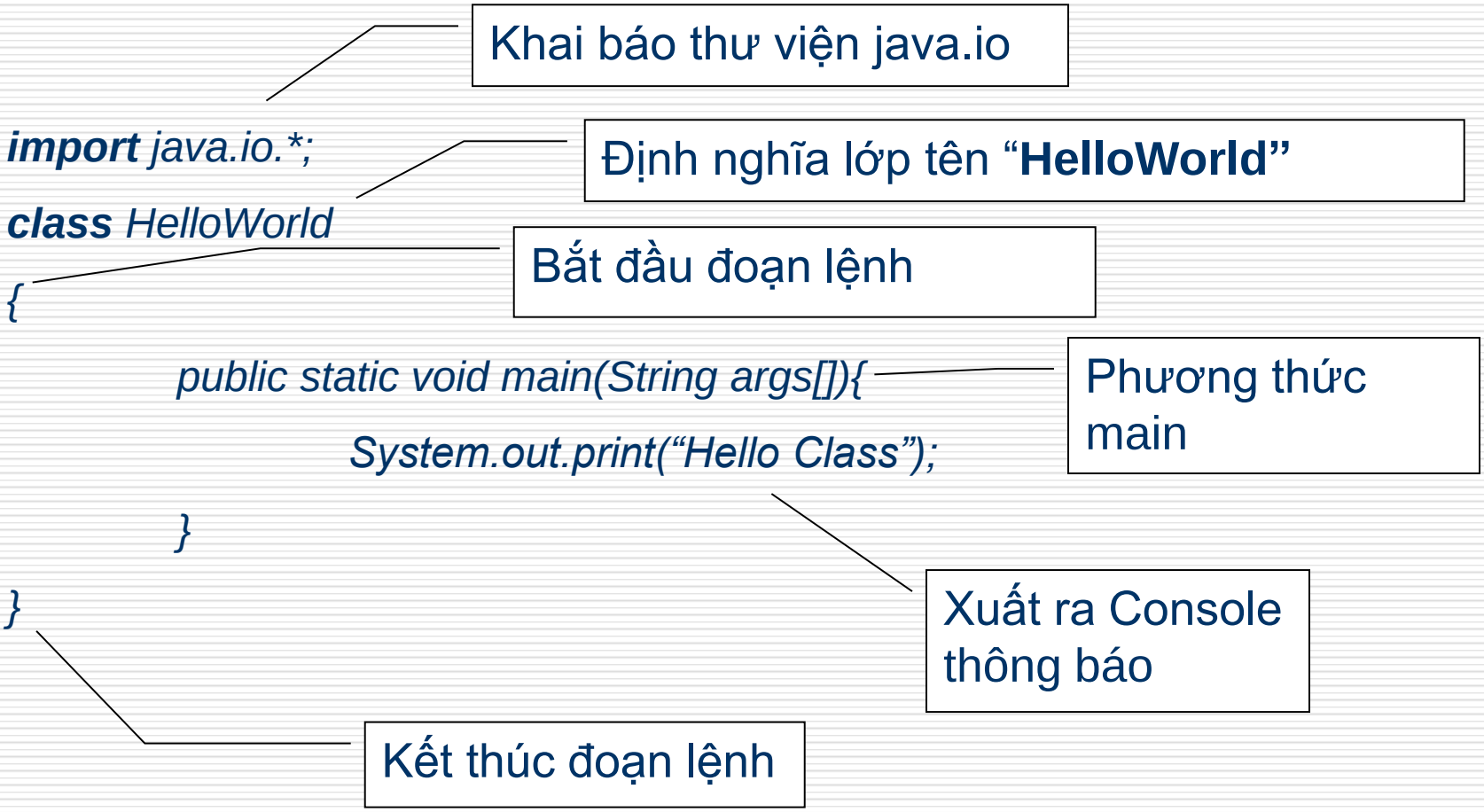
VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
**UNIVERSITY OF INFROMATION TECHNOLOGY**


## Chapter 2

# JAVA CORE

Lecturer: MSc. Kiet Van Nguyen
Faculty of Information Science and Engineering
University of Information Technology, VNU-HCM

# Testing before a new lesson



# Recall

1. In the Java programming language, all source code is first written in plain text files ending with the _____ extension.

A.Javac

B.java

C.class

D.txt

# Recall

2. The Java compiler generates

A. machine code

B. source code

C. byte code

B. HTML

# Recall

3. JVM stands for

A. Java Variable Machine

B. Java Variable Method

C. Java Virtual Method

D. Java Virtual Machine

# Recall

4. Write the statement to

a) compile HelloWorldApp application

b) Run the program

# Recall

5. Write the signature of the main method?

# Today's Objectives

- Identifiers
- Variable & Constant
- Data Types
- Operators & expressions
- Control Structures
- Wrapper Class
- Examples

# Identifiers

- An identifier is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($).
- An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word. (“Java Keywords,” for a list of reserved words).
- An identifier cannot be true, false, or null.
- An identifier can be of any length.

# Exercise: Choose correct identifiers? Why incorrect?

MyVariable, My Variable, myvariable, x, i, my_Variable, _myvariable, a+c, $myvariable, sum_&_difference, sum_of_array, MYVARIABLE, dataflair123, 123gkk, variable-2, O'Reilly

# Variable

- Variables are containers for storing data values
- To create a variable, you must specify the type and assign it a value:
  - ✓ Case 1:

    *<data type> <variable_name>;*

    *<variable_name> = <value>;*
  - ✓ *Case 2:*

    *<data type> <variable_name> = <value>;*

# Variables (Cont.)

- Two different types of variables: member variable and local variable.
- Member variable: DON'T NEED to initialize variable value (assigned default value).
- Local variable: MUST to initialize variable value.

# Variables (Cont.)

```
class Car {
        String licensePlate = ""; // member variable
        double speed; = 0.0; // member variable
        double maxSpeed; = 123.45; // member variable
        boolean isSpeeding() {
                double excess; // local variable
                excess = this.maxSpeed - this.speed;
                if (excess < 0)
                        return true;
                else
                        return false;
                }
        }
```

# Exercise

This is a named storage location in the computer's memory.

A. class

B. keyword

C. variable

D. operator

# Constant

- A constant is a variable whose value cannot change once it has been assigned.

  **final datatype CONSTANTNAME = VALUE;**

- For example,

**final** int x = 10; // declare a integer constant x = 10

**final** long y = 20L; // declare a long constant y = 20

# Exercise

This keyword is used to declare a named constant.

A. constant

B. namedConstant

C. final

D. concrete

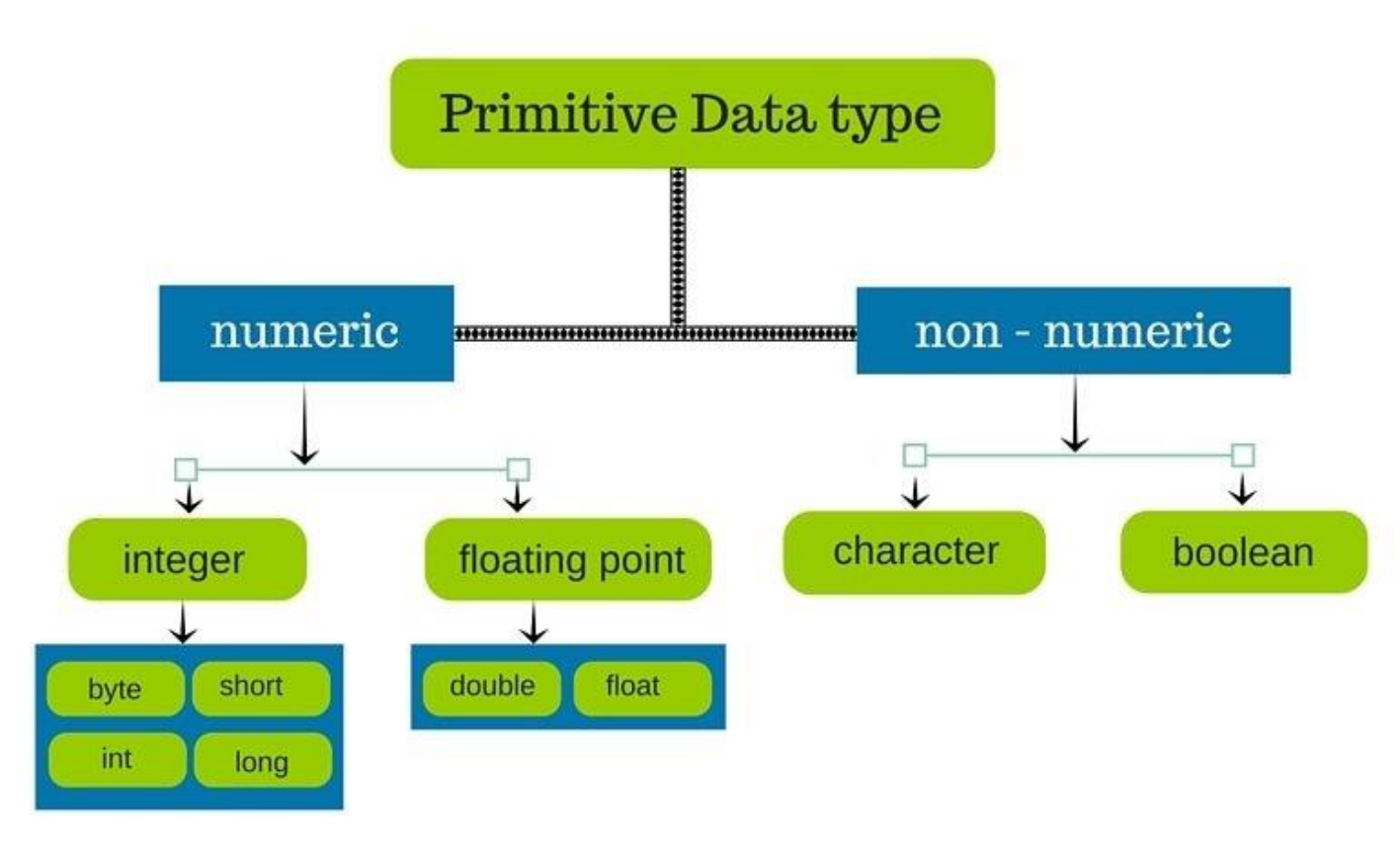
# Special Constants

| Escape Sequence | Description |
|---|---|
| \t | Insert a tab in the text at this point. |
| \b | Insert a backspace in the text at this point. |
| \n | Insert a newline in the text at this point. |
| \r | Insert a carriage return in the text at this point. |
| \f | Insert a formfeed in the text at this point. |
| \' | Insert a single quote character in the text at this point. |
| \" | Insert a double quote character in the text at this point. |
| \\ | Insert a backslash character in the text at this point. |

# Data Type

- Primitive data type
- Reference data type

# Primitive Data Types



# Primitive Data Types (Cont.)

| Reserved Word | Data Type | Size | Range of Values |
|---|---|---|---|
| byte | Byte Length Integer | 1 bytes | $-2^8$ to $2^7-1$ |
| short | Short Integer | 2 bytes | $-2^{16}$ to $2^{16}-1$ |
| int | Integer | 4 bytes | $-2^{32}$ to $2^{31}-1$ |
| long | Long Integer | 8 bytes | $-2^{64}$ to $2^{63}-1$ |
| float | Single Precision | 4 bytes | $-2^{32}$ to $2^{31}-1$ |
| double | Real number with double | 8 bytes | $-2^{64}$ to $2^{62}-1$ |
| char | Character ( 16 bit unicode ) | 2 bytes | 0 to 216 - 1 |
| boolean | Has value true or false | A boolean value | true or false |

# Exercise

Which of the following is not primitive data type?

A. String

B. double

C. boolean

D. int

# Type Casting

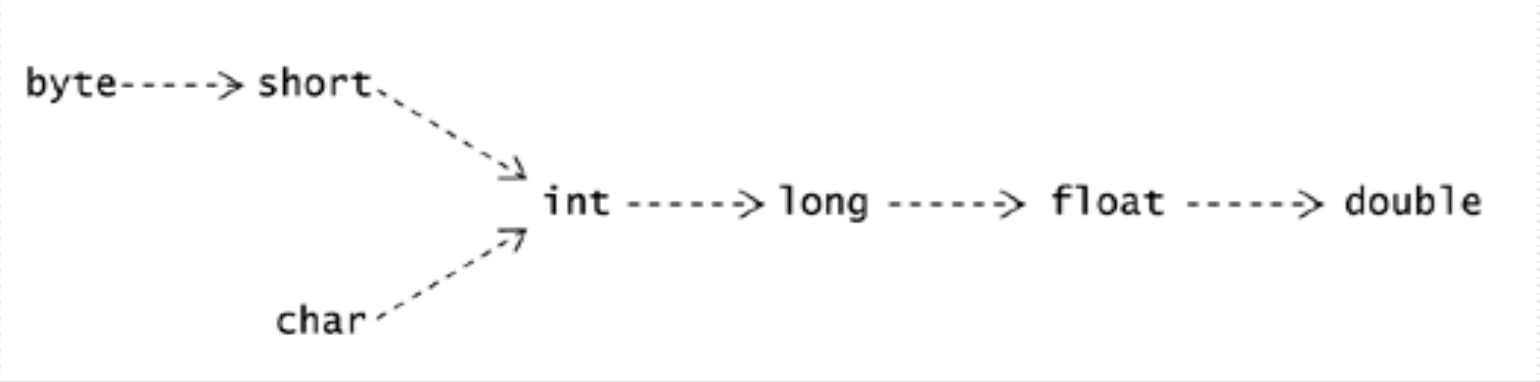
- Type narrowing: ***must cast***

  ***<tên biến 2> = (kiểu dữ liệu) <tên biến 1>***

  **int i = (int)3.0**

  - Type widening: ***don't need cast***

  **double d = 3;**



# Type Casting (Cont.)

- Cannot convert between boolean type and int and vice versa.
- Conversion rules:

1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.

# Type Casting (Cont.)

- **For example**

*1. byte x = 5;*

*2. byte y = 10;*

*3. byte z = x + y;*

*// third line?*

*// should: byte z = (byte) (x + y);*

# Reference data type

- **Declaring a reference variable**

  *<Object Type> <Object Variable>;*

- **Initialize an object**

  *<Object Type> <Object Variable>* = ***new*** *<Object Type>*;

- **Access object components**

  *<Object Variable>*.*<Properties>*

  *<Object Variable>*.*<Methods>*

# Reference data type (Cont.)

- **Array Type**
  - ✓ Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
  - ✓ You access an array element by referring to the index number.
  - ✓ To declare an array, define the variable type with **square brackets**:

*<DT>[] <Array Name>; // mảng 1 chiều*

*<DT> <Array Name>[]; // mảng 1 chiều*

*<DT>[][] <Array Name>; // mảng 2 chiều*

*<DT> <Array Name>[][]; // mảng 2 chiều*

# Reference data type (Cont.)

- **Initialization**

*int arrInt[] = {1, 2, 3};*

*char arrChar[] = {'a', 'b', 'c'};*

*String arrString[] = {"ABC", "EFG", "GHI"};*

- **Allocating & acceessing an array**

*int [] arrInt = **new** int[100];*

*int arrInt[100]; // Error.*

indexes of a n-element array: from ***0*** to ***n-1***

# Operators and expressions

## Arithmetic Operators

| Operator | Description | Example |
|---|---|---|
| + (Addition) | Adds values on either side of the operator. | A + B will give 30 |
| - (Subtraction) | Subtracts right-hand operand from left-hand operand. | A - B will give -10 |
| * (Multiplication) | Multiplies values on either side of the operator. | A * B will give 200 |
| / (Division) | Divides left-hand operand by right-hand operand. | B / A will give 2 |
| % (Modulus) | Divides left-hand operand by right-hand operand and returns remainder. | B % A will give 0 |
| ++ (Increment) | Increases the value of operand by 1. | B++ gives 21 |
| -- (Decrement) | Decreases the value of operand by 1. | B-- gives 19 |

**A = 10**
**B = 20**

# Operators and expressions (Cont.)

- **Relational Operators**

| Operator | Meaning |
|---|---|
| = = | ? |
| != | ? |
| > | ? |
| < | ? |
| >= | ? |
| <= | ? |
| \|\| | OR |
| && | AND |
| ! | NOT |

# Operators and expressions (Cont.)

- **Assignment Operators**

| Operator | Example | Meaning |
|---|---|---|
| = | a = b | assign a = b |
| += | a += 5 | a = a + 5 |
| -= | b -= 10 | b = b – 10 |
| *= | c *= 3 | c = c * 3 |
| /= | d /= 2 | d = d/2 |
| %= | e %= 4 | e = e % 4 |

# Exercise

Write Java statements that accomplish the following.

a. Declare int variables a and b.

b. Initialize an int variable x to 10 and a char variable ch to 'y’.

c. Declare and initialize a double variable payRate to 12.50.

d. Declare a boolean variable ans and set the value of ans to true.

# Exercise

Write the output of the following expressions.

a. System.out.println(13 / 4);

b. System.out.println(2 + 12 / 4);

c. System.out.println(21 % 5);

d. System.out.println(3 - 5 % 7);

e. System.out.println(17.0 / 4);

f. System.out.println(8 - 5 * 2.0);

g. System.out.println(14 + 5 % 2 - 3);

k. System.out.println(15.0 + 3 / 2);

# Exercise

What is the value of each variable after the last statement executes?

```
int a, b, c;
double x, y;
a = 17;
b = 15;
a = a + b / 4;
c = a % 3 + 4;
x = 17 / 3 + 6.5;
y = a / 4.0 + 15 % 4 - 3.5;
```

# Exercise

Suppose x, y, and sum are int variables and z is a double variable. What value is assigned to sum variable after statement executes? Suppose x = 3, y = 5, and z = 14.1.

```
sum = x + y + (int) z;
```

# Exercise

Write equivalent statements using combined assignment for the following, if possible.

```
x = 2 * x;
x = x + y - 2;
sum = sum + num;
y = y / (x + 5);
```

# Programming Exercise

**Hours, Minutes and Seconds**
Write a program that asks the user to enter seconds as integer. The program should compute and display the number of hours, number of minutes and number of seconds in that seconds. For example

**Input:** 4205 seconds.

**Output:**
Hours : 1
Minutes : 10
Seconds : 5

```java
import java.util.Scanner;

public class Seconds
{
    public static void main(String[] args)
    {
        int seconds; // To hold seconds.

        // Create a Scanner object to read input.
        Scanner console = new Scanner(System.in);

        // Get seconds from the user.
        System.out.print("Enter seconds ");
        seconds = console.nextInt();

        // Calculate hours in that seconds.
        int hours = seconds / 3600;

        // Calculate remaining minutes in that seconds.
        seconds = seconds % 3600;
        int minutes = seconds / 60;

        // Calculate remaining seconds in that seconds.
        seconds = seconds % 60;

        // Display result.
        System.out.println("Hours: " + hours);
        System.out.println("Minutes: " + minutes);
        System.out.println("Seconds: " + seconds);
    }
}
```

# Operators and expressions (Cont.)

- **Conditional Operator**

  ***Syntax***: *<Condition> ? <Expression 1> : < Expression 2>*

  ***For instance***:

```
int x = 10;
int y = 20;
int Z = (x<y) ? 30 : 40;
// Results: z = 30 because (x < y) is true.
```

# Operators and expressions (Cont.)

- **Conditional Operator**

  ***Syntax***: *<Condition> ? <Expression 1> : < Expression 2>*

  ***For instance***:

```
int x = 10;
int y = 20;
int Z = (x<y) ? 30 : 40;
// Results: z = 30 because (x < y) is true.
```

# Operators and expressions (Cont.)

```
int num;    // holds value of integer

// Create a Scanner object for keyboard input.
Scanner console = new Scanner(System.in);

// Get an integer.
System.out.print("Enter integer : ");
num = console.nextInt();

// Get absolute value of num
num = (num < 0) ? -num : num;

System.out.println("Absolute value is " + num);
```

Input: *5*
*Output:?*

*Input:-15*
*Output:?*

# Control Structures (Cont.)

- ***if … else***

  **Form 1:**

  *if (<conditions>) {*

  *<statements>;*

  *}*

  **Form 2:**

  *if (<conditions>) {*

  *<statements block 1>;*

  *}*

  *else {*

  *<statements block 2>;*

  *}*

# Control Structures (Cont.)

```
int number;

// Create a Scanner object to read input.
Scanner console = new Scanner(System.in);

// Get number from the user.
System.out.print("Enter an integer: ");
number = console.nextInt();

// Determine even or odd.
if (number % 2 == 0)
{
    System.out.println("number is even");
}
else
{
    System.out.println("number is odd");
}
```

# Control Structures (Cont.)

- ***switch … case***

```
switch (<expression>) {
    case <value_1>:
            <code block 1>;
            break;
    ….
    case <value_n>:
            <code block n>;
            break;
    default:
            <default code block>;
}
```

# Control Structures (Cont.)

Write a program that prompts the user to enter grade. Your program should display the corresponding meaning of grade as per the following table

| Grade | Meaning |
|---|---|
| A | Excellent |
| B | Good |
| C | Average |
| D | Deficient |
| F | Failing |

```
char grade; // To hold grade

// Create a Scanner object to read input.
Scanner console = new Scanner(System.in);

// Get grade from the user.
System.out.print("Enter grade: ");
grade = console.next().charAt(0);

// Determine and display grade
switch (grade)
{
case 'A':
    System.out.println("Excellent");
    break;
case 'B':
    System.out.println("Good");
    break;
case 'C':
    System.out.println("Average");
    break;
case 'D':
    System.out.println("Deficient");
    break;
case 'F':
    System.out.println("Failing");
    break;
default:
    System.out.println("Invalid input");
```

# Loop Controls

- **Form 1:**

```
while (<Conditions>) {
    <code block>;
}
```

- **Form 2:**

```
do {
    <code block>;
    } while (Conditions);
```

- **Form 3:**

```
for (statement 1;statement 2; statement 3) {
    <code block>;
    }
```

# Loop Controls

```
int count = 1;     // count is initiliazed

while (count <= 10)     // count is tested
{
   System.out.print(count + " ");
   count++;             // count is changed
}
```

# Loop Controls

```
int  value;      // to hold data entered by the user
int  sum = 0;    // initialize the sum
char choice;     // to hold 'y' or 'n'

// Create a Scanner object for keyboard input.
Scanner console = new Scanner(System.in);

do
{
   // Get the value from the user.
   System.out.print("Enter integer: ");
   value = console.nextInt();

   // add value to sum
   sum = sum + value;

   // Get the choice from the user to add more number
   System.out.print("Enter Y for yes or N for no: ");
   choice = console.next().charAt(0);
}
while ((choice == 'y') || (choice == 'Y'));
```

# Exercise

Write code to draw two pictures as follows.

```
*
***
*****
*******
*********
```

```
1
212
32123
4321234
543212345
```

# Break

- To jump out of a **loop or switch**

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

# Continue

- Continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

# Input from CONSOLE

- **Class java.util.scanner**

| | | | |
|---|---|---|---|
| public boolean | **nextBoolean**() Details | public int | **nextInt**() Details |
| public byte | **nextByte**() Details | public String | **nextLine**() Details |
| public byte | **nextByte**(int radix) Details | public long | **nextLong**() Details |
| public double | **nextDouble**() Details | | |
| public float | **nextFloat**() Details | | |

# Wrapper Class

| Data type | Wrapper Class (java.lang.*) | Note |
|---|---|---|
| boolean | Boolean | - package: maybe contain two or more classes.<br>- Besides Wrapper Class, java.lang package provides basic classes to support designs in Java: String, Math, … |
| byte | Byte | |
| short | Short | |
| char | Character | |
| int | Integer | |
| long | Long | |
| float | Float | |
| double | Double | |

# See more Java basic

https://www.w3schools.com/java/

# Q&A

# Thank you!