Object-Orientated Programming

CMM024

GOLOVINE (2023)

Summary

- Overview of CMM024
- Programming with Java using Visual Studio Code
- First Program (Project and main function)
- Variables
- Input and output
- Next Session

Module Organisation

- Session 1 to 4 are dedicated to procedural programming
    - Procedural programming paradigm is about computer coding that let a device perform specific tasks. Procedural code instructs a computer or other device to complete a task using logical steps
- Session 5 is reflection week…. Time to practice what you learned in previous weeks!
- Session 6 to 10 are dedicated to object-orientated programming (OOP)
    - A programming paradigm that relies on the concept of classes and objects. It is about structuring a software program into simple, reusable pieces of code blueprints (called classes), which are used to create individual instances of objects. There are many object-oriented programming languages, including C++, Java, and Python.

Session 1 (Procedural Programming)

Introduction to:

Programming

Variables

I/O Functions

CMM024

What is Java

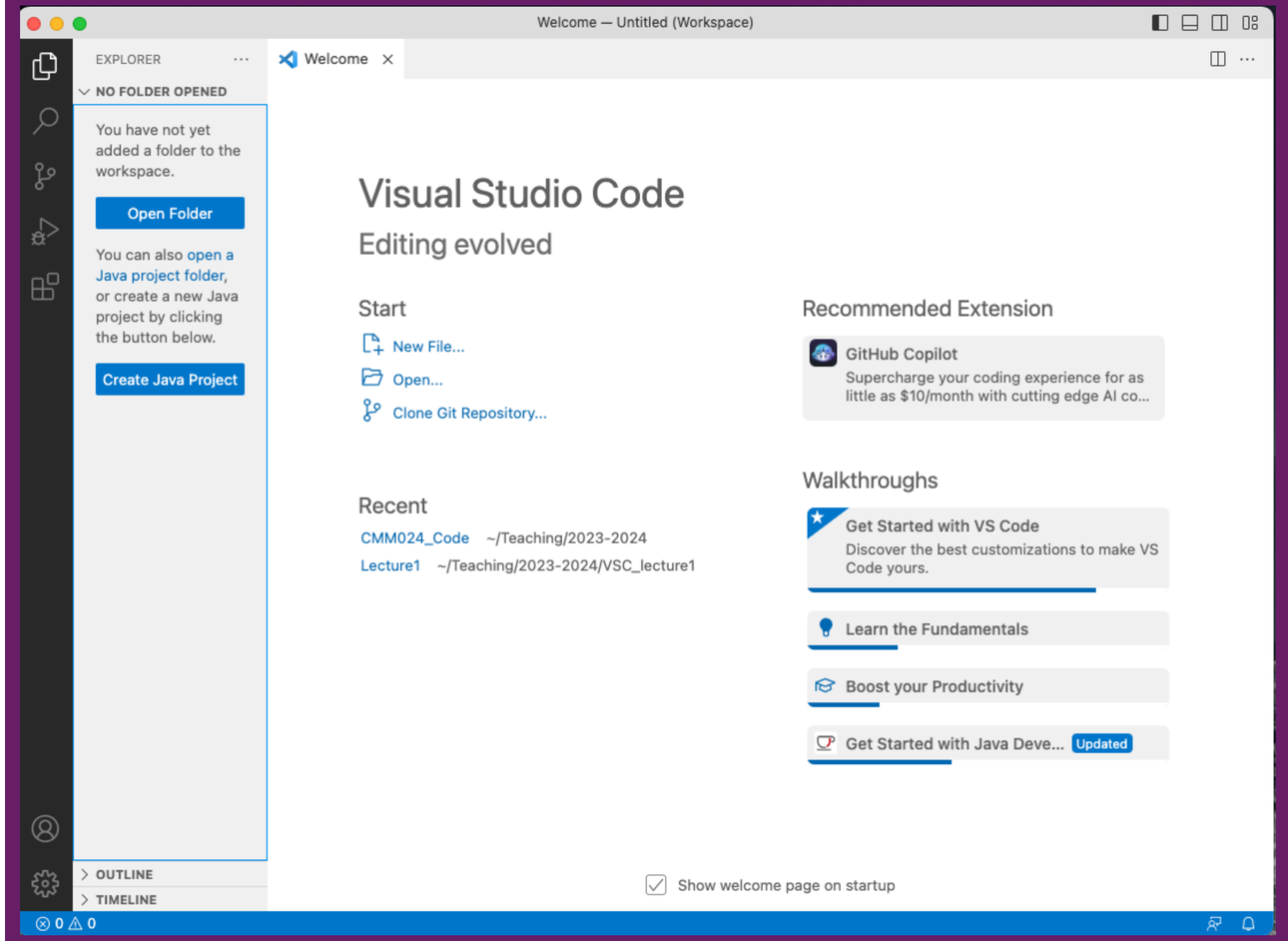
- Class-based and object-oriented: You can create objects that can be used across projects, saving you time
- Portable: Java runs on a ‘write once, use anywhere’ principle, meaning that your can be used on other projects on any platform
- Secure: Once created, all Java code is converted to bytecode, which cannot be read by humans and thus secure
- Note just two facts:
  - 3 billion mobile phones run Java
  - 125 million TV devices run Java!

Visual Studio Code

Visual Studio Code is a free and open source integrated development environment for development on Windows, Mac and other platforms. It is created by Microsoft.

The Code Editor simplifies the development java project, and by adding other extensions you can program using other languages etc..

VSC works on Projects. Hence you must create a project, then you can create Java files.



Visual Studio Code

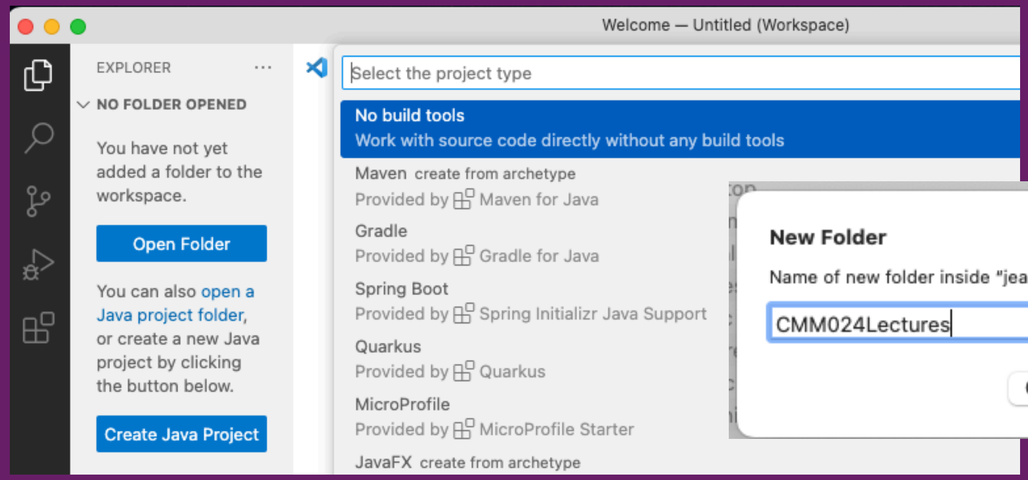
Creating a project

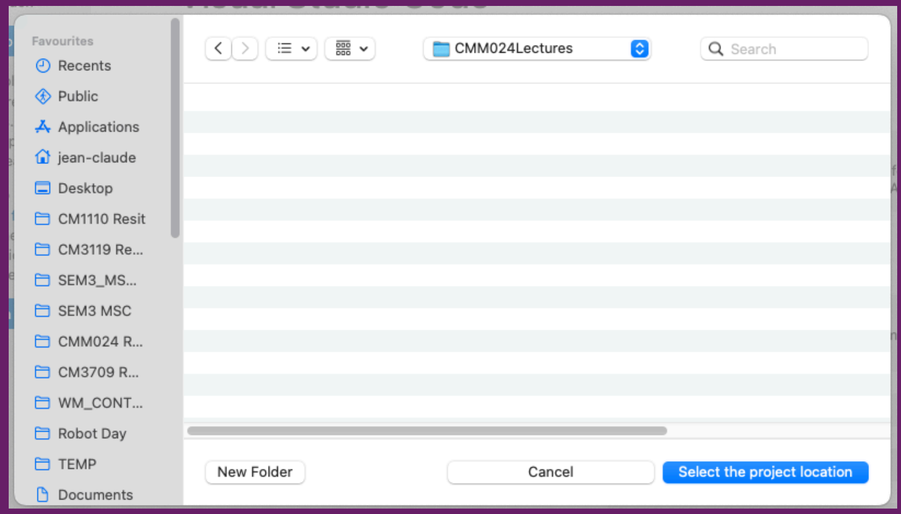
Let’s create your first project!

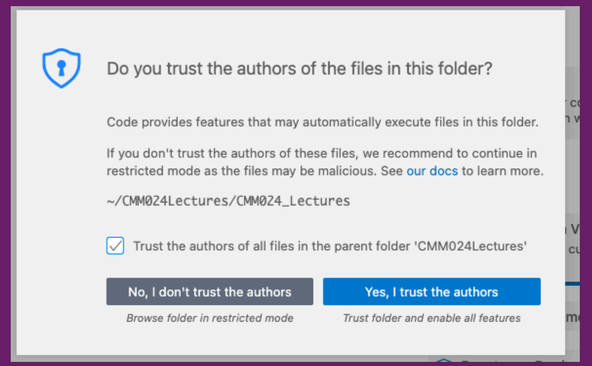
Here is an overview. First you create a project folder where all your project folders and java files will reside, and then you name the project itself.

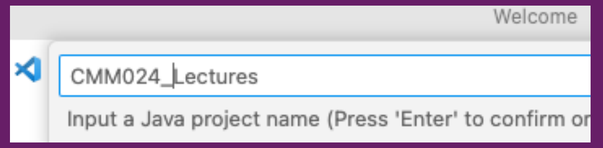
We will call it *CMM024Lectures*

LET’S CODE









First Program(Demo)

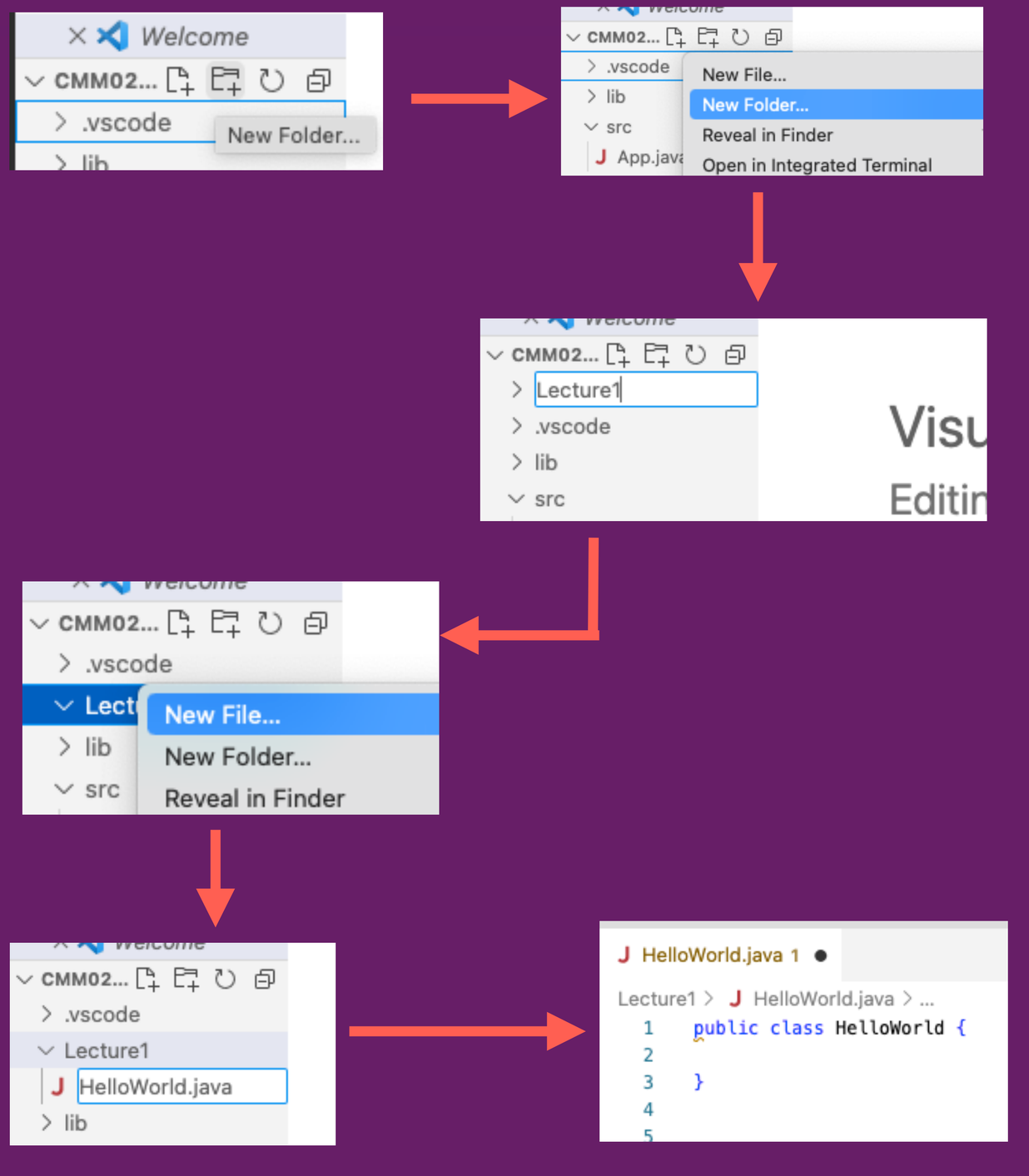
First we need to create a folder (I.e. project folder), then we can add subfolders (packages) that will contain our Java files.

Finally, we can code a simple program that display “Hello World” !

```
public static void main(String[] args){
    System.out.println("Hello World");
}
```

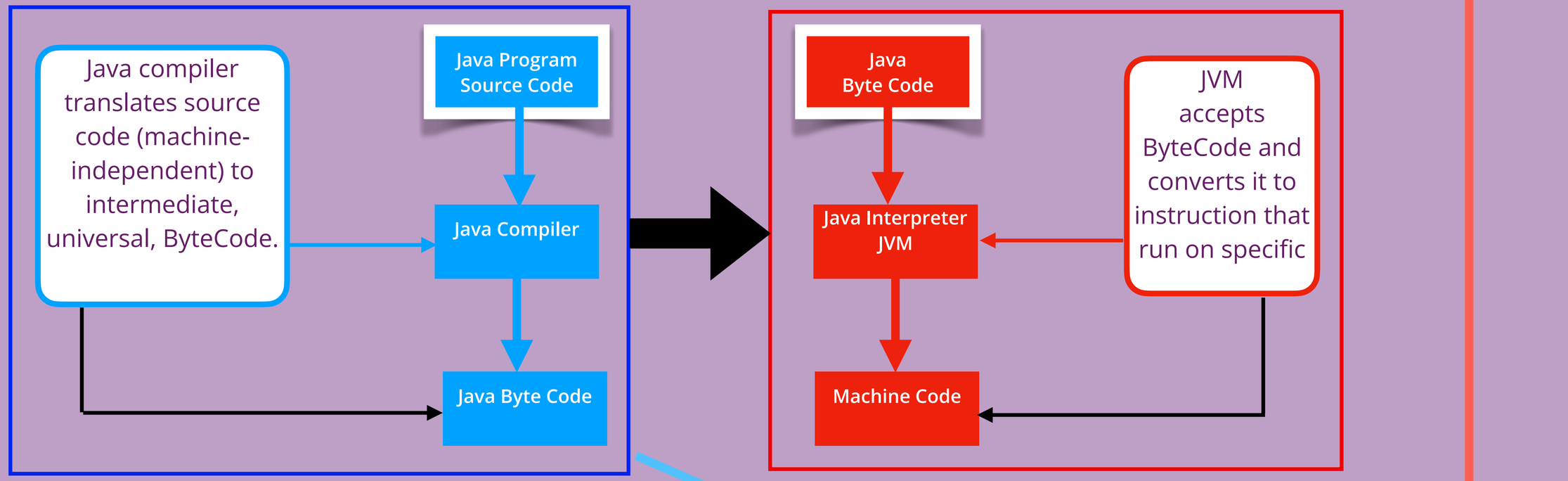
```
PROBLEMS     OUTPUT     DEBUG CONSOLE

jean-claude@MacBook-Pro-2 CMM024_Lect
 -XX:+ShowCodeDetailsInExceptionMessa
Hello World
jean-claude@MacBook-Pro-2 CMM024_Lect
```

LET’S CODE



How your Java code is run

10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000



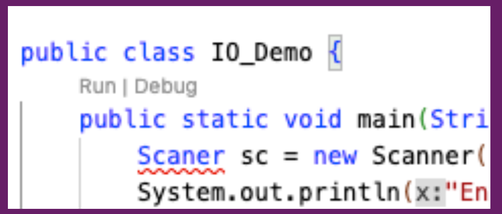
- First you create some Java code
- Second, the code (if no compile errors) is turned to byte code in a file with extension ".class"
- Third, the byte code (in the file ".class") is passed to the JVM, which turn that code to machine code
- Fourth, the machine code is run and we should get the desired output

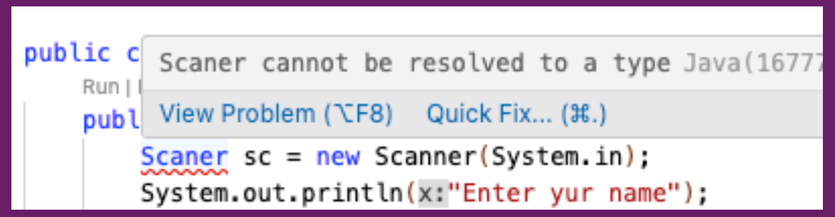
```
public int getX() {
  return x;
}

public void setX(int x) {
  this.x = x;
}
```

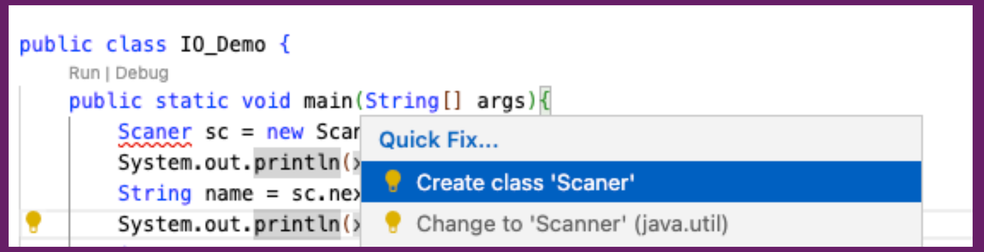
```
6
7   public getX()i
8     aload 0
9     getfield Example.x : i
10    ireturn
11
12  public setX(i)v
13    aload 0
14    iload 1
15    putfield Example.x : i
16    return
```

Errors (Demos)

- The most common error is spelling mistakes
- When VSC tell us cannot resolve, it means it does not know that the word with error is about. If you click on Quick Fix, he wants to create the class, which means it cannot find anything called “Scaner”.
- The reason is it is called “Scanner” !

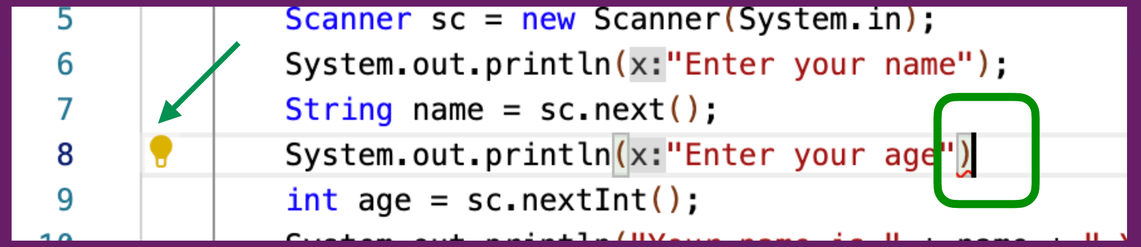


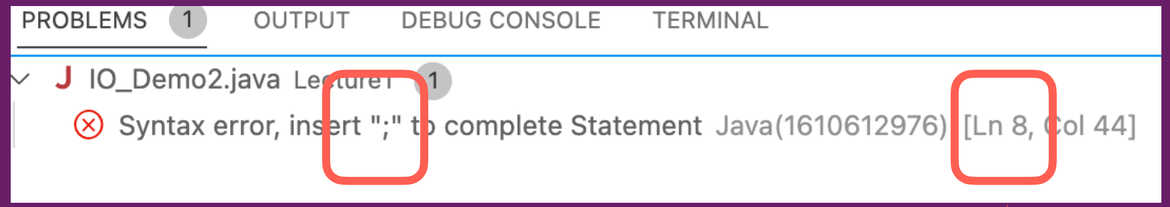




Errors (Demos)

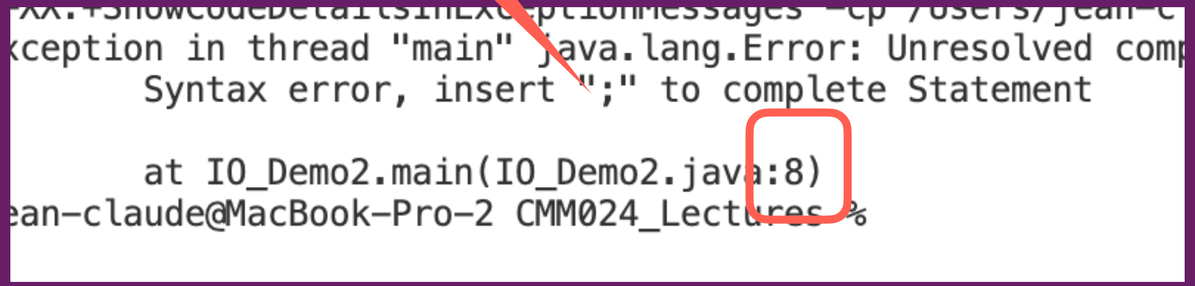
- The next most common is not finishing your line of code with a semi colon
- You must terminate your lines (i.e., statements) with a semi colon!
- If you try to run the code, the compiler will tell which line contains the error, often what to do!
    - ✳ Here, “insert ; to complete statement.





IF YOU COMPILE THE CODE

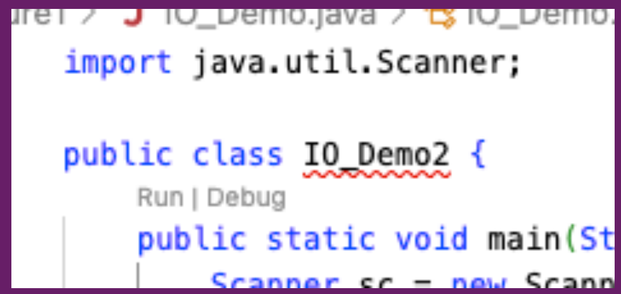
ERROR LINE 8

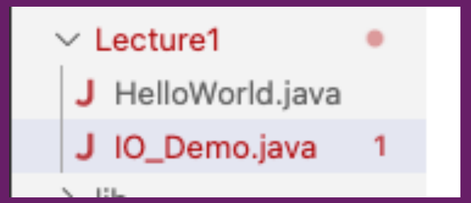


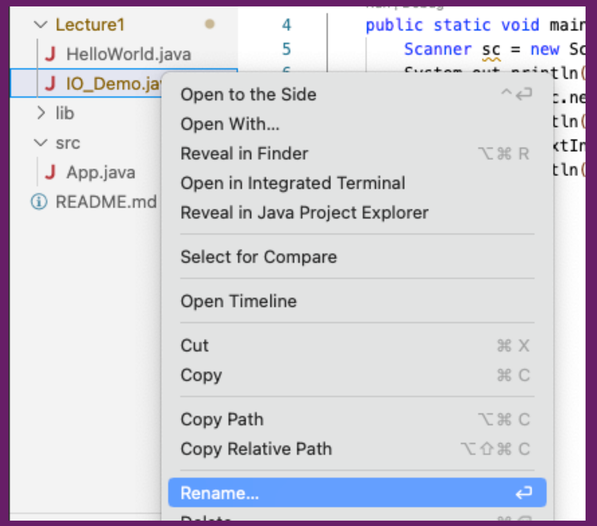
Errors (Demos)

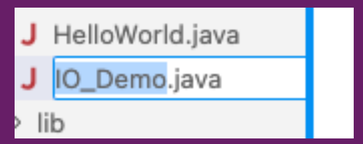
- One of the typical error is to rename a class without changing the file name. You must use the Refactor tool!
- Here you try to rename the class but you must bear in mind that the filename MUST be the same of the name of the class. Hence here we have an error. The name of the file is red.
- In VSC, renaming the file will call the refactor tool and ask you to continue by pressing OK.

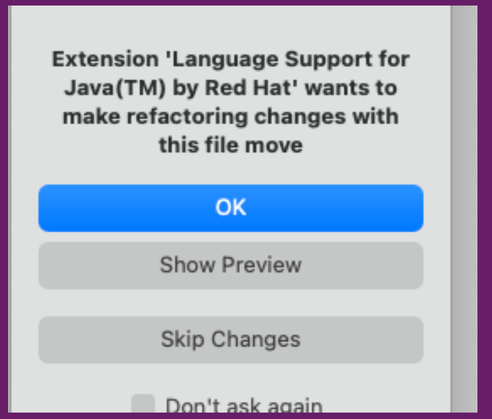
I just renamed the class manually

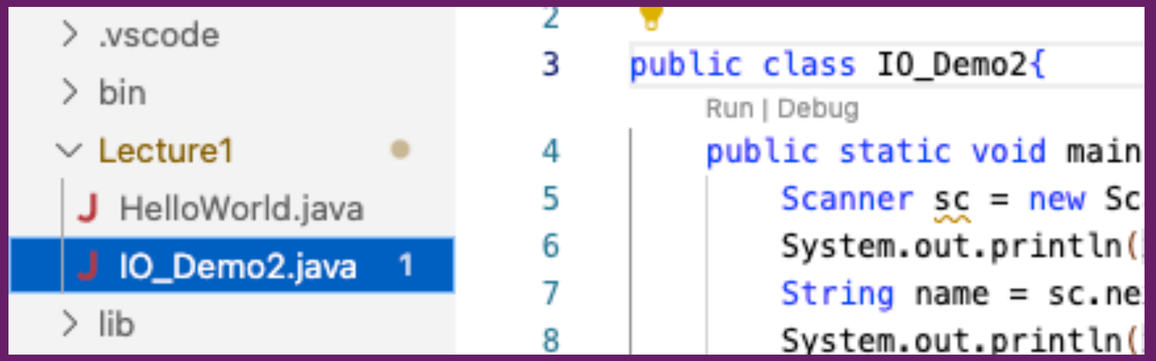








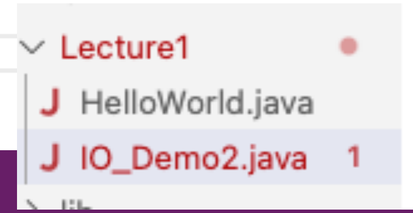


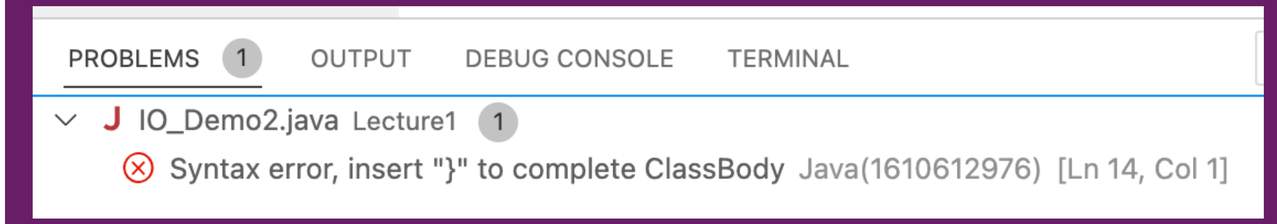


Errors (Demos)

- Another common error is to forget brackets. As we will see, brackets define scopes for a lot of things in programming, one missing will lead to compile error.
- Here I commented out one bracket. Was soon as I do this, a compile error is generated.
- If you try to run the code, VSC will tell you what is happening. Just comply!
- Red is always bad news….

```
2
3   public class IO_Demo2 {
        Run | Debug
4       public static void main(String[] args){
5           Scanner sc = new Scanner(System.in);
6           System.out.println(x:"Enter your name");
7           String name = sc.next();
8           System.out.println(x:"Enter your age");
9           int age = sc.nextInt();
10          System.out.println("Your name is " + name + " You age is " + age);
11          sc.close();
12
13      //}
14  }
15
```






```
Exception in thread "main" java.lang.Error: Unresolved c
        Syntax error, insert "}" to complete ClassBody

        at IO_Demo2.main(IO_Demo2.java:14)
jean-claude@MacBook-Pro-2 CMM024_Lectures %
```

Errors (Demos)

- Java is strongly typed. Everything in Java must relate to some specific type.
- Here, name as no type so Java does not know what to do with name and how to handle it.
- You must always use a type Java knows about. A class name, a primitive type etc.
- If you run the code, the java compiler tells you it cannot resolve NAME to a variable. This is true as a variable MUST have a type.

```
4       public static void main(String[] args){
5           Scanner sc = new Scanner(System.in);
6           System.out.println(x:"Enter your name");
7           name = sc.next();
8           System.out.println(x:"Enter your age");
9           int age = sc.nextInt();
10          System.out.println("Your name is " + name + " You
11          sc.close();
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

IO_Demo2.java Lecture1 2

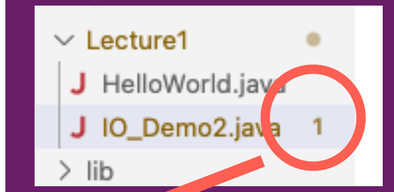
name cannot be resolved to a variable Java(33554515) [Ln 7, Col 10]

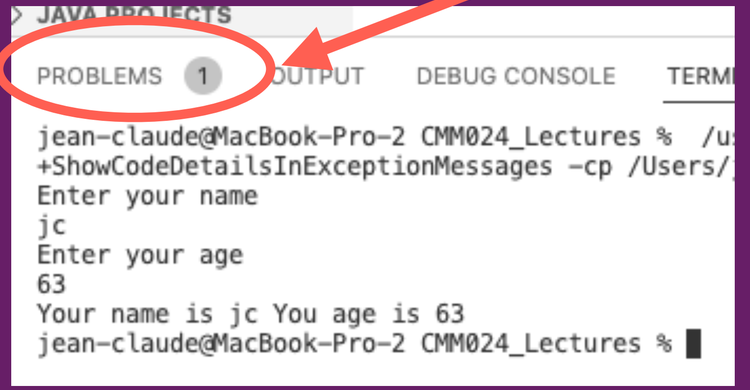
name cannot be resolved to a variable Java(33554515) [Ln 10, Col 46]

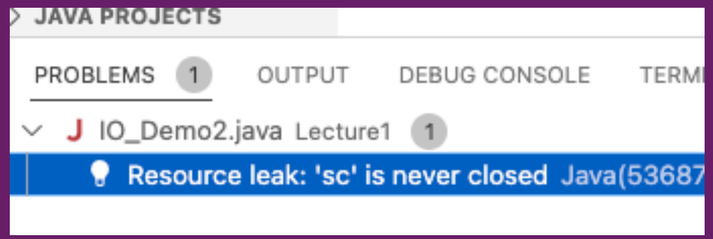
```
/bin IO_Demo2
Exception in thread "main" java.lang.Error: Unresolv
        name cannot be resolved to a variable
        name cannot be resolved to a variable

        at IO_Demo2.main(IO_Demo2.java:7)
jean-claude@MacBook-Pro-2 CMM024_Lectures %
```

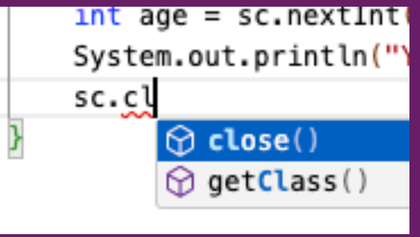
Errors (Demos)

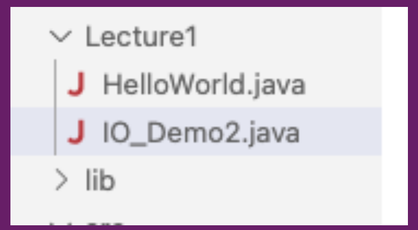
- There are errors that are more difficult to deal with.
- Here is an example. The code appears to be fine (no red) but the name is brown. This means that there is an error but it is not severe. What is it?
- Bottom of the script, problems has 1 entry. Click on it.
- Java tells us that the input scanner must be closed. It was opened when we created it.
- Adding the line “sc.close();” will resolve all the problems. The file name is normal colour meaning that is error free (red otherwise) and problem free (brown otherwise)











Another Demo (practice)

A more complicated program that asks the user some information and display it on the console

First, create another Java file called IO_Demo.java and enter the code

```
import java.util.Scanner;

public class IO_Demo {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your name”);
        String name = sc.next();
        System.out.println("Enter your age");
        int age = sc.nextInt();
        System.out.println("Your name is " + name
           + " You age is " + age);
    }
}
```

LET’S CODE

```
jean-claude@MacBook-Pro-2 CMM024_Lect
_Lectures ; /usr/bin/env /Library/Jav
ents/Home/bin/java -XX:+ShowCodeDetai
Lectures/CMM024_Lectures/bin IO_Demo
Enter yur name
jcg
Enter your age
63
Your name is jcg You age is 63
jean-claude@MacBook-Pro-2 CMM024_Lect
```

Variables

- Unlike some other reprogramming language Java has specific types. It is what is known was “statically- typed” or “strongly-typed”.
- You have numerical values (1, 220, 9.99) corresponding to specific types (int, double, float, byte, short)
- You have character based types such as char “A” or a combination of characters “Golovine” called a String with a corresponding type known as String. There is another type of java character based that can only be only “true, false” known as the boolean type. (No maybe!)
- All these types apart from String, are known as primitive types with their first character started with a lower case character. Notice that String has a uppercase one and thus is an Class name.
- Another important type you will see later is the array. Once the length is set, it cannot be changed. However, there another similar type ArrayList (like String it is a class) which is more flexible and is dynamic.

Numerical type recap

Java has 8 (so called "primitive") variable types:	
Name of data type	**Type of data it holds**
byte	whole number values i.e. integer values byte uses 8-bits short uses 16-bits int uses 32-bits long uses 64-bits
short	
int	
long	
float	floating point values i.e. numerical data types with fraction parts float uses 32 bits double uses 64 bits, more precise than float
double	
char	single text characters
boolean	logical (true or false) values

Maximum values

byte b = 100

short s = 10000

int i = 1000000000

long num = 1000000000000000000L

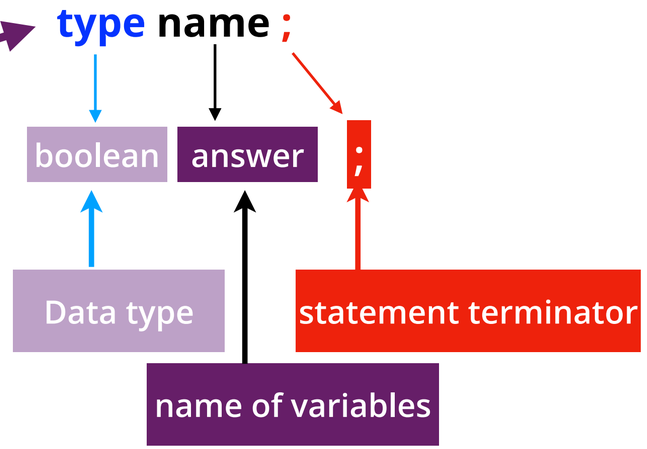
float f =
100000000000000000000000000000000000000.0F

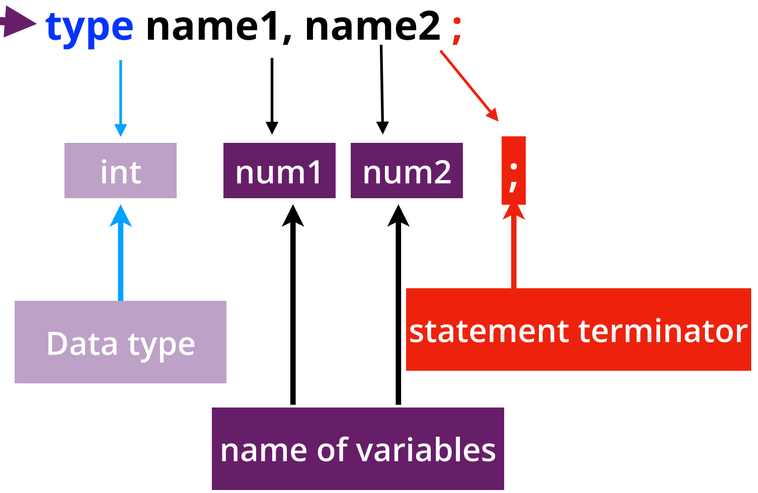
double d =
1000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
0000000.0;

Syntax of Variable Declaration

```
boolean answer; //declares a boolean variable
int num1, num2; //declares 2 integer variables
double x, y, z; //declares 3 double variables
```

Note: each variable name is separated by a coma!





Syntax of Variable Declaration

Note: each statement is separated by a coma!

```
boolean answer;
answer = true;

double pi;
pi = 3.1415;

int num1 = 1234, num2 = 7;

double length = 5.43, height = 2.666;

boolean answer1 = true, answer2 = false;
```

If you assign a value that is not the right type, the compiler will show an error and the code will not compile as 9.32 is either a float or a double type but not a boolean!

```
boolean answer;
answer = 9.32;
```

```
System.out.println("Your name is " + name + " You age is " + age)

                Type mismatch: cannot convert from double to boolean
boolean a       View Problem (⌥F8)    Quick Fix... (⌘.)
answer = 9.32;
```

Arithmetic in Java Demo

In order to make a program running you must first create a class, which is given a name.

Then you must write the main function as it is in the screenshot. If you do not your program will never run.

Then, you can write your code inside the main method. You can declare some variable and at the same time assign a value to them. You can change their value at any time.

Then you can compute some arithmetic here I + J and I - J.

We will explain more on class and the main method in later sessions.

```
public class FirstArithmeticCalculations {

    public static void main(String[] args) {
        //declare and assign value to integers i & j
        int i = 8;
        int j = 4;
        //display these values
        System.out.println("i = " + i + " j = " + j);
        //re-assign j to a value of 11
        j = 11;
        //display j value
        System.out.println("J is now " + j);
        //compute i+j and i-j and display these values
        System.out.println("i + j = "
                + (i + j) + " i - j " + (i - j));
    }

}
```

```
run:
i = 8 j = 4
J is now 11
i + j = 19 i - j -3
BUILD SUCCESSFUL (total
```

LET'S CODE

Arithmetic (important notes)

- One of the issues when conducting arithmetic operations in a strongly type language like Java is what type the result will be?
- In general, when using different types, the result will be similar to the one that is more precise. If you use a int and double and divide both, the result will be a double

```
i = 5;
d = 2;
System.out.println("d / i = " + (d / i));
System.out.println("i * d = " + (i * d));
```

```
d / i = 0.4
i * d = 10.0
```

- If you divide two integers, the same applies, which means that even though the result can be a decimal value, the result will be an integer, and thus you will get truncation

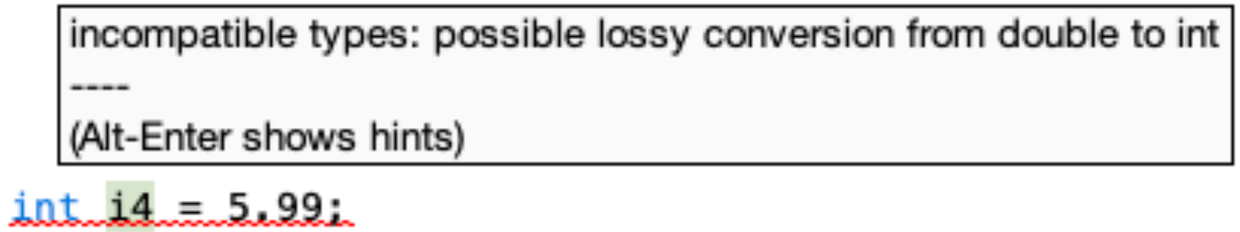
```
i = 5;
i2 = 2;
System.out.println("i / i2 = " + (i / i2));
System.out.println("");
```

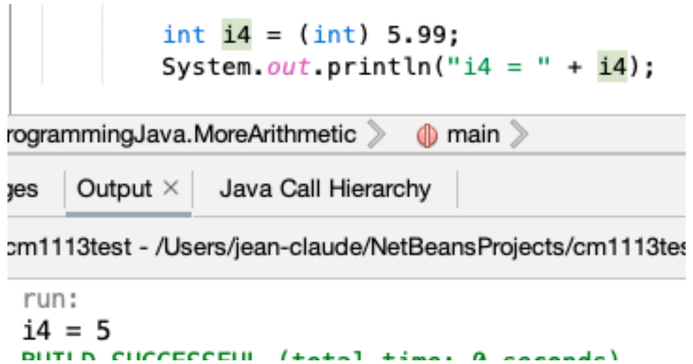
```
i / i2 = 2
```

Arithmetic (Converting)

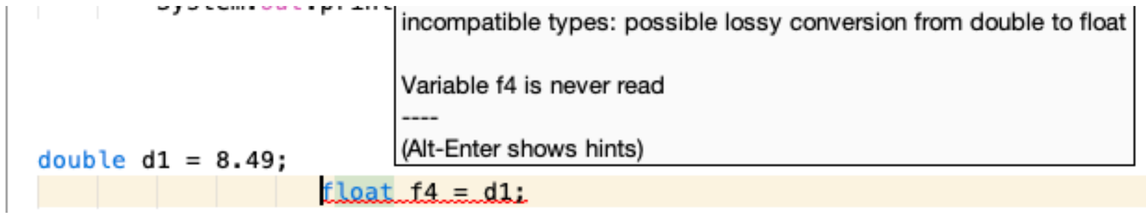
- Another issue is converting between different numerical types. For instance, if you try to convert a decimal value 5.99 into an integer, which is a whole number, you will loose the decimal part and end up with 5. However Java will not let you do this unless you cast that decimal value into an int

- As you can see an error is generated. You can force Java to comply by casting here “(int)” and transform the decimal value to an integer, thus removing the decimal part.





- The same applies to float and double with the float being less precise as the double.



Using the Scanner class (Demo)

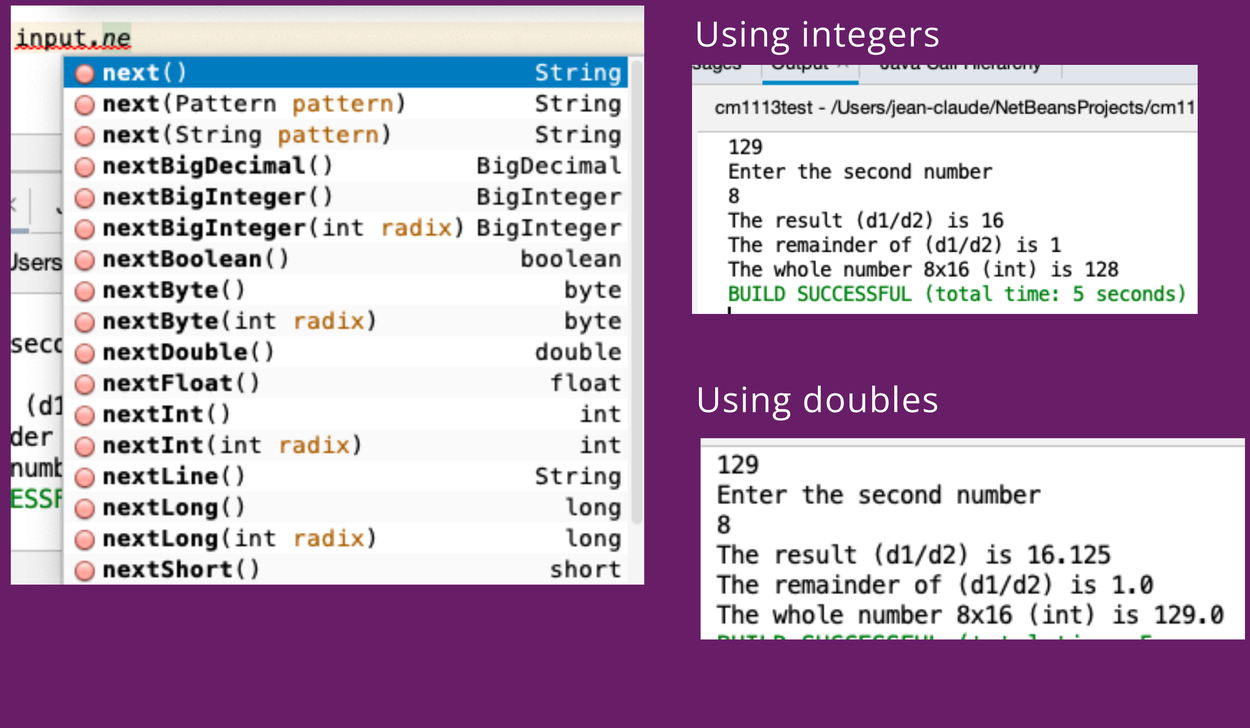
The Scanner is a very good way to get user input from the console. It will provide you with many input types, the default (next()) being a String.

In this example, we use the console to enter two integers, divide the first one by the second (remember the rule about loosing the decimal part) find the result (it will be an integer) and the remainder after the operation is conducted.

Had we used doubles, the result would have been 16.125, but the remainder would still be 1.0 as it assumes remainder after a whole number division.

LET'S CODE

```
public static void main(String[] args) {
      int i1;
      int i2;
      int remainder;
      int result;

      Scanner input = new Scanner(System.in);
      System.out.println("Enter the first number");
      i1 = input.nextInt();
      System.out.println("Enter the second number");
      i2 = input.nextInt();
      result = (i1 / i2);
      remainder = (i1 % i2);
      System.out.println("The result (i1/i2) is " + result);
      System.out.println("The remainder of (i1/i2) is " + remainder);
}
```



Next week

- Loops (*for*, *while* and *do while*)
- Conditional statements - Control flows (*if and switch*)
- Look at in built math methods
- Go over examples of how to use these constructs