java -version : → whether java is installed or not

JDK

javac - version
      └→ ✓

JRE

JVM
javac

Will learn java

         └→ 1.java

.class
↑
ByteCode

Compile

javac ←

                    └→ execute / Interpret

Env. variable to configure javac

→ Edit System Env. Variable.

→ Env. Variables ⟨ user → Specific user

System → for All the System Users.

→ path → Edit → NEW → paste the path of JDK folder

C:\Program Files\Java\jdk1.8.0_271\bin

→ OK → OK → OK

→ Open New cmd and run command
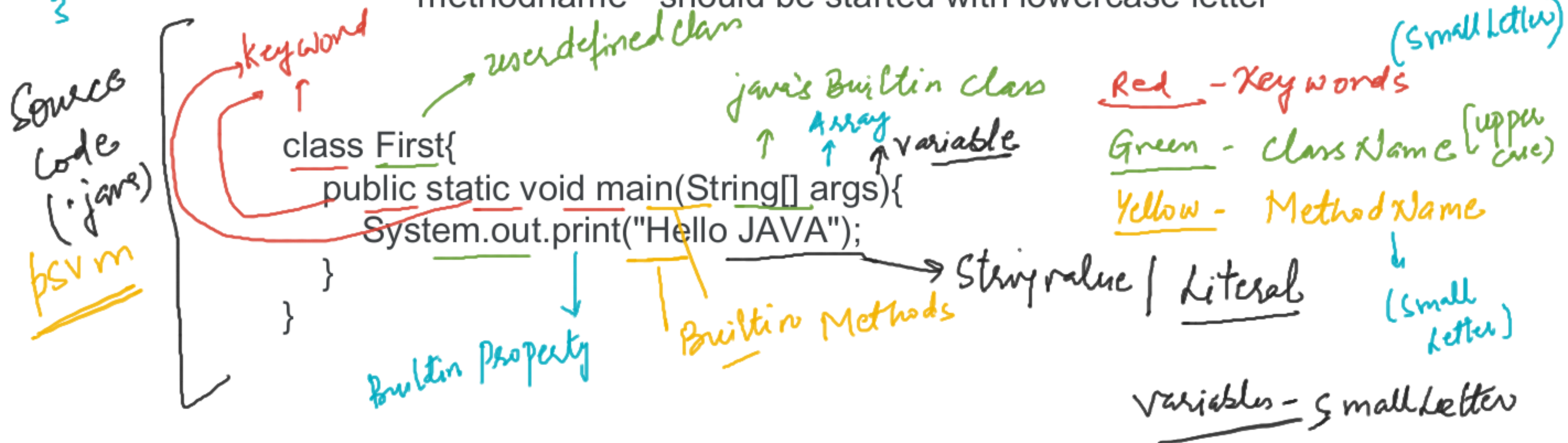
javac -version → javac 1.8

Notepad ++] →

Xotepad

Name Convention: relevant / proper name

Creating any class, fileName,methodName

Classname - First letter of class Name should be in uppercase and
File name: is also starts with upplercase letter

{
block

}

methodname - should be started with lowercase letter

keyword   userdefined class

java's Builtin class

Array

variable   (Small Letter)

Red - Keywords

Green - Class Name (upper case)

Source
Code
(.java)

psvm

```
class First{
    public static void main(String[] args){
        System.out.print("Hello JAVA");
    }
}
```

String value / Literal

Yellow - Method Name

(small Letter)

Builtin Property

Builtin Methods

variables - small Letter

(i) ClassName and FileName is different — Yes, it compiles & executes the code successfully.

(ii) ClassName & Filename is Same → Yes, "

(iii) { If we write 'public' keyword infront of class, incase ClassName and FileName is diff. → Error at compile-time.

We need to keep the file & ClassName Same, if we are using public keyword

Program Execution Flow

1. Source Code → Test.java

↓

compile → [javac Test.java]

↓

ByteCode → First .class

2. Executed Interpret the ByteCode → JVM executes it

[java First]

JVM starts the execution from main() method.

```java
public class Test{           → Creating a class
    public static void main(String[] args){        → main method from when JVM
        System.out.print("Hello JAVA");                starts the execution
    }
}
```

To print the statement