FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

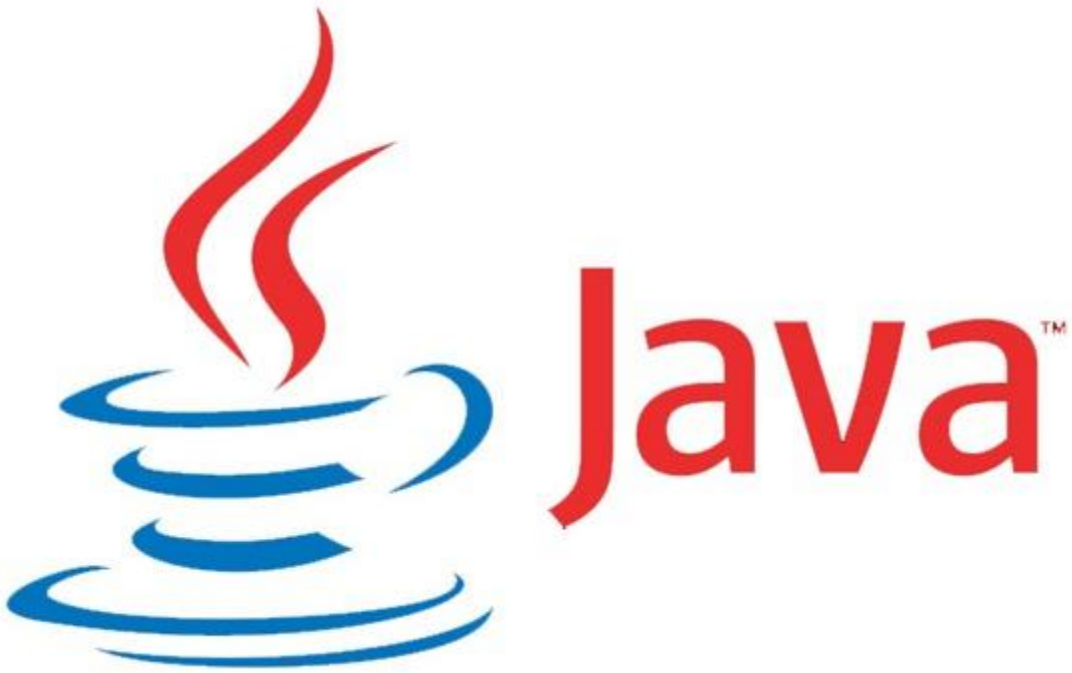
Session 02
Learning the Java Language

cantho.fpt.edu.vn

Objectives

- Information about structure, variables, input, output, primitive data types, type conversions and explicit casting, conditional operator, operators.
- Structure of select constructs, loop constructs.
- How to define and use array in Java.

Java language programming



Java program structure

```
package com.fuct;
public class Program{
    public static void main(String[] args){
        // executable code
    }
}
```

Save as → Program.java

- com.fuct: package contains class
  - Use lowercase and dots. Package is folder, class is file.
- Program: class name.
  - Must be the same as java file name. Capitalize the first letter of each word.
- main(): The method starts running
  - Class can have many methods but only main() called automatically when the application is run.

Java vs C

```
S1:  // HelloWorld.java: Hello World program
S2:  import java.lang.*;
S3:  class HelloWorld
     {
S4:      public static void main(String args[])
         {
S6:          System.out.println("Hello World");
         }
     }
```

```
/* helloworld.c: Hello World program */
#define <stdio.h>

void main(int argc, char *argv[])
{
   printf("Hello World\n");
}
```

Variables

```
1  public class MyClass{
2      public static void main(String[] args){
3          int a = 5;
4          int b = 7;
5          int c = a + b;
6          System.out.println("Sum:  " + c);
7      }
8  }
```

- Assign a value of 5 to a, 7 to b and sum a + b to c, then output to the screen.
- a, b and c are integer variables.
- Each variable has its own data type.

Declare variable

- Syntax:

```
<Data Type> <Variable name> [= <Value>];
```

- Ex:

```
int a; //Declare variable has not initial value
double b = 5;//Declare variable has initial value
```

- Declare variables of the same type

```
int a, b = 7, c;
```

- Assign a value to the variable

```
c = 8;
a = 10;
```

Variable naming conventions

Variable naming conventions:

- Must start with either an alphabetic character, an underscore (_), or a dollar sign ($), and may contain any of these characters plus numeric digits.
- No other characters are allowed in variable names.
- The name has UPPERCASE/lowercase distinction.
- Not ~~start by number~~, not use ~~keywords~~.

Keywords of Java

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				

Example: valid variables?

A
- abc

B
- 1abc — False

D
- ab_c

E
- _abc

F
- $abc

G
- ab-c — False

H
- _123

I
- $123

J
- if — False

K
- $if

Overview of the Java Operators (1)

- Operators in Java, in Descending Order of Precedence

Category	Operators
Unary	++ -- + - ! ~ (type)
Arithmetic	* / %
	+ -
Shift	<< >> >>>
Comparison	< <= > >= instanceof
	== !=

Shifts bits of op1 right by distance op2; fills with 0 bits on the left side

Overview of the Java Operators (2)

- Operators in Java, in Descending Order of Precedence …

Bitwise	& ^ |
Short-circuit	&& ||
Conditional	?:
Assignment	= *op=*

```
if (x) {
   a = b;
 }
 else {
   a = c;
}
```

Replace with:

a=x?b:c;

Conditional operator

- Syntax:

```
<condition> ? <value 1> : <value 2>
```

- Interpretation:
  - If `<condition>` is true value then result of expression is `<value 1>`, else is `<value 2>`.
- Example: Maximum of 2 number a and b

```
int a = 1, b = 9;
int max = a > b ? a : b;
```

Example

- The value of the following expressions?

```
a) 5 + 3 * 2 != 11
b) 5 + 3 * 2 == 11
c) 5 <= 3 * 2
d) 5 > 3 * 2
e) 5 + 2 >= 3
f) true && false
g) 5 < 2 && true
h) 5 < 3 * 2 || 11 != 11
i) 5 > 3 * 2 || 11 >= 11
j) !(5 > 3 * 2) && 11 >= 11
```

Descending Order of Precedence

- The value of the expression?

```
a. 5 + 2 * 3
b. 5 + 2 % 3
c. 5 % 2 * 3
d. 3*2 > 2 + 7
e. ++2 * 3
```

Input

- java.util.Scanner allow input data from keyboard
- Create Scanner object
  - `Scanner scanner = new Scanner(System.in)`
- Methods:
  - `scanner.nextLine()`
    - Input a line from keyboard.
  - `scanner.nextInt()`
    - Input a integer from keyboard
  - `scanner.nextDouble()`
    - Input a decimal from keyboard

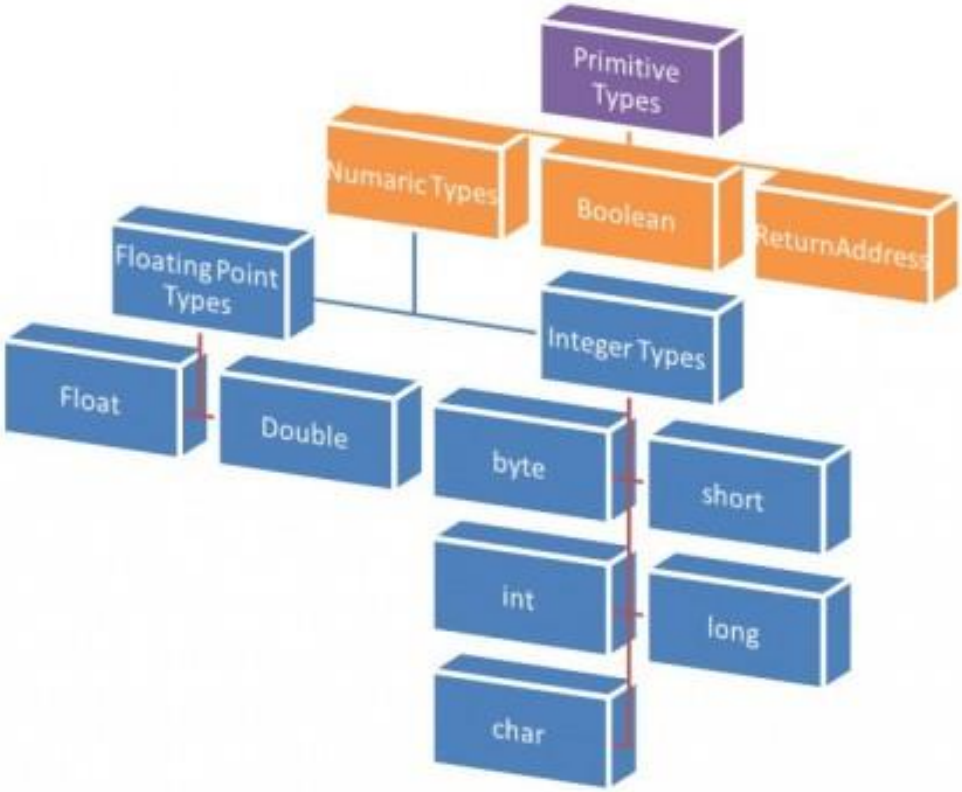
Output

- `System.out.print()`: Output on multiple lines.
- `System.out.println()`: Output on single line.
- `System.out.printf()`: Output has format, character formatting:
  - `%d`: Integer
  - `%f`: Decimal
    - Defaut has 6 decimal
    - `%.3f` Format 3 decimal
  - %s: String
- Example:

```
System.out.print("FPT");
System.out.println(" University");
System.out.printf(" Have %d campus", 4);
```

Primitive Data Types (1)

- A *primitive* is a simple non-object data type that represents a single value. Java's primitive data types are:
  - boolean
  - char
  - byte
  - short
  - int
  - long
  - float
  - double



Primitive Data Types (2)

Type	Defaut	Bit	Minimum	Maximum
Byte	0	8	-128	+127
Short	0	16	-32,768	+32,767
Int	0	32	-2^{31}	$+2^{31}-1$
Long	0L	64	-2^{63}	$+2^{63}-1$
Float	0.0F	32	$-3.40292347 \times 10^{38}$	$+3.40292347 \times 10^{38}$
Double	0.0	64	$-1.79769313486231570 \times 10^{308}$	$+1.79769313486231570 \times 10^{308}$
Boolean	False	1	False	True
Char	'\u0000'	16	'\u0000'	'\uffff'

Type Conversions and Explicit Casting

Automatic type conversion:

```
int x;
double y;
y = 2.7;
x = y;
```

narrowing conversion

Explicit cast

Syntax: **`x = (type) expression;`**

Example: `int x = (int)2.7;`

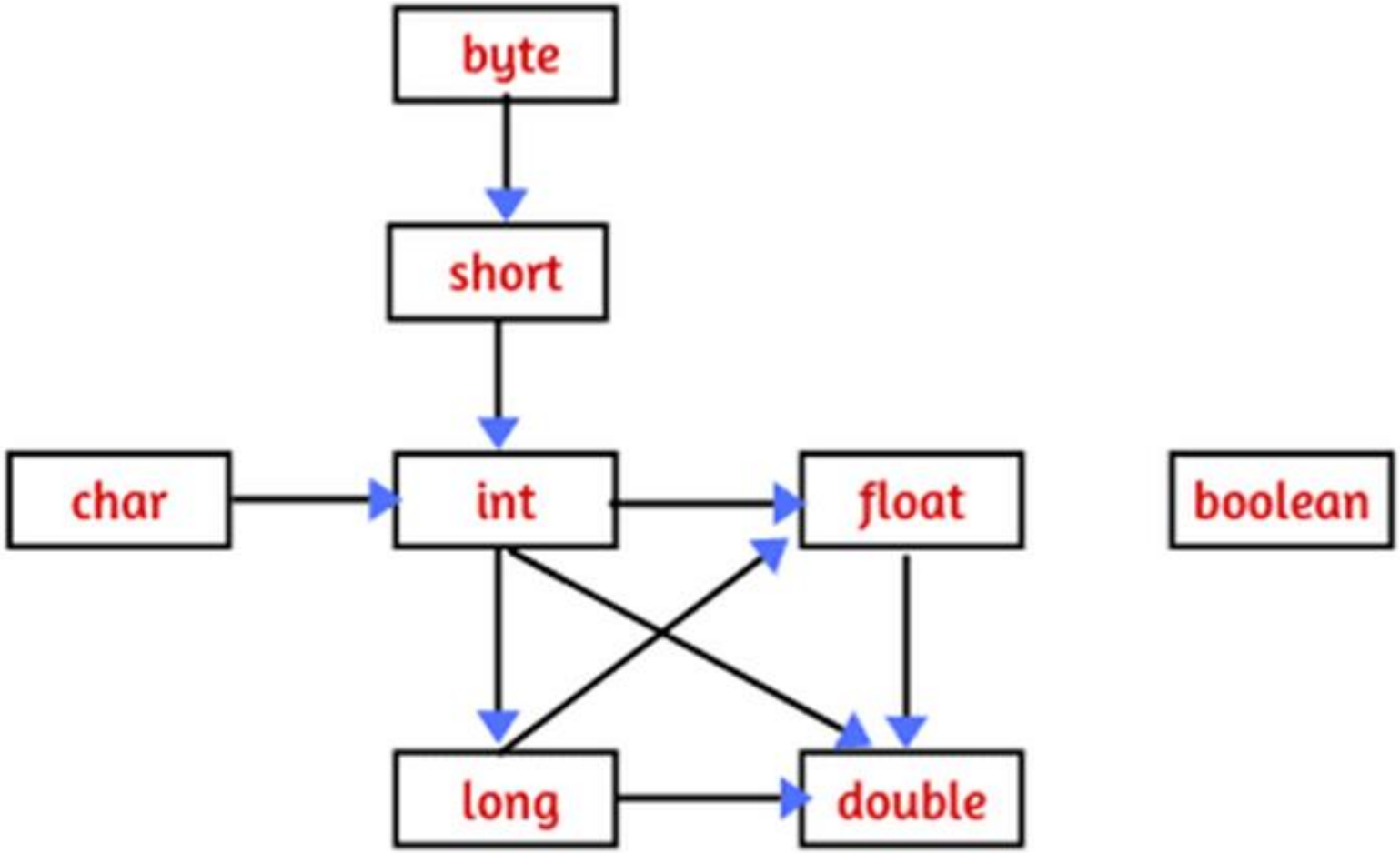


Fig: Automatic type conversion that Java allows.

Converts string to primitive type

- Expression 1

```
String a = "3";
String b = "4";
String c = a + b;
```

=> c = ?

String => Primitive type
byte Byte.parseByte(String)
short Short.parseShort(String)
int Integer.parseInt(String)
long Long.parseLong(String)
float Float.parseFloat(String)
double Double.parseDouble(String)
boolean Boolean.parseBoolean(String)

- Expression 2

```
int a = Integer.parseInt("3");
int b = Integer.parseInt("4");
int c = a + b;
```

=> c = ?

Demo

- Write a program to enter the unit price and quantity from the keyboard using the scanner's nextLine(), then convert it to integer and real numbers to calculate the money.

Practices

- // - permittion user input name and year of birth,
- // - calculate user age, print user name and user age,

Slot 4 cont…

Select Constructs

Select 1/2	Select 1/n
if if…else if…else if…else :? operator	switch…case

Syntax: If... else

```
if (<<conditional>>) {
    << Statement 1 >>
}
else {
    << Statement 2 >>
}
```

```
if (<<conditional 1>>){
    << Statement 1 >>
}
else if (<<conditional 2>>){
    << Statement 2 >>
}
…
else {
    << Statement N+1 >>
}
```

Practices

- // - permittion user input name and year of birth,
- // - calculate user age, print user name and user age,
- // - if user age >=18 print note: (adult)

Practices

- If-Else:
  - mark -> degree classification
    - Input: mark from 0.0 – 10.0
    - Output: text
    - Example: input: 9.0 output: Very good
      input: 11 output: invalid
  - mark -> degree classification
    - Input: mark A,B,C,D,F
    - Output: text
    - Example: input: A output: Very good
      input: S output: invalid

Practices

Solve equation: ax + b = 0

- Input: a and b
- Output: x = ?
    - if a=0 b<>0: no x
    - if a=0 b=0: too many x
    - x=-b/a

Switch ... case

```
switch (variable or expression) {
    case constant_1:
        statement(s);
        break;
    case constant_2:
        statement(s);
        break;
    ...
    case constant_n:
        statement(s);
        break;
    default:
        statement(s);
}
```

Practices

- Switch - case:
  - mark -> degree classification
    - Input: mark A,B,C,D,F
    - Output: text
    - Example: input: A output: Very good
      input: S output: invalid

Demo

- Write a program that allows user inputting a simple expression containing one of four operators +, -, *, / then the result is printed out to the monitor.
- Input format: num1 operator num2

```java
/* Program: Calculator (+, -, *, /) of 2 number a and b
Input: float a, float b and operator (+, -, *, /)
Output: Result of the operation */
package democ2;
import java.util.Scanner;
public class DemoC2 {
    public static void main(String[] args) {
        // TODO code application logic here
        float a, b;
        char op = '+';
        Scanner ip = new Scanner(System.in);
        System.out.println("Program: Calculator (+, -, *, /) of 2 number a and b");
        System.out.println("---Input a: ");
        a = ip.nextFloat();
        System.out.println("---Input b: ");
        b = ip.nextFloat();
        System.out.println("---Operator: ");
        op = ip.next().charAt(0);//Input 1 character
        switch (op) {
            case '+': System.out.println(a + " + " + b + " = " + (a + b)); break;
            case '-': System.out.println(a + " - " + b + " = " + (a - b)); break;
            case '*': System.out.println(a + " * " + b + " = " + (a * b)); break;
            case '/': if (b == 0) { System.out.println("b # 0.");
                } else { System.out.println(a + " / " + b + " = " + (float) a / b);} break;
            default: System.out.println("Error!!! Operator is not valid.");
        }
    }
```

```
run:
Program: Calculator (+, -, *, /) of 2 number a and b
---Input a:
15
---Input b:
3
---Operator:
/
15.0 / 3.0 = 5.0
BUILD SUCCESSFUL (total time: 35 seconds)
```

The Loop Constructs

- Java provides three loop constructions. Taken from C and C++, these are:
  - while()
  - do … while()
  - for()

Example

- Print Even numbers from 1 to n or 1 to 100

```
// FOR
int n = 100;
System.out.print("\n Even Numbers from 0 to "+n+" are: ");
for (int i = 0; i <= n; i=i+2) {
        System.out.print(i + " ");      }

// WHILE
int i = 0; n = 100;
System.out.print("\n Even Numbers from 0 to "+n+" are: ");
while (i<=n) {
        System.out.print(i + " ");
        i=i+2;  }

// DO..WHILE
i = 0; n = 100;
System.out.print("\n Even Numbers from 0 to "+n+" are: ");
do {
        System.out.print(i + " ");
        i=i+2;
} while (i<=n);
```

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)
```

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

Java Naming Conventions (1)

- ***Classes and Interfaces***
  - Class names should be nouns, in mixed case with the first letter of each internal word capitalized.
  - Interfaces name should also be capitalized just like class names.
  - Ex:

```
Interface  Bicycle
Class MountainBike implements Bicyle
```

- ***Methods***
  - Methods should be verbs, in mixed case with the first letter lowercase and with the first letter of each internal word capitalized.
  - Ex:

```
void changeGear(int newValue);
void speedUp(int increment);
```

Java Naming Conventions (2)

- ***Constant variables***
  - Should be all uppercase with words separated by underscores ("_").
  - There are various constants used in predefined classes like Float, Long, String etc.
  - Ex:

```
static final int MIN_WIDTH = 4;
public static final float
POSITIVE_INFINITY = 1.0f / 0.0f;
```

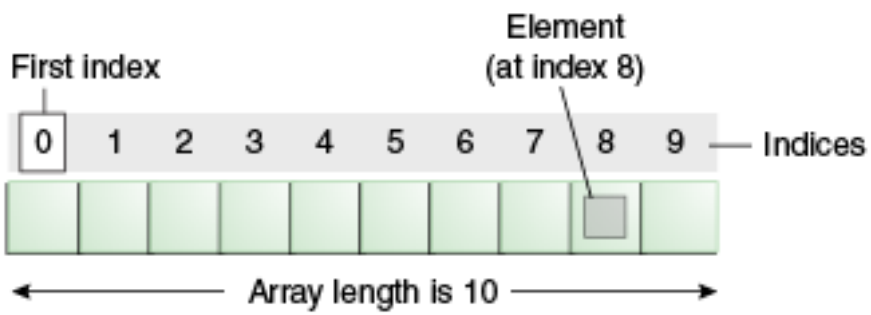
Java Naming Conventions (2)

- ***Packages***
  - The prefix of a unique package name is always written in **all-lowercase ASCII letters** and should be one of the top-level domain names, like com, edu, gov, mil, net, org.
  - Subsequent components of the package name vary according to an organization's own internal naming conventions.
  - **Ex:**

```
com.sun.eng
com.apple.quicktime.v2
// java.lang packet in JDK
java.lang
```

Arrays (1)

- An *array* is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- Each item in an array is called an *element*, and each element is accessed by its numerical index.



	0	1	2	3	
	a[0][0]	a[0][1]	a[0][2]	a[0][3]	0
a	a[1][0]	a[1][1]	a[1][2]	a[0][3]	1
	a[2][0]	a[2][1]	a[2][2]	a[2][3]	2

Arrays(2)

- Declaring a Variable to Refer to an Array
  - `datatype[] arr;`
  - `datatype arr[];`
- Creating, Initializing, and Accessing an Array
  - `arr = new datatype[size];`
  - `datatype[] arr = new datatype[size];`
  - `datatype[] arr = {elem1, elem2,…};`
- Example:
  - `int[] anArray;`
  - `float anArrayOfFloats[];`
  - `anArray = new int[10];`

Arrays (2)

- Copying Arrays
  - Use arraycopy method from System class.
- Use index to distinguish elements. Index of arrays from 0.
  - After `a[2] = a[1]*5`, array a is {4, 3, 12, 7}
- Use length attribute to get number of element of array.

```
int a[] = {3, 4, 5, 7};
a[2] = a[1]*4; //3*4=12
System.out.println(a.length);
```

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.

Demo

- Input array of integer from keyboard and output: print this arrays on screen.

```java
import java.util.Scanner;

/*Program: Input array of integer from keyboard and output: print this arrays on screen.
Input: Size of array, element of array
Output: Print list of array
 */

public class DemoArray {
    public static void main(String[] args) {
        Scanner ip = new Scanner(System.in);
        System.out.print("Input number of elements: ");
        int size = ip.nextInt();
        int a[] = new int[size];
        System.out.println("Input elements: ");
        for(int i = 0; i<size;i++)
        {
            System.out.print("---A ["+i+"]: ");
            a[i]=ip.nextInt();
        }
        System.out.print("Array is: ");
        for(int i = 0; i<size;i++)
        {
            System.out.print(a[i] + " ");
        }
    }
}
```

Output - DemoC2 (run)

```
run:
Input number of elements: 4
Input elements:
---A [0]: 1
---A [1]: 2
---A [2]: 33
---A [3]: 5
Array is: 1 2 33 5 BUILD SUCCESSFUL (total time: 7 seconds)
```

Elements of Java Style

- Proper Use of Indentation
  - Statements within a block of code should be indented relative to the starting/ending line of the enclosing block.
- Use Comments Wisely
- Placement of Braces
  - Opening brace at the end of the line of code that starts a given block. Each closing brace goes on its own line, aligned with the first character of the line con.
- Descriptive Variable Names

Summary

- Information about structure, variables, input, output, primitive data types, type conversions and explicit casting, conditional operator, operators.
- Structure of select constructs, loop constructs.
- The core concepts behind object-oriented programming: objects, interfaces, classes, and inheritance.

ĐẠI HỌC FPT CẦN THƠ