|  |  |  |
| --- | --- | --- |
| **fColour** | **Meaning** | **Example** |
| **COLOUR** | Words With Special Meaning | Java |
| **COLOUR** | Library Package & Class | import java.util.Scanner; |
| **COLOUR** | Key Words / Special Character or Symbol Used With the Key Words | int |
| **COLOUR** | Method | name.length(); |
| **COLOUR** | Things Should Be Filled | System.out.print(‘Print What?’); |
| **COLOUR** | Actual Variable / Actual Data / Actual Number / Actual String | System.out.print(“Hello World”); // Output will be ‘Hello World’. |

**Contents**

[1. Beginning 2](#_Toc194922407)

[2. Import & Package 4](#_Toc194922408)

[3. Identifier Naming 5](#_Toc194922409)

[4. Data Type & Variables 6](#_Toc194922410)

[5. Operation & Arithmetic 8](#_Toc194922411)

[6. Casting & Conversion 9](#_Toc194922412)

[7. Control Flow (Selection & Loop) 10](#_Toc194922413)

[8. Library Class (java.lang.String) 11](#_Toc194922414)

[9. Library Class (java.util.Scanner) 12](#_Toc194922415)

[10. Library Class (javax.swing.JOptionPane) 13](#_Toc194922416)

[11. Others 14](#_Toc194922417)

# Beginning

1. **Main Structure :**

|  |
| --- |
| Class Header  Access Modifier  Class Name    Method Body  (Method Block)  Class Body  (Class Block)  Return Type  Main Method  (Method Name)  Statement  Parameter Name  Parameter |

* If more than one Class is in a source code file, only one of them may be public.
* The public class name and the filename of the source code file must match. (Example : Class Name = HelloWorld, File Name = HelloWorld.java)

1. **Class :**

* A Java source code file contains one or more Java classes.
* class is the essential Java Key Word since classes are central to Java.
* Programming in Java consists of defining a number of classes :
* Every program is a Class. (A program is defined by using one or more classes.)
* All programmer-defined types are classes.

|  |  |
| --- | --- |
| **Single Class** | **Multiple Class** |
|  |  |

1. **Statement :**

* Represents an action or a sequence of actions.
* Every Statement in Java ends with a Semicolon ( ; ).

1. **Method :**

* A collection of Statement that performs a sequence of Operation contained in a Class.
* If a Method is intended to be used to communicate with or pass information to an Object, it should be declared public.
* Parameter :
* Argument :

|  |  |  |
| --- | --- | --- |
| **Method** | **Explain** | **Example** |
| main Method | * main Method is public and visible from anywhere that can see this Class. * main Method is static, meaning that this Method can be run without creating an instance of the Class. * Key Word void indicates the Data Type returned from this Method is nothing or no value. * args is a part of the Parameter of the main Method. It takes Arguments of an Array of Strings.   args is the short form of the Arguments, it can be change to any other Parameter Name. |  |
| Other Method | * Method Declaration : ‘AccessModifier’ ‘Non-AccessModifier’ ‘DataType’ ‘MethodName’(‘Parameter’) {} * Method Invocation (Call) : ‘MethodName’(‘Argument’); * Self-Defined Method : Programmers need to declare the Method by themselves before use it. * Library Method : Programmers can use them directly without any declaration. |  |

1. **Block :**

* Groups the components of the program using the Brace ( {} ) in the program.
* Every Class has a Class Block that groups the data and the Method of the Class.
* Every Method has a Method Block that groups the data and the Method of the Class.
* Block may be nested, meaning that one Block can be placed within another.

1. **Compile & Run :**

* All command is run in the Command Prompt (cmd) of the computer.
* Prefer to execute the Command Prompt (cmd) at the folder of the Java program that need to be Compile and Run.

**A screenshot of a computer

AI-generated content may be incorrect.**

|  |  |  |
| --- | --- | --- |
| **Command Demo** | **Explain** | **Example** |
| javac HelloWorld.java | * Compile the Java program. * Notice the .java file extension is needed. * This will result in a file named HelloWorld.class being created. |  |
| java HelloWorld | * Run the Java program. * Notice there is no file extension here. (File extension not a must.) * The Java command assumes the extension is .class. |

1. **Comment :**

* Help the programmers to communicate and understand the program.
* Not a programming Statement, thus ignored by the compiler.

|  |  |  |  |
| --- | --- | --- | --- |
| **Comment** | **Symbol** | **Explain** | **Example** |
| Single Line Comment | // |  |  |
| Multiple Line Comment | /\* \*/ |  |  |
|  | /\*\* \*/ | * 生成说明文档 |  |

1. **Output :**

|  |  |  |
| --- | --- | --- |
| **Command** | **Explain** | **Example** |
| System.out.print(n); | * Output n. |  |
| System.out.println(n); | * Output n and go to new line. |

Access modifier

A screenshot of a computer

AI-generated content may be incorrect.

# Import & Package

1. **Libraries**

* Java programs are usually not written from scratch.
* There are hundreds of Library Classes for all occasions.
* Library Classes are organized into packages.
* You can import names for all the classes in a package by using a Wildcard (\*).

|  |  |  |
| --- | --- | --- |
| **Package Name** | **Class Name** | **Explain** |
| java.lang | \* | * An imported automatically into all class. * It defines System, Math, Object, String and other commonly used classes. * Therefore, java.lang is no required to import. |
| String | * Contain many Method that help with the manipulation of String Object. * Method : String(), toLowerCase(), toUpperCase(), charAt(x), length() |
| java.util | \* | * Muscellaneous utility classes. |
| Scanner |  |
| java.awt | \* | * Windowing and graphics toolkit. |
| Color |  |
| javax.swing | \* | * GUI development package. |
| JOptionPane |  |
| JButton |  |

1. **Key Words (Import & Package) :**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Key Words** | **Explain** | **Example** |
| 1. | import | * import ‘PackageName‘.’ClassName’; * import Statement at the top of the source file let you refer to Library Classes by their names. | OR |
|  | | | |
| 2. | package | * Java is a Package-Centric Language. * For good organization and name scoping, put all classes into packages. |  |

Compile & Run inside parent directory/folder :

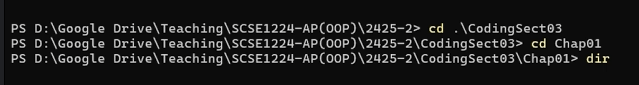
javac Pack/book/Goo.java

java Pack/book/Goo.java (The directory should you put inside the command depends on the location of you at the parent folder)

Compile & Run at anywhere :

javac -classpath E:/JavaNote E:/JavaNote/Pack/book/Goo.java

java -classpath E:/JavaNote Pack/book/Goo

****

A black background with white text

AI-generated content may be incorrect.

Cmd : dir 可以显示当前目录下的文件和子目录

Cmd : cd可以返回至上一个目录

# Identifier Naming

1. **Identifier Naming (Must Follow) :**

* All the Java Class, Variable and Method need names which is Identifier.
* Identifier must start with a Letter, a Currency Character ($) or a Connecting Character such as the Underscore ( \_ ).
* Identifier cannot start with a Number.
* Identifier can contain any combination of Letter, Currency Character, Connecting Character or Number after the first Character.
* Identifier can be any length.
* Identifier cannot be any Java Key Words.
* Identifier are case-sensitive, utm and Utm are difference.
* Identifier cannot be true, false or null.

1. **Classes & Interfaces Naming (Recommanded Naming Standard) :**

* First Letter should be capitalized.
* Several words are linked together to form name.
* The first Letter of the inner words should be uppercase.
* For classes, the names should typically be nouns. For example :
* Animal
* Account
* PrintWriter
* For interfaces, the names should typically be adjectives. For example :
* Runnable
* Serializable

1. **Methods Naming (Recommanded Naming Standard) :**

* First Letter should be lowercase.
* The first Letter of the inner words should be uppercase. (camelCase)
* The names should typically be verb-noun pairs. For example :
* getBalance
* doCalculation
* setCustomerName
* calcArea

1. **Variables Naming (Recommanded Naming Standard) :**

* First Letter should be lowercase.
* The first Letter of the inner words should be uppercase. (camelCase)
* The names should be short and meaningful names. For example :
* nama
* buttonWidth
* accountBalance
* myString

1. **Constants Naming (Recommanded Naming Standard) :**

* Java Constant are created by marking Variable static and final.
* All Letter should be uppercase with Underscore ( \_ ) as seperators. For example :
* MIN\_HEIGHT

# Data Type & Variables

1. **Data Types & Primitive Variables :**

* Once a Primitive Variable has been declared, its primitive type can never change, although in most cases its value can change.
* Primitive Variable can be declared as Class Variable (static), Instance Variable, Method Parameter or Local Variable.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Key Words** | **Bit Width** | **Range** | **Explain** | **Example** |
| 1a. | byte | 1 Bytes (8 Bits) | * -(2^7) to (2^7)-1 * -128 to 127 | * Integer. * The size of the stored Integer cannot exceed the range of the Data Type. * int is the default Data Type for Integer in the Java. * L or l must be added after the Integer if it is stored in the long Data Type is not in range of int Data Type, |  |
| 1b. | short | 2 Bytes (16 Bits) | * -(2^15) to (2^15)-1 * -32768 to 32767 |
| 1c. | int | 4 Bytes (32 Bits) | * -(2^31) to (2^31)-1 * -2,147,483,648 to 2,147,483,647 |
| 1d. | long | 8 Bytes (64 Bits) | * -(2^63) to (2^63)-1 |
|  | | | | | |
| 2a. | float | 4 Bytes (32 Bits) | * 7 Digit | * Floating Point. * The digit of the stored Floating Point cannot exceed the range of the Data Type. * The Integer stored in the float or double Data Type will automatic add a 0 at fraction part. * double is the default Data Type for Floating Point in the Java. * F or f must be added after the Floating Point if it is stored in the float Data Type.   (Integer stored in the float Data Type is allowed not put F or f.) |  |
| 2b. | double | 8 Bytes (64 Bits) | * 16 Digit |
|  | | | | | |
| 3. | char | 2 Bytes (16 Bits) | * 0 to (2^16)-1 | * Character. * The size of the stored Integer cannot exceed the range of the Data Type. * The result will show in Character form based on the ASCII Table. |  |
|  | | | | | |
| 4. | boolean | 1 Bytes (8 Bits) | * true / false | * - |  |

1. **Object & References Variables :**

* A Reference Variable is used to refer to (or access) an Object.
* A Reference Variable is declared to be of a specific type and that type can never be changed.
* Reference Variable can be declared as Static Variable, Instance Variable, Method Parameter or Local Variable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Key Words** | **Bit Width** | **Explain** | **Example** |
| 1. | new |  | * Create new Object. * ‘ReferenceVariable’ = new ‘Object’(); | --- |

1. **Instance Variables :**

* Instance Variable are defined inside the Class but outside of any Method.

A black background with white text

AI-generated content may be incorrect.

1. **Local Variables :**

* Local Variable declared within a Method.
* Local Variable starts its life inside the Method, it’s also destroyed when the Method has completed.
* Local Variable are always on the stack, not the heap.
* Local Variable declarations can’t use most of the Modifier that can be applied to Instance Variable, such as public, private, protected, transient, volatile, abstract or static.
* Local Variable can be marked final.

1. **Final Variables :**

* Final Variable declaring a Variable with the final Keyword makes it a Constant.
* Constant represent permanent data that never changes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Key Words** | **Bit Width** | **Explain** | **Example** |
| 1. | final |  | * Create Constant. * final ‘DataType’ ‘Variable’; | --- |

# Operation & Arithmetic

* 1st Precedence → Method Call
* 2nd Precedence → Unary Operators
* 3rd Precedence → new, Casting
* 4th Precedence → Arithmetic Operators (Multiplication / Division / Modulus )
* 5th Precedence → Arithmetic Operators (Plus / Minus)
* 6th Precedence → Assignment
* Solve from left to right if the Precedence to solve is same.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Symbol** | **Explain** |
| **Assignment** | = | * Used to change the value of a Variable. * “Variable” = “Expression”; * “Expression” consists of a : * Variable * Number * Mix of Variable, Number and Operator * Method Invocation * Assignment Operator is automatically executed from right to left, so Assignment Statement can be chained :   “Variable1” = “Variable2” = “Expression”; |
| **Arithmetic** | + | * Addition. |
| - | * Subtraction. |
| \* | * Multiplication. |
| / | * Division. |
| % | * Modulus. (Calculate the remainder after doing the division.) * Example : * 7 % 3 = 1 * 7 % 5 = 2 |
| **Shorthand Assignment** | += | * ‘Variable’ += ‘Expression’; same as ‘Variable’ = ‘Variable’ + ‘Expression’; * Example : a += 3 / b; same as a = a + (3 / b); |
| -= | * ‘Variable’ -= ‘Expression’; same as ‘Variable’ = ‘Variable’ - ‘Expression’; |
| \*= | * ‘Variable’ \*= ‘Expression’; same as ‘Variable’ = ‘Variable’ \* ‘Expression’; |
| /= | * ‘Variable’ /= ‘Expression’; same as ‘Variable’ = ‘Variable’ / ‘Expression’; |
| %= | * ‘Variable’ %= ‘Expression’; same as ‘Variable’ = ‘Variable’ % ‘Expression’; |
| **Unary** | + | * Positive Symbol : +‘Variable’ * Negative Symbol : -‘Variable’ * Precedence from right to left.   Example : +-+’Variable’ = +(-(+‘Variable’) = -‘Variable’ |
| - |
| **Increment**  **(Unary)** | ++ | * Preincrement : ++‘Variable’   Do the Operation first by using value ‘Variable’ original before plus ‘Variable’ with 1.   * Postincrement : ‘Variable’++   Plus ‘Variable’ with 1 first before do the Operation by using value ‘Variable’ after plus with 1. |
| **Decrement**  **(Unary)** | -- | * Predecrement : --‘Variable’   Do the Operation first by using value ‘Variable’ original before minus ‘Variable’ with 1.   * Postdecrement : ‘Variable’—   Minus ‘Variable’ with 1 first before do the Operation by using value ‘Variable’ after minus with 1. |

# Casting & Conversion

1. **Type Casting :**

* Type Casting lets you convert Primitive Value from one Data Type to another.
* Implicit Casting : The conversion happens automatically.
* Explicit Casting : Programmer tells the compiler the type to cast

|  |  |  |  |
| --- | --- | --- | --- |
| **Casting** | **Conversion** | **Explain** | **Example** |
| Implicit Casting | Widening Conversion | * Putting a smaller thing into a bigger container. (Example : byte to int) * The original Data Type will automatically become the larger one (Right Side).      * No data lose. |  |
| Explicit Casting | Narrowing Conversion | * Putting a larger thing into a smaller container. (Example : int to byte) * ‘Variable’ = (‘DataType\_Variable’)(‘Expression’);      * Might losing data. |  |
| * byte, short and char data is automatically become an int data during calculation in an Expression that involve any Operator. * Explicit Casting must be applied during a Variable of byte, short or char Data Type accept a data from the Expression involve byte, short or char data and Operator. |  |

1. **Conversion :**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Method** | **Explain** | **Example** |
| 1. | Byte.parseByte() | * Convert data to byte. * Byte.parseByte(‘Expression’) |  |
| 2. | Short.parseShort() | * Convert data to short. * Short.parseShort(‘Expression’) |
| 3. | Integer.parseInt() | * Convert data to int. * Integer.parseInt(‘Expression’) |
| 4. | Long.parseLong() | * Convert data to long. * Long.parseLong(‘Expression’) |
| 5. | Float.parseFloat() | * Convert data to float. * Float.parseFloat(‘Expression’) |
| 6. | Double.parseDouble() | * Convert data to double. * Double.parseDouble(‘Expression’) |

# Control Flow (Selection & Loop)

1. **Selection :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Key Words** | **Bit Width** | **Explain** | **Example** |
| 1a. | if |  |  |  |
| 1b. | else if |  |  |  |
| 1c. | else |  |  |  |
|  | | | | |
| 2a. | switch |  |  |  |
| 2b. | case |  |  |  |

1. **Loop :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Key Words** | **Bit Width** | **Explain** | **Example** |
| 1a. | while |  |  |  |
| 1b. | for |  |  |  |
| 1c. | do  while |  |  |  |
|  | | | | |
| 2a. | break |  |  |  |
| 2b. | continue |  |  |  |

# Library Class (java.lang.String)

* Package: java.lang.String

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Object** | **Explain** | **Example** |
| 1. | String | * String. * The String Object which already exist in the Java Standard Library. * Can store a String (Including Whitespace). |  |
| **No.** | **Method** | **Explain** | **Example** |
| 2. | String() | * Create a new String Object. * ‘Variable’ = new String(“‘String’”); |  |
| 3. | toUpperCase() | * Convert all String Character to uppercase. * ‘Variable’.toUpperCase(); * “‘String’”.toUpperCase(); |
| 4. | toLowerCase() | * Convert all String Character to lowercase. * ‘Variable’.toLowerCase(); * “‘String’”.toLowerCase(); |
| 5. | charAt(x) | * Pick certain String Character. * x is the location of the Character. * ‘Variable’.charAt(‘LocationCharacter’); * “‘String’”.charAt(‘LocationCharacter’); |
| 6. | length() | * Determine the length of a String. (Including Whitespace) * ‘Variable’.length(); * “‘String’”. length(); |

# Library Class (java.util.Scanner)

* Package : java.util.Scanner

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Object** | **Explain** | **Example** |
| 1. | Scanner |  |  |
| **No.** | **Method** | **Explain** | **Example** |
| 2. | Scanner() |  |  |
| 3. | nextLine() |  |  |
| 4. | nextInt() |  |  |
| 5. | nextDouble() |  |  |
| 6. | close() |  |  |

# Library Class (javax.swing.JOptionPane)

* Package : javax.swing.JOptionPane

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Object** | **Explain** | **Example** |
| 1. | JOptionPane |  |  |
| **No.** | **Method** | **Explain** | **Example** |
| 2. | showMessageDialog() |  |  |
| 3. | showInputDialog() |  |  |
| 4. | System.exit() |  |  |

# Class & Object & Method

1. **Classes :**

* Class may have Instance Variable to define data fields / attributes.
* Class may have Instance Method to define behaviours.
* Example Code & UML Class Diagram for class Employee :

|  |  |
| --- | --- |
| A screenshot of a computer code  AI-generated content may be incorrect. | A diagram of a class  AI-generated content may be incorrect. |

1. Object :
2. Method :

A computer screen shot of a program code

AI-generated content may be incorrect.

# Others

VScode Hint Enable / Disable : <https://stackoverflow.com/questions/72325695/how-to-disable-inline-hints-on-vscode-java>

VScode Run | Debug : <https://stackoverflow.com/questions/59970865/how-remove-run-debug-in-the-editor>

VScode caused by: java.lang.classnotfoundexception

|  |  |
| --- | --- |
| A screenshot of a computer program  AI-generated content may be incorrect. | * During error caused by: java.lang.classnotfoundexception happen, ensure the corresponding Java file that need to be executed must exist inside the JAVA PROJECTS. * If not exist, add the folder of the Java file located manually by clicking Add Folder to Java Source Path. |

this.‘variable’ :

A computer screen with white text

AI-generated content may be incorrect.

避免同样的variable名和parameter名互相影响 用this能直接标准左方是上方的variable.

return后可以直接call

A screen shot of a computer program

AI-generated content may be incorrect.

或者





A screenshot of a computer screen

AI-generated content may be incorrect.

