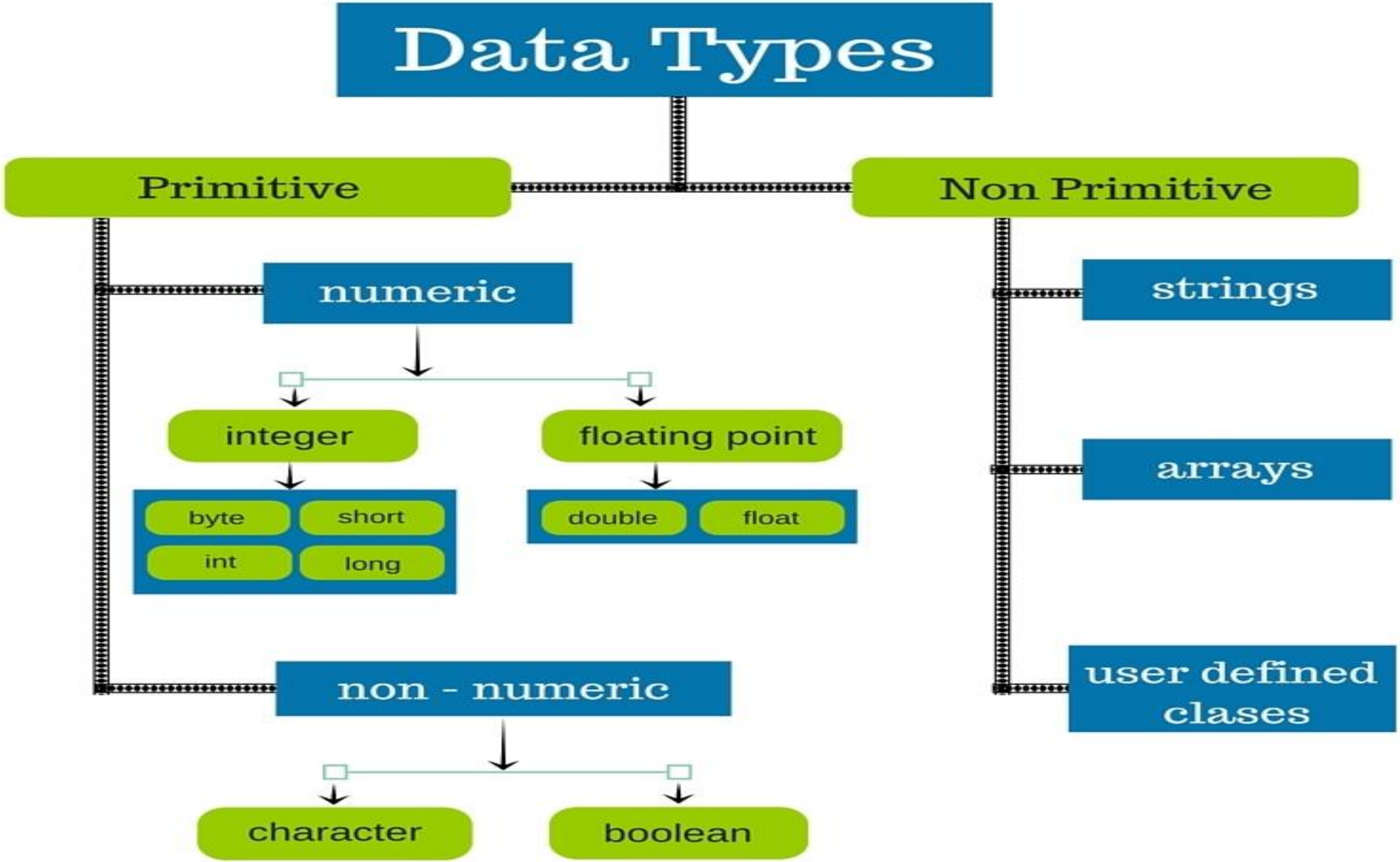

Reserved words (53)
Keywords (50)
Reserved Literal(3)
true
false
null
Used Keywords (48)
Unused Keywords (2)
if
else
for
...
const
goto

List of Java Keywords:

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	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

Variables in Java

- The Syntax for declaring a variable in java: **data_type variable_name;**
- **Types of Variables in Java:** Basically there are three types of variables in java :
  1. Local Variable
  2. Instance Variable
  3. Static Variable
- **Variables Scope and Lifetime:**
  - The **scope of a variable**
  - The **lifetime of a variable**
- **Scope vs. Lifetime in Java:**
  - **Scope** refers to the range of code where you can access a variable. We can divide scope into:
    1. Method body scope variable is accessible in method body only (local variables, parameters)
    2. Class definition scope variable is accessible in the class definition (instance variables)
  - **Lifetime** refers to the amount of time a variable (or object) exists. We can divide lifetime into categories too:
    1. Method body lifetime exists on method body lifetime entry, disappears on method body exit (local variables, parameters).
    2. Class definition lifetime exists as long as the object is around (instance variables).


Data Types
Primitive
Non Primitive
numeric
strings
integer
floating point
arrays
byte
short
int
long
double
float
non - numeric
user defined clases
character
boolean

Data Types	Valid Values	Default Values	Size	Range	Example
boolean	true, false	false	1 bit	–	boolean isEnabled = true;
byte	integer	0	8 bits	–	byte thiss =1;
char	unicode	\u0000	16 bits	–	char a =’a’;
short	integer	0	16 bits	[-32,768, 32,767]	short counter =1;
int	integer	0	32 bits	[-2,147,483,648, 2,147,483,647]	int I = 10;
long	integer	0	64 bits	[-9,223,372,036,854, 775,808, 9,223,372,036,854, 775,807]	long song = 100;
float	floating point	0.0	32 bits	–	float pi = 3.14F;
double	floating point	0.0	64 bits	–	Double team = 1e1d;

Type Casting in Java

Example :

```
int x;
double y = 2.5;
x = (int)y;
```

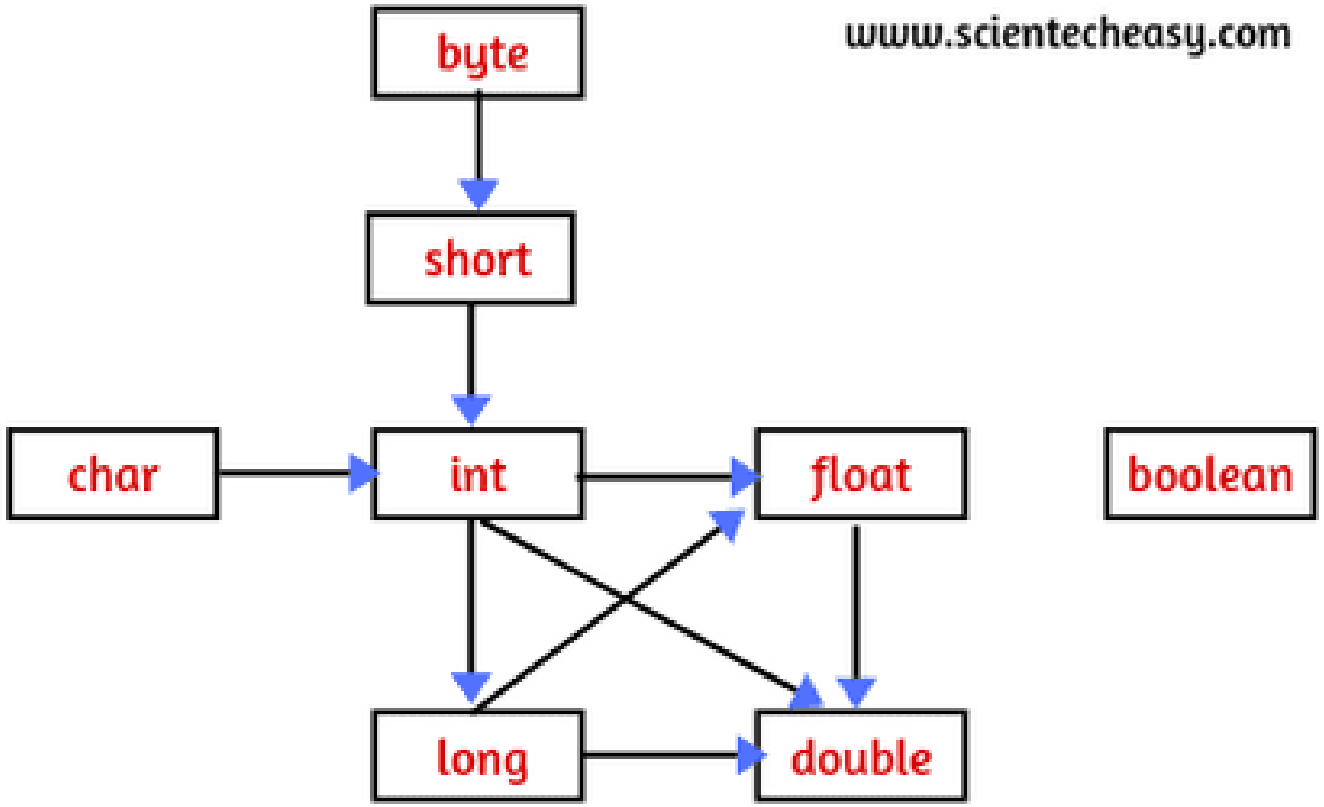


Fig: Automatic type conversion that Java allows.

Operators in Java

Operator	Symbol	Description	Example

Addition Operator	+	Adds the left operand with the right operand and returns the result.	int a=10;
			int b=5;
			int c = a+b; //15

Subtraction Operator	–	Subtracts the left operand and right operand and returns the result.	int a=10;
			int b=5;
			int c = a-b; //5

Multiplication Operator	*	It multiples the left and right operand and returns the result.	int a=2;
			int b=3;
			int c=a*b; //6

Division Operator	/	Divides the left operand with the right operand and returns the result.	int a=10;
			int b=5;
			int c=a/b; //0

Modulo Operator	%	Divides the left operand with the right operand and returns the remainder.	int a=10;
			int b=5;
			int c=a%b; //2

Symbol	Description	Example

=	Assign the right operand to the left operand.	1.int a=10;
		2.int b=20;
		3.a+=4; //a=a+4 (a=10+4)
		4.b-=4; //b=b-4 (b=20-4)

+=	Adding left operand with right operand and then assigning it to variable on the left.	int a=5; a += 5; //a=a+5;
-=	subtracting left operand with right operand and then assigning it to variable on the left.	int a=5; a -= 5; //a=a-5;
*=	multiplying left operand with right operand and then assigning it to the variable on the left.	int a=5; a *= 5; //a=a*5;
/=	dividing left operand with right operand and then assigning it to variable on the left.	int a=5; a /= 5; //a=a/5;
%=	assigning modulo of left operand with right operand and then assigning it to the variable on the left.	int a=5; a %= 5; //a=a%5;

Operator	Symbol	Description	Example
Equal to	==	returns true of the left-hand side is equal to the right-hand side.	5==3 is evaluated to false.
Not Equal to	!=	returns true of the left-hand side is not equal to the right-hand side.	5!=3 is evaluated to true.
Less than	<	returns true of the left-hand side is less than the right-hand side.	5<3 is evaluated to false
Less than or equal to	<=	returns true of the left-hand side is less than or equal to the right-hand side.	5<=5 is evaluated to true
Greater than	>	returns true of the left-hand side is greater than the right-hand side.	5>3 is evaluated to true
Greater than or Equal to	>=	returns true of the left-hand side is greater than or equal to the right-hand side.	5>=5 is evaluated to true

Instance of Operator	instanceOf	It compares an object to a specified type.	String test = “asdf”;
			boolean result;
			result = test instanceOf String;
			//true

Operator	Symbol	Description	Example
Logical OR	||	It returns true if either of the Boolean Expression is true.	false || true is evaluated to true
Logical AND	&&	It returns true if all the Boolean Expressions are true	false && true is evaluated to false.

Operator	Symbol	Description	Example
Bitwise OR	|	It compares corresponding bits of two operands. If either of the bits is 1, it gives 1. If not, it gives 0.	int a=12, b=25; int result = a|b; //29
Bitwise AND	&	It compares corresponding bits of two operands. If either of the bits is 1, it gives 1. If either of the bits is not 1, it gives 0.	int a=12, b=25; int result = a&b; //8
Bitwise XOR	^	It compares corresponding bits of two operands. If corresponding bits are different, it gives 1. If corresponding bits are the same, it gives 0.	int a=12, b=25; int result = a^b; //21
Bitwise Complement	~	It inverts the bit pattern. It makes every 1 to 0 and every 0 to 1.	int a=35; int result = ~a; //-36

Operator	Symbol	Description	Example
Unary minus	–	It is used for negating the values.	int a=5.2; int b = -a; //-5.2
Unary plus	+	It is used for giving positive values.	int a=5.2; int b = +a; //5.2
Increment Operator	++	It is used for incrementing the value by 1.	int a=5.2; int b = ++a; //6.2
Decrement Operator	—	It is used for decrementing the value by 1.	int a=5.2; int b = –a; //4.2
Logical NOT Operator	!	It is used for inverting a Boolean value	boolean a=false; boolean b=!a; //true

Operator	Symbol	Description	Example
Left Shift Operator	<<	It is used to shift all of the bits in value to the left side of a specified number of times.	10<<2 //10*2^2=10*4=40
Right Shift Operator	>>	It is used to move left operands value to right by the number of bits specified by the right operand.	10>>2 //10/2^2=10/4=2

Ternary Operator in Java:

Example:

```
int a=2;
int b=5;
int c = (a<b)?a:b;
```

Output: 2

Level	Operator	Description	Associativity
16	[] . ()	access array element access object member parentheses	left to right
15	++ —	unary post-increment unary post-decrement	not associative
14	++ — + – ! ~	unary pre-increment unary pre-decrement unary plus unary minus unary logical NOT unary bitwise NOT	right to left
13	() new	cast object creation	right to left
12	* / %	multiplicative	left to right
11	+ – +	additive string concatenation	left to right
10	<< >> >>>	shift	left to right
9	< <= > >= instanceof	relational	not associative
8	== !=	equality	left to right
7	&	bitwise AND	left to right
6	^	bitwise XOR	left to right
5	|	bitwise OR	left to right
4	&&	logical AND	left to right
3	||	logical OR	left to right
2	?:	ternary	right to left
1	= += -= *= /= %= &= ^= |= <<= >>= >>>=	assignment	right to left

Test case:

```
int a = 10;
        int b=20;
        int c;
        System.out.println(c = a);
        System.out.println(b += a);
        System.out.println(b -= a);
        System.out.println(b *= a);
        System.out.println(b /= a);
        System.out.println(b %= a);
        System.out.println(b ^= a);
```

```java
int a = 10;
        boolean b=true;
        System.out.println(a++);
        System.out.println(++a);
        System.out.println(a--);
        System.out.println(--a);
        System.out.println(!b);
```

```java
package Demo;

public class TernaryOperators {

 public static void main(String[] args) {
        int a = 20, b = 10, c = 30, res;
        res = ((a > b) ? (a > c)? a: c: (b > c)? b: c);
        System.out.println("Max of three numbers = "+ res);

 }
}
```

Control Flow Statements in Java

1. **Decision-making statements:** if-then, if-then-else, switch
2. **Looping statements:** for, while and do-while
3. **Branching statements:** break, continue, return

In Java, the decision-making statements are as follows:

1. if Statement
2. if-else statement
3. if-else if .. statement
4. Case statement

Syntax:

```
if (expression)
{
    statement;
}
```

Syntax:

```
if (expression)
{
    statement-1;
}
else{
    statement-2;
}
```

Syntax:

```
if (expression)
{
    statement-1;
}
else if{
    statement-2;
}
else {
    statement-3;
}
```

Syntax:

```
switch(expression)
{
case value1:
Statement-1;
break;

case value2:
Statement-2;
break;

.
.
.

Case valueN:
Statement-N;
break;
}
```

Types of looping statements in Java:

There are three types of looping statements in Java. They are as follows:

1. **For loop**
2. **While loop**
3. **Do while loop**

```
for(initialization; condition; increment/decrement)
{
    Statement(s);
}
```

```java
for(int num = 1; num <= 5; num++)
        {
          System.out.println(num);
        }
```

```java
String array[] = {"Ron", "Harry", "Hermoine"};

//enhanced for loop
for (String x:array)
{
    System.out.println(x);
}
```

```
while(condition)
{
     Statement(s);
}
```

```java
int x = 1;

// Exit when x becomes greater than 4
while (x <= 4)
{
    System.out.println("Value of x:" + x);

    // Increment the value of x for
    // next iteration
    x++;
}
```

```
do{
     Statement(s);
}
while(condition);
```

```java
int x = 21;
do
{
    // The line will be printed even
    // if the condition is false
    System.out.println("Value of x:" + x);
    x++;
}
while (x < 20);
```

Types of Branching Statement in Java:

In Java, there are three Branching Statements. They are as follows:

1. **Break Statement**
2. **Continue Statement**
3. **Return Statement**

```java
// Initially loop is set to run from 0-9
for (int i = 0; i < 10; i++)
{
    // terminate loop when i is 5.
    if (i == 5)
        break;

    System.out.println("i: " + i);
}
System.out.println("Loop complete.");
```

```java
for (int i = 0; i < 10; i++)
{
    // If the number is even
    // skip and continue
    if (i%2 == 0)
        continue;

    // If number is odd, print it
    System.out.print(i + " ");
}
```

```java
public static void main(String args[])
{
    boolean t = true;
    System.out.println("Before the return.");

    if (t)
        return;

    // Compiler will bypass every statement
    // after return
    System.out.println("This won't execute.");
}
```

First Java Program

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hi everybody, Welcome to Java");
}
}
```

Variable: Khai báo, sử dụng biến và hằng số

```
import java.time.Year;
public class vidu_khaibaobien {
public static void main(String[] args) {
String hovaten="Do Ha Phuong";
final int namsinh=1990;
int tuoi=Year.now().getValue()-namsinh;
System.out.println("Ho va ten: "+hovaten);
System.out.println("Tuoi: "+tuoi);
}
}
```

Nhập và xuất trong Java

```
import java.util.Scanner;

public class vidu_nhapvaxuat {
public static void main(String[] args) {
Scanner nhap = new Scanner(System.in);
System.out.println("Nhap vao ho va ten: ");
String hoten=nhap.nextLine();
System.out.println("Nhap vao nam sinh: ");
int namsinh=nhap.nextInt();
System.out.println("Xin chao: "+hoten+" Sinh nam:"+namsinh);
}
}
```

```
import java.io.IOException;
import java.util.Scanner;
public class Vidu_Nhap_va_Xuat {
public static void main(String[] args) throws
IOException {
Scanner input = new Scanner(System.in);
   int i;   short s;   byte b;   long l;   float f; double d;
   boolean bo;   String str;   char c;
   System.out.print("Nhập 1 số nguyên: ");
   i = input.nextInt();
   System.out.println("Sau khi nhập, i = " + i);
   System.out.print("Nhập 1 số nguyên short: ");
   s = input.nextShort();
   System.out.println("Sau khi nhập, s = " + s);
   System.out.print("Nhập 1 số nguyên byte: ");
   b = input.nextByte();
   System.out.println("Sau khi nhập, b = " + b);
   System.out.print("Nhập 1 số nguyên long: ");
   l = input.nextLong();
System.out.println("Sau khi nhập, l = " + l);
   System.out.println("Nhập 1 số thực float: ");
   f = input.nextFloat();
   System.out.println("Sau khi nhập, f = " + f);
   System.out.print("Nhập 1 số thực double: ");
   d = input.nextDouble();
   System.out.println("Sau khi nhập, d = " + d);
   System.out.print("Nhập 1 giá trị boolean: ");
   bo = input.nextBoolean();
   System.out.println("Sau khi nhập, bo = " + bo);
   System.out.print("Nhập 1 chuỗi: ");
   input.nextLine();
   str = input.nextLine();
   System.out.println("Sau khi nhập, str = " + str);
   System.out.print("Nhập 1 ký tự: ");
   c = (char) System.in.read();
   System.out.println("Sau khi nhập, c = " + c);
  }
}
```

Tài liệu tham khảo

- https://docs.oracle.com/javase/tutorial/
- https://www.javatpoint.com/java-tutorial
- https://techvidvan.com/tutorials/java-class/
- https://dotnettutorials.net/lesson/variables-in-java/