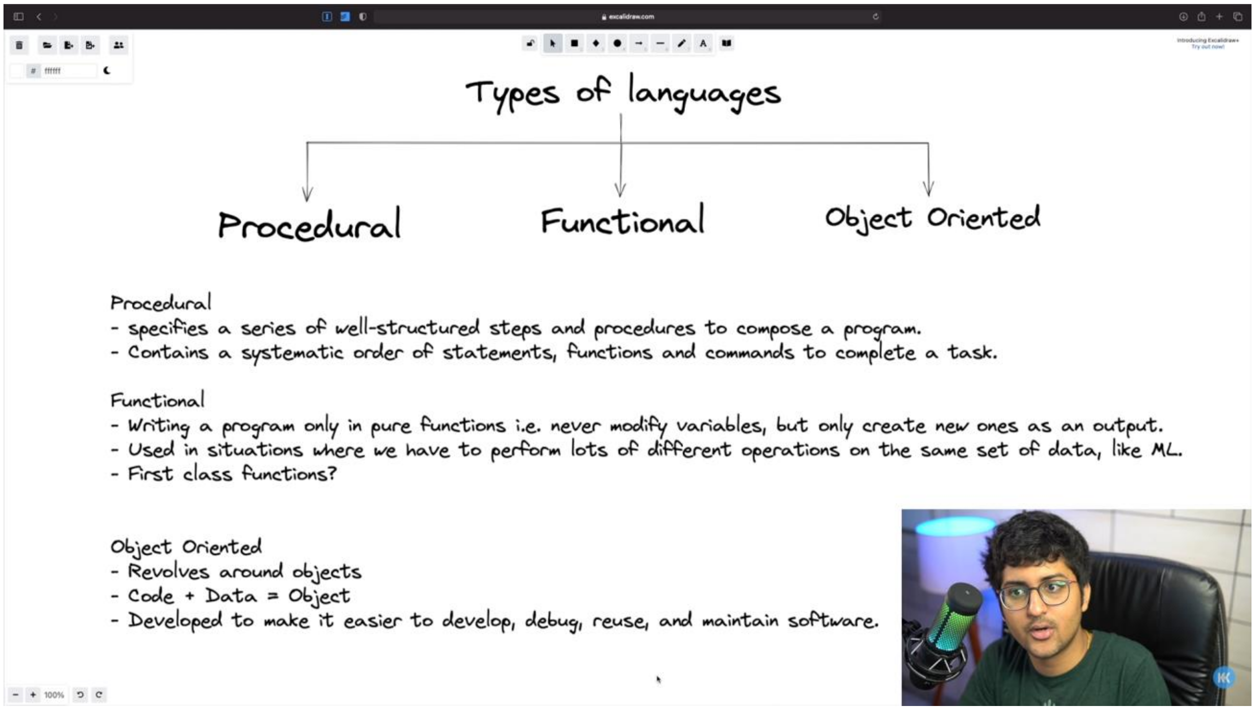


Types of languages

Procedural

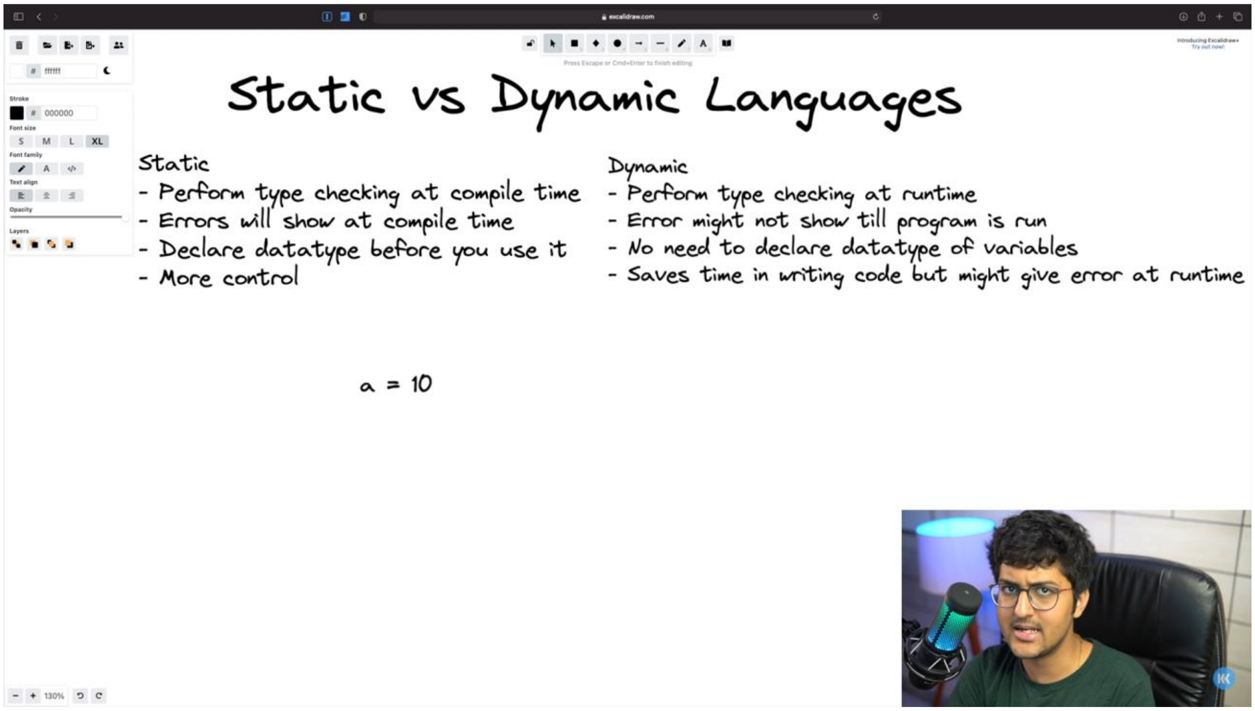
Functional

Object Oriented

Procedural
- specifies a series of well-structured steps and procedures to compose a program.
- Contains a systematic order of statements, functions and commands to complete a task.

Functional
- Writing a program only in pure functions i.e. never modify variables, but only create new ones as an output.
- Used in situations where we have to perform lots of different operations on the same set of data, like ML.
- First class functions?

Object Oriented
- Revolves around objects
- Code + Data = Object
- Developed to make it easier to develop, debug, reuse, and maintain software.

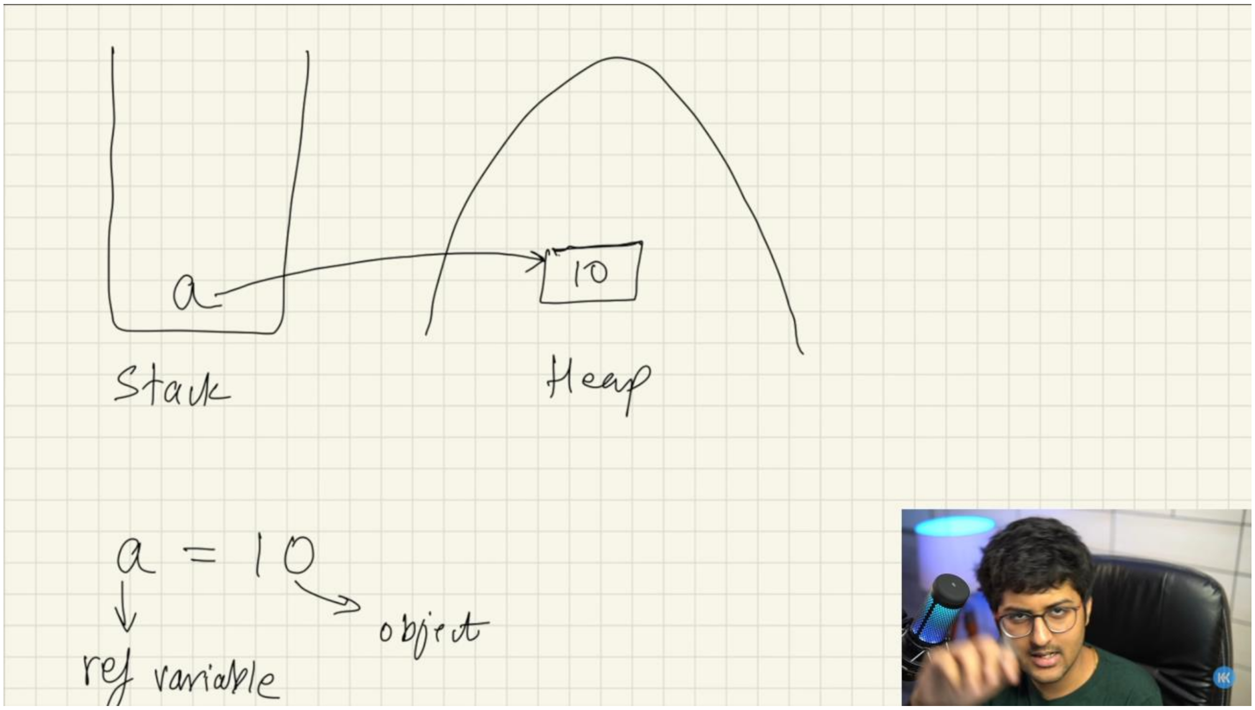


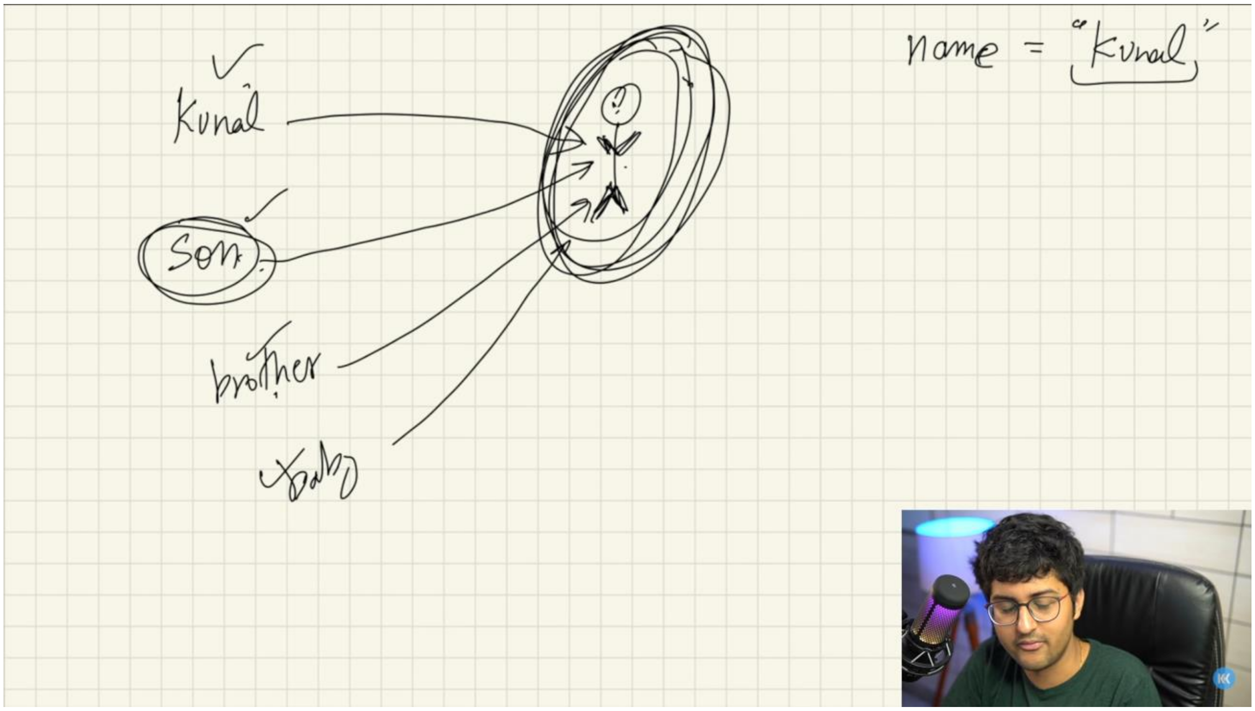
Static vs Dynamic Languages

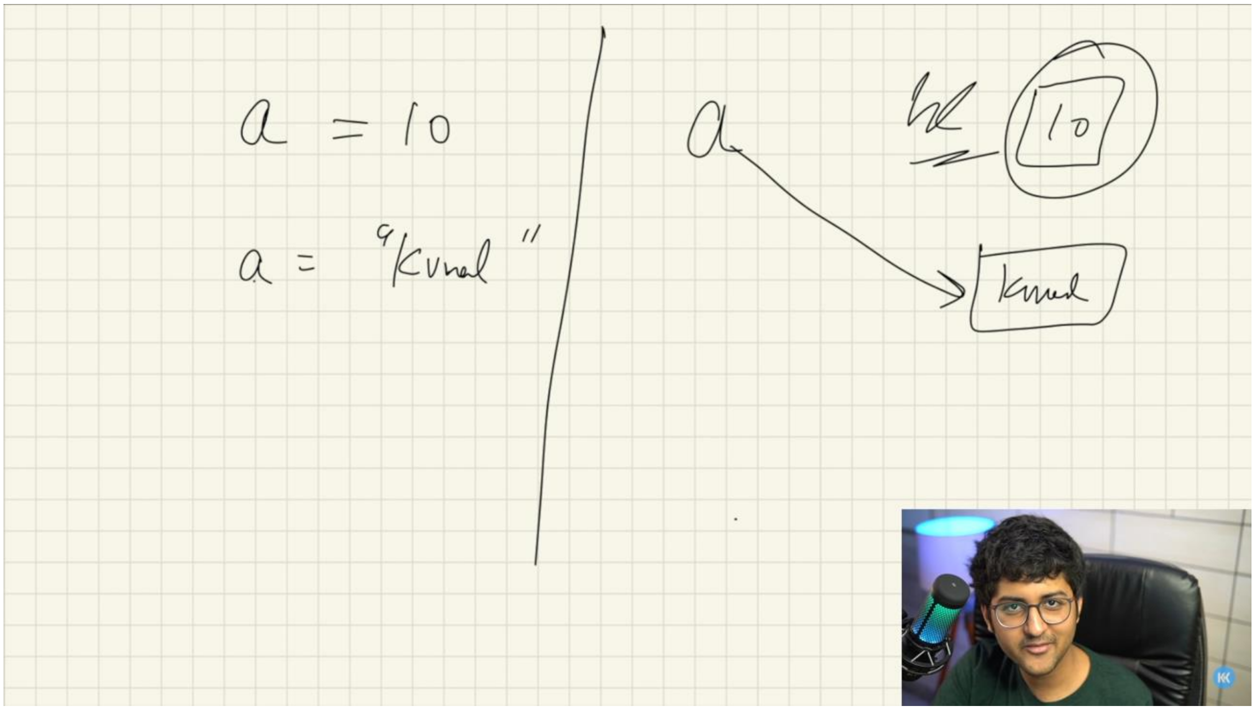
Static
- Perform type checking at compile time
- Errors will show at compile time
- Declare datatype before you use it
- More control

Dynamic
- Perform type checking at runtime
- Error might not show till program is run
- No need to declare datatype of variables
- Saves time in writing code but might give error at runtime

a = 10


a
10
Stack
Heap
a = 10
ref variable
object

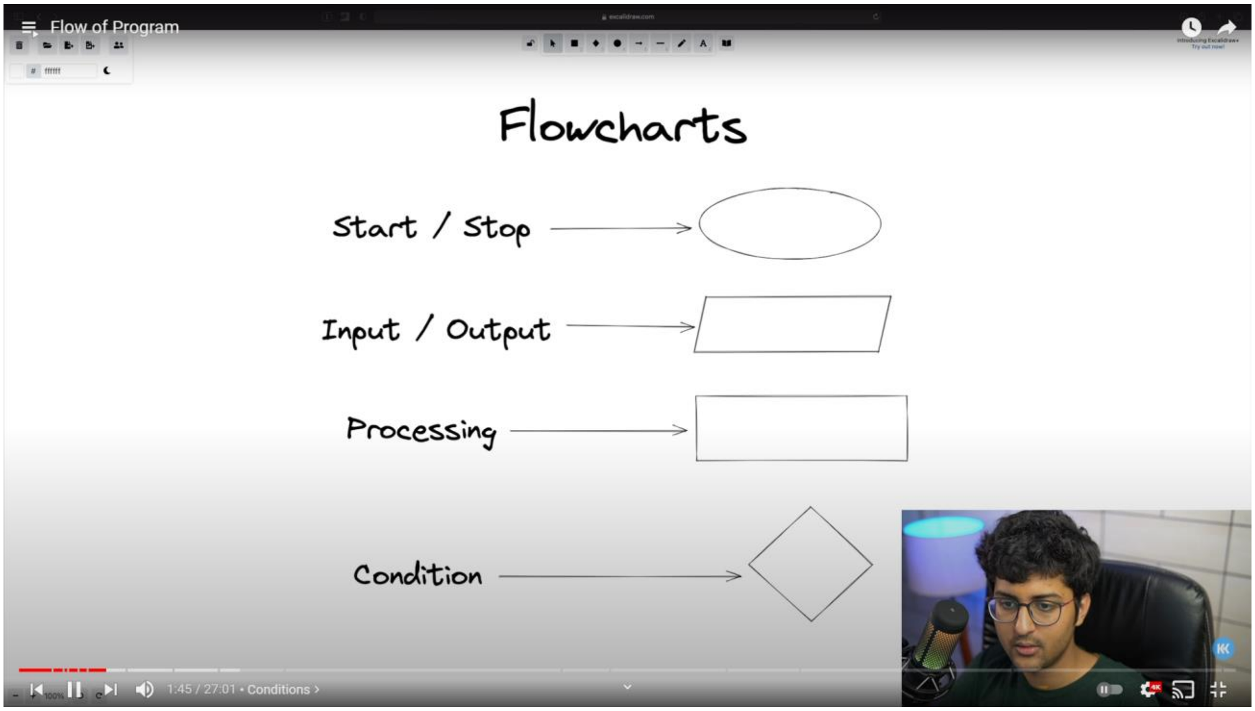

Kunal
Son
brother
baby
name = "Kunal"

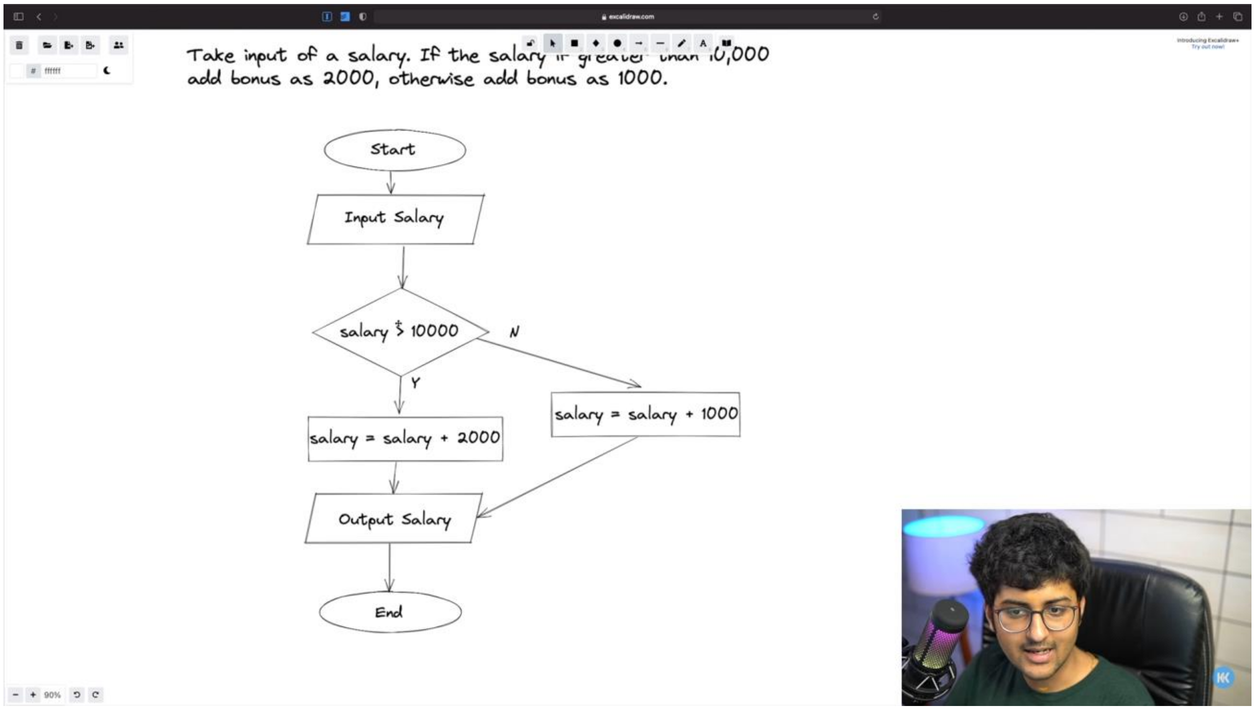


a = 10

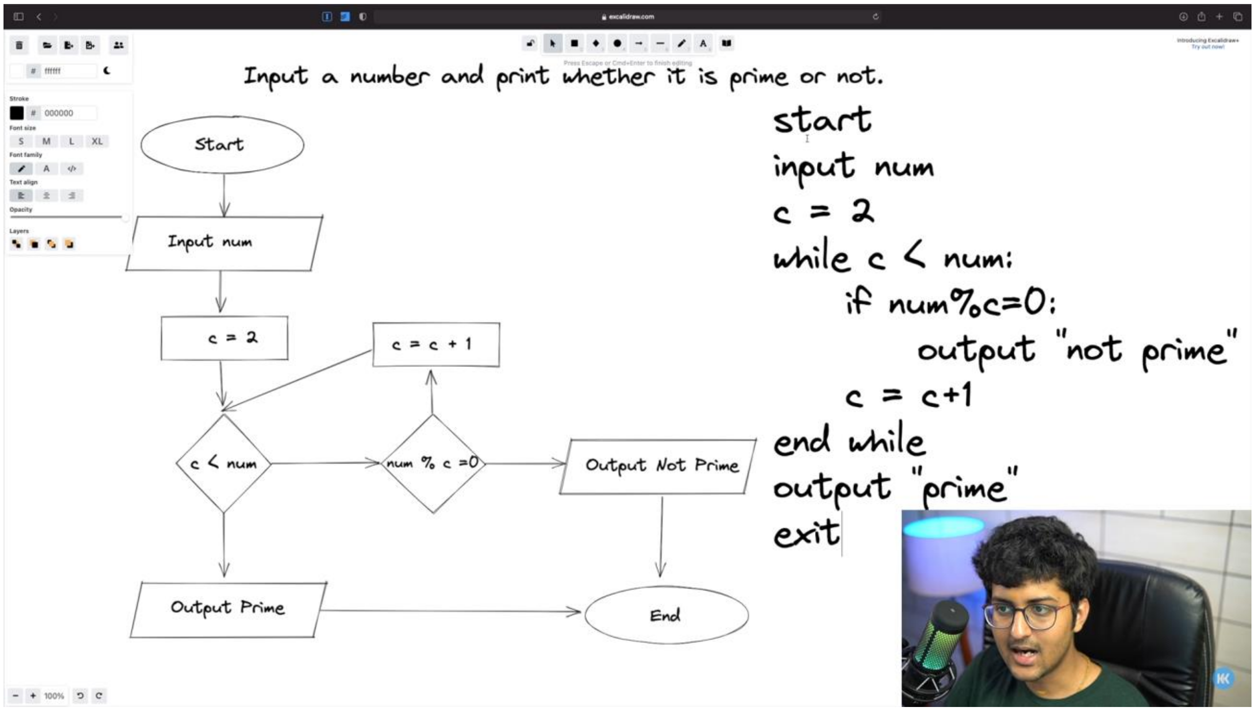
a = "Kunal"

Flow of Program



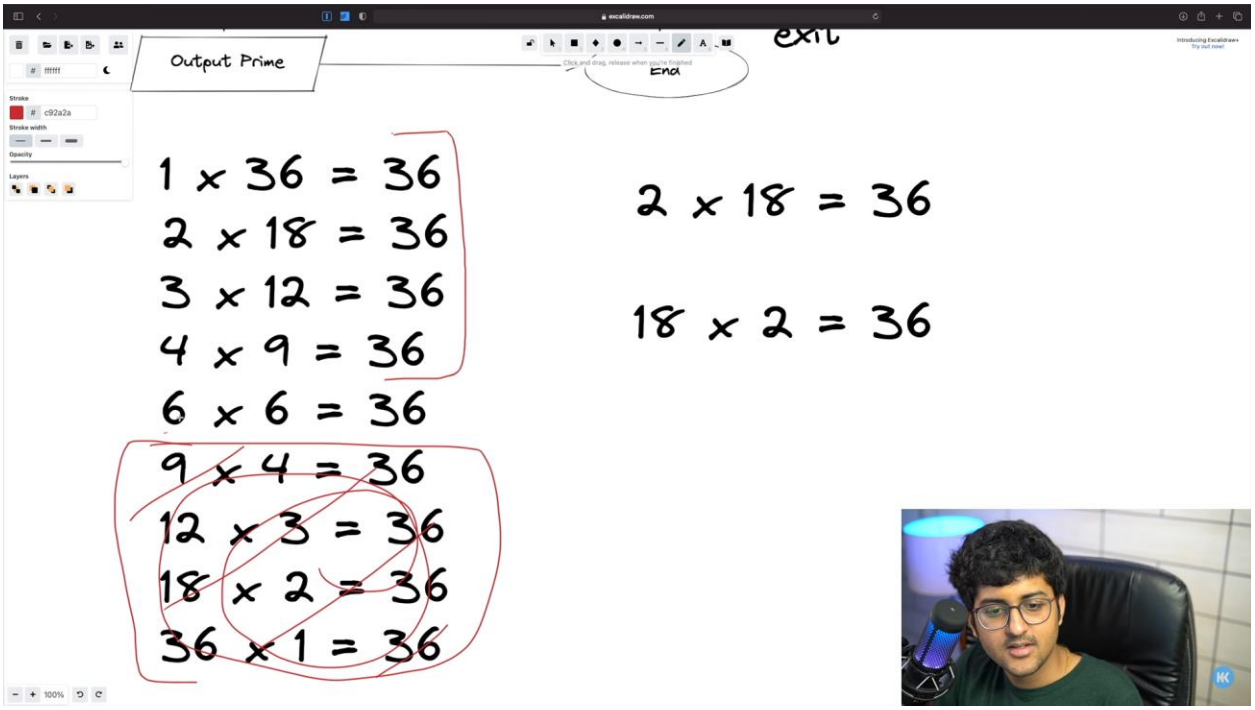


Take input of a salary. If the salary is greater than 10,000 add bonus as 2000, otherwise add bonus as 1000.



Input a number and print whether it is prime or not.

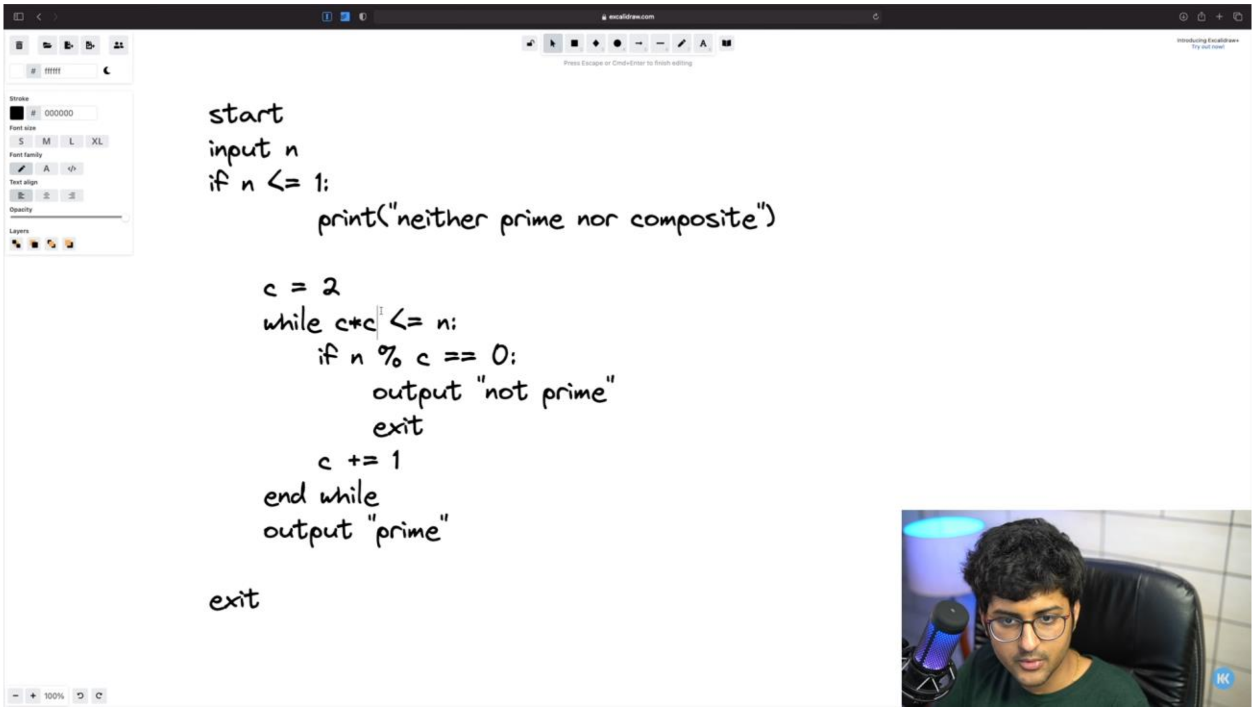
```
start
input num
c = 2
while c < num:
    if num%c=0:
        output "not prime"
    c = c+1
end while
output "prime"
exit
```



1 x 36 = 36
2 x 18 = 36
3 x 12 = 36
4 x 9 = 36
6 x 6 = 36
9 x 4 = 36
12 x 3 = 36
18 x 2 = 36
36 x 1 = 36

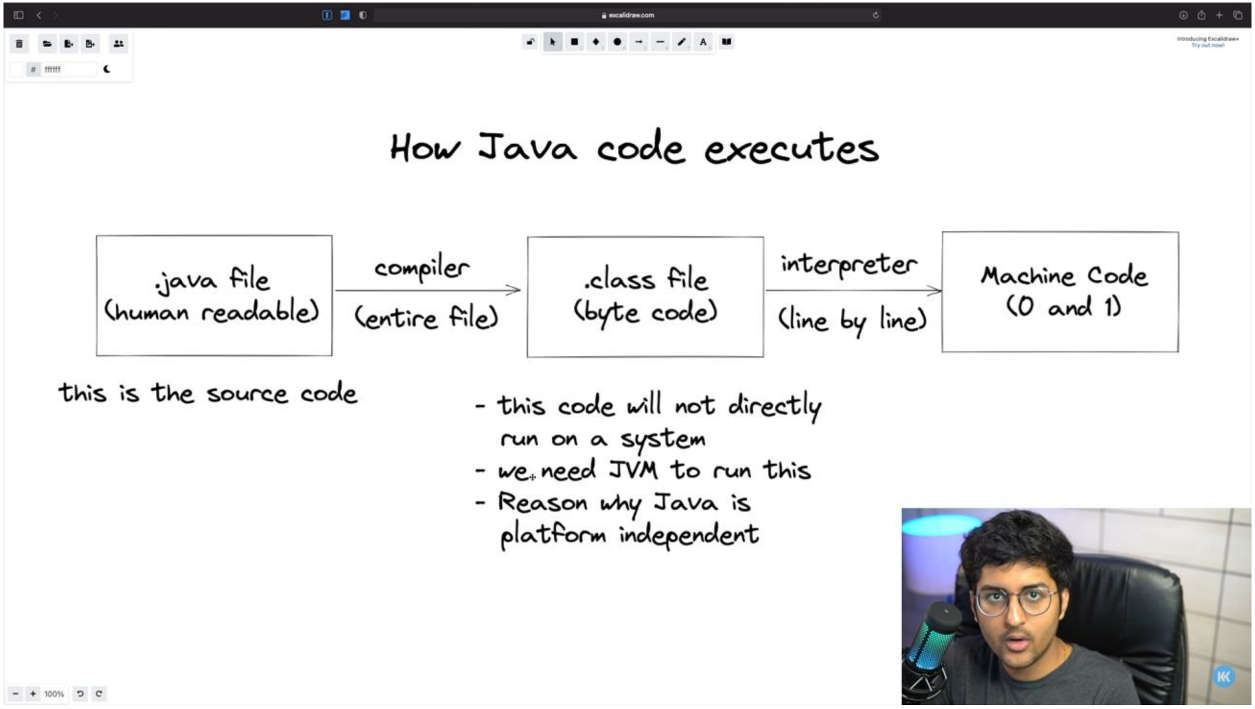
2 x 18 = 36

18 x 2 = 36



```
start
input n
if n <= 1:
        print("neither prime nor composite")

    c = 2
    while c*c <= n:
        if n % c == 0:
            output "not prime"
            exit
        c += 1
    end while
    output "prime"

exit
```

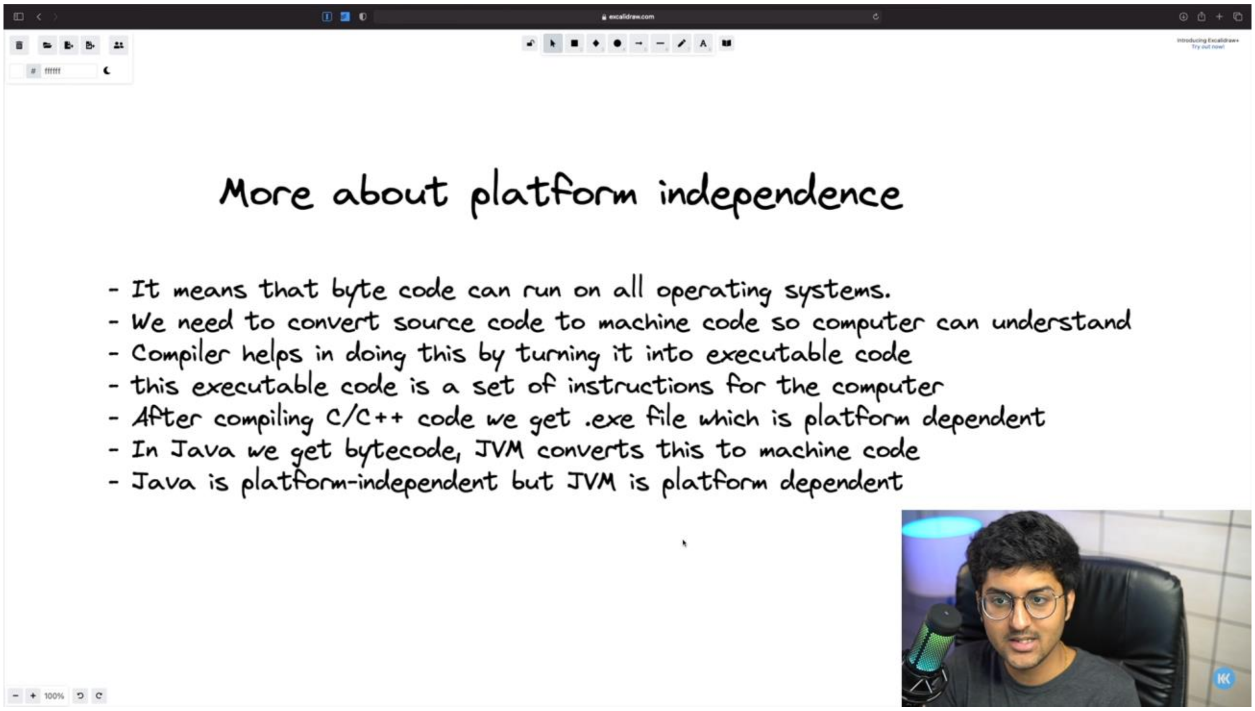
Introduction to Java



How Java code executes

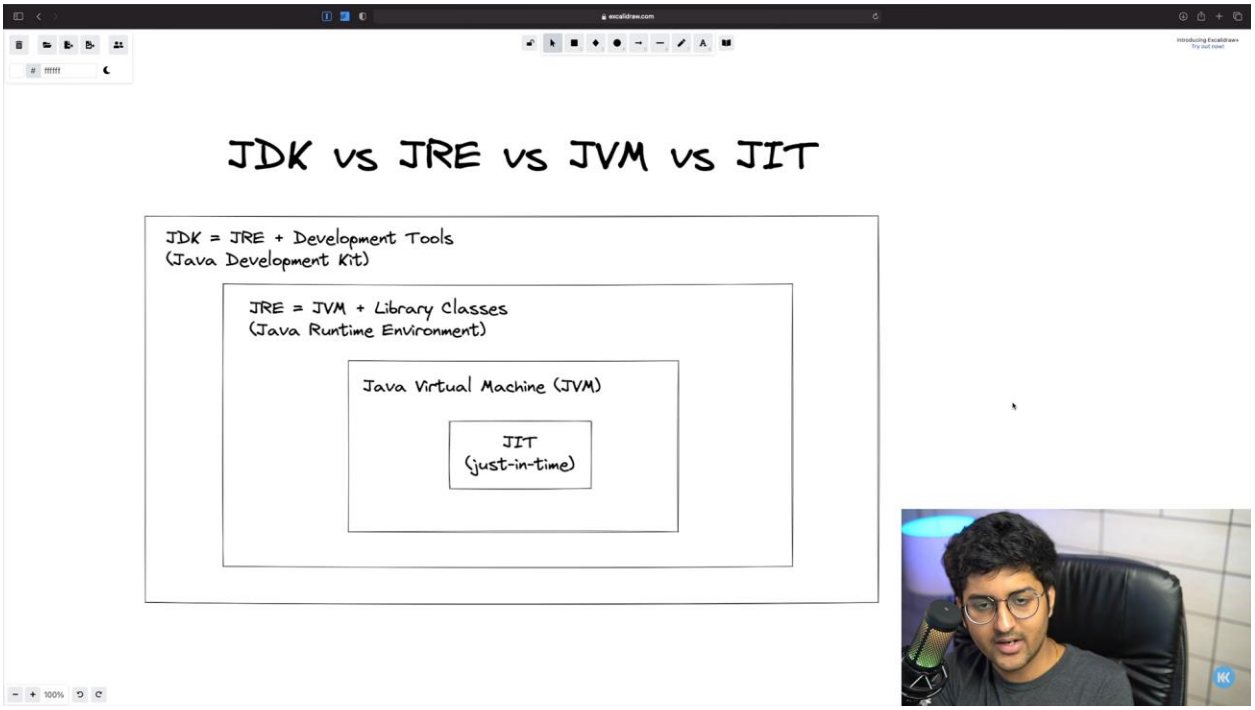
this is the source code

- this code will not directly run on a system
- we need JVM to run this
- Reason why Java is platform independent



More about platform independence

- It means that byte code can run on all operating systems.
- We need to convert source code to machine code so computer can understand
- Compiler helps in doing this by turning it into executable code
- this executable code is a set of instructions for the computer
- After compiling C/C++ code we get .exe file which is platform dependent
- In Java we get bytecode, JVM converts this to machine code
- Java is platform-independent but JVM is platform dependent



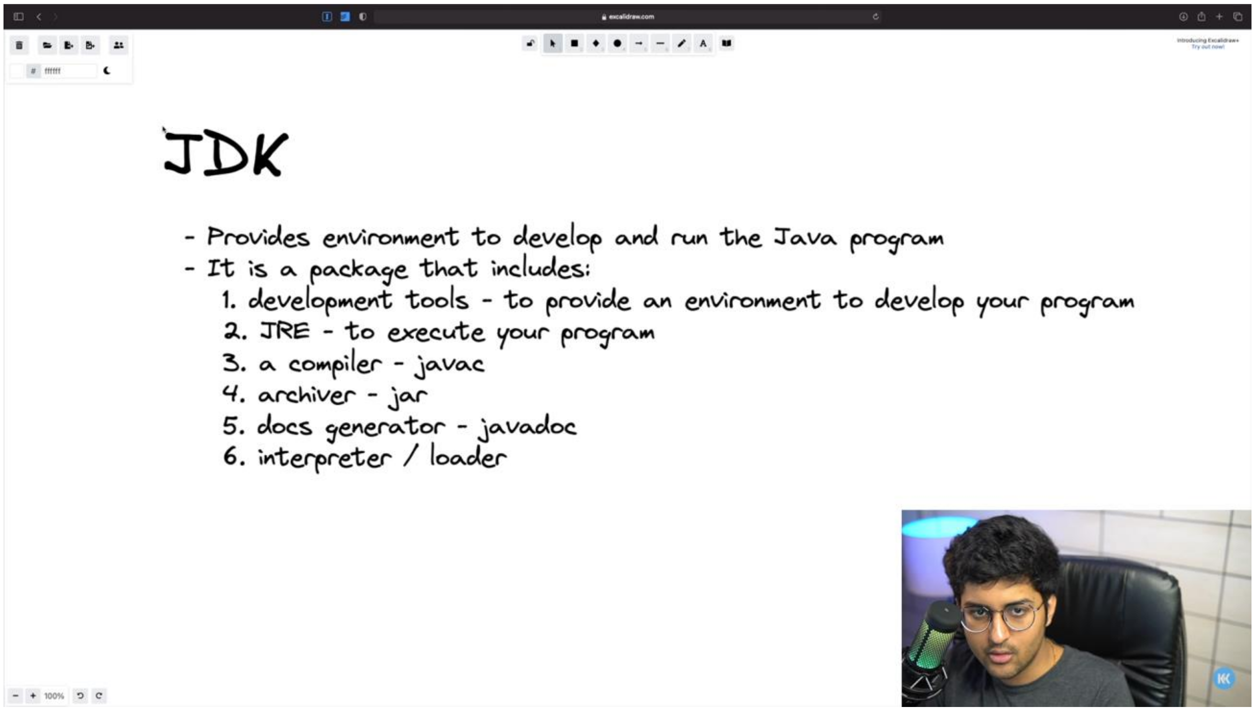
JDK vs JRE vs JVM vs JIT

JDK = JRE + Development Tools
(Java Development Kit)

JRE = JVM + Library Classes
(Java Runtime Environment)

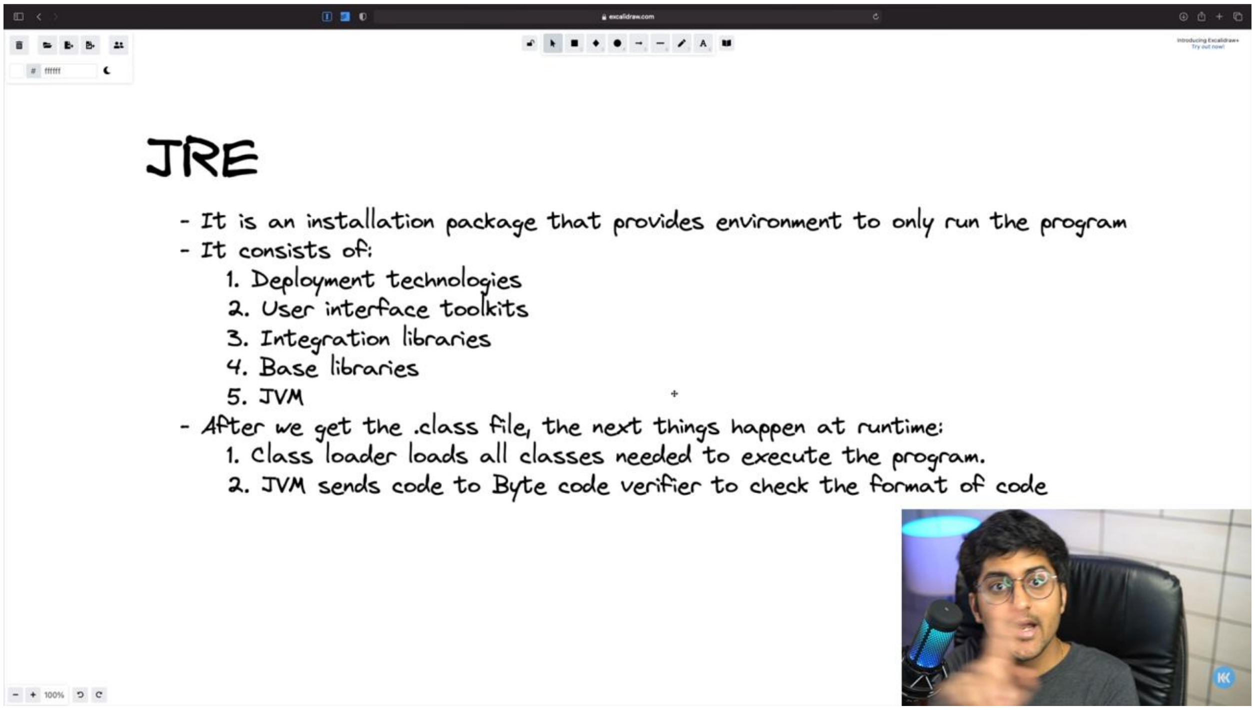
Java Virtual Machine (JVM)

JIT
(just-in-time)



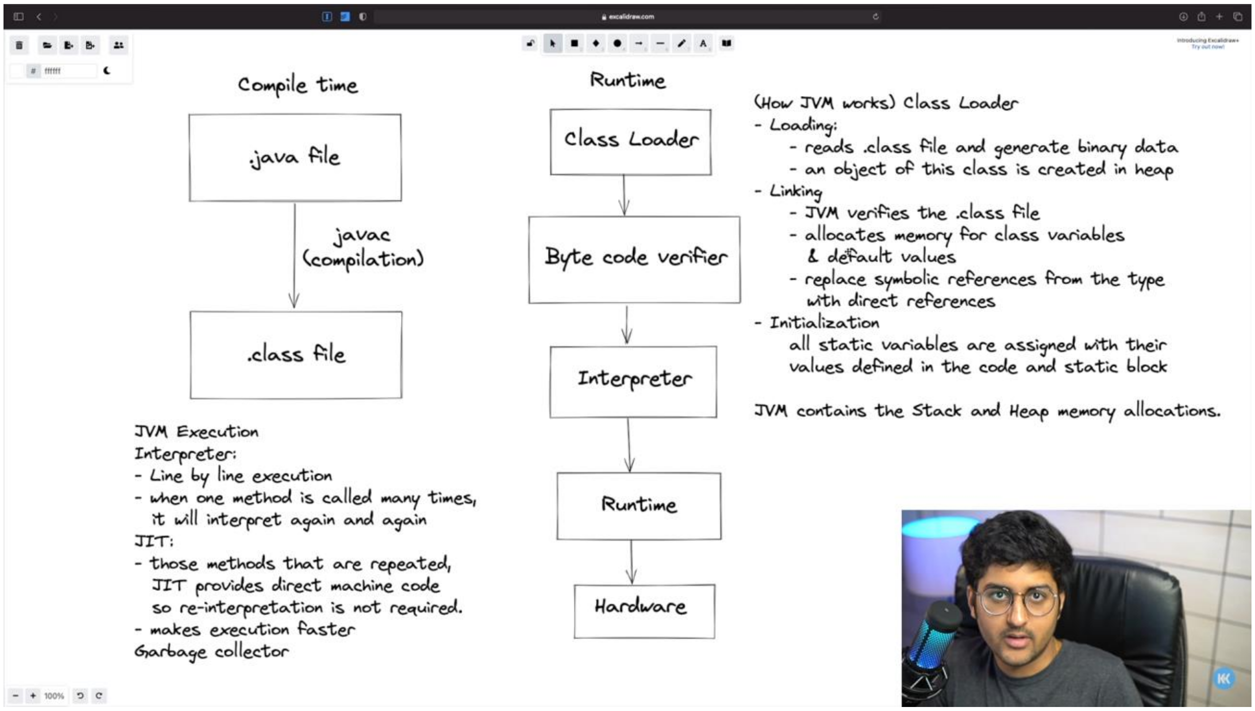
JDK

- Provides environment to develop and run the Java program
- It is a package that includes:
  1. development tools - to provide an environment to develop your program
  2. JRE - to execute your program
  3. a compiler - javac
  4. archiver - jar
  5. docs generator - javadoc
  6. interpreter / loader



JRE

- It is an installation package that provides environment to only run the program
- It consists of:
  1. Deployment technologies
  2. User interface toolkits
  3. Integration libraries
  4. Base libraries
  5. JVM
- After we get the .class file, the next things happen at runtime:
  1. Class loader loads all classes needed to execute the program.
  2. JVM sends code to Byte code verifier to check the format of code



Compile time

java file

javac
(compilation)

.class file

Runtime

Class Loader

Byte code verifier

Interpreter

Runtime

Hardware

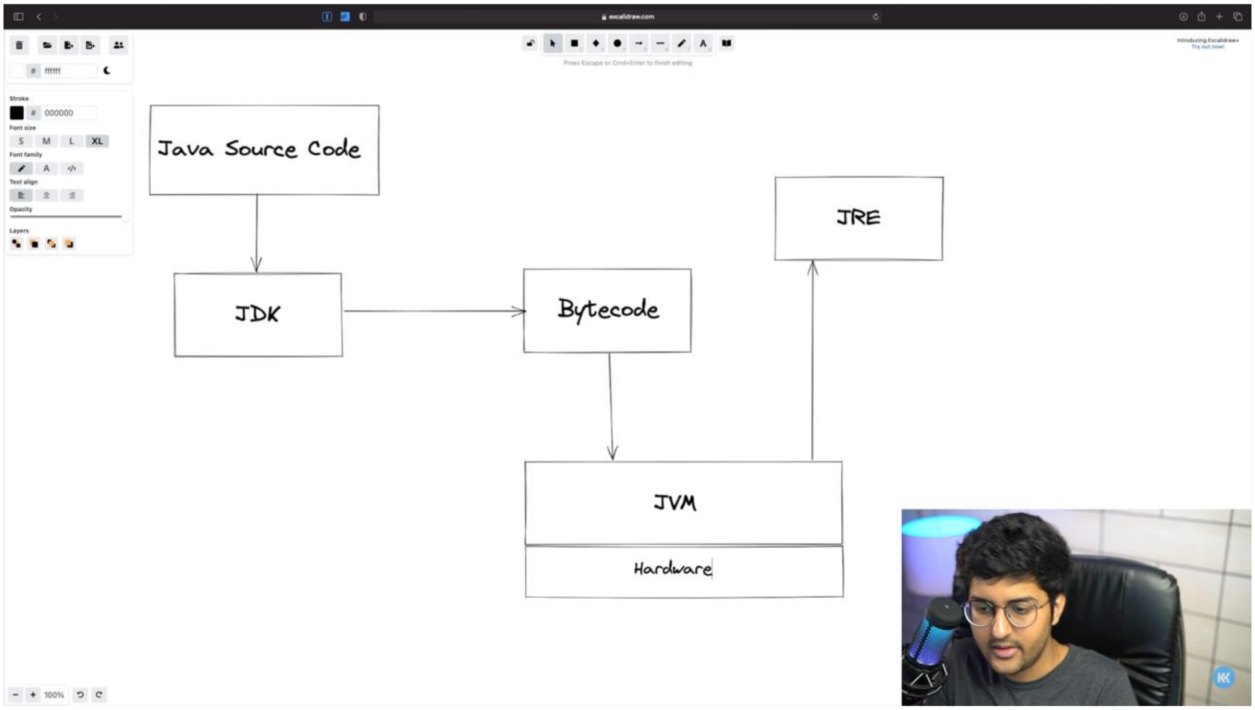
JVM Execution
Interpreter:
- Line by line execution
- when one method is called many times, it will interpret again and again

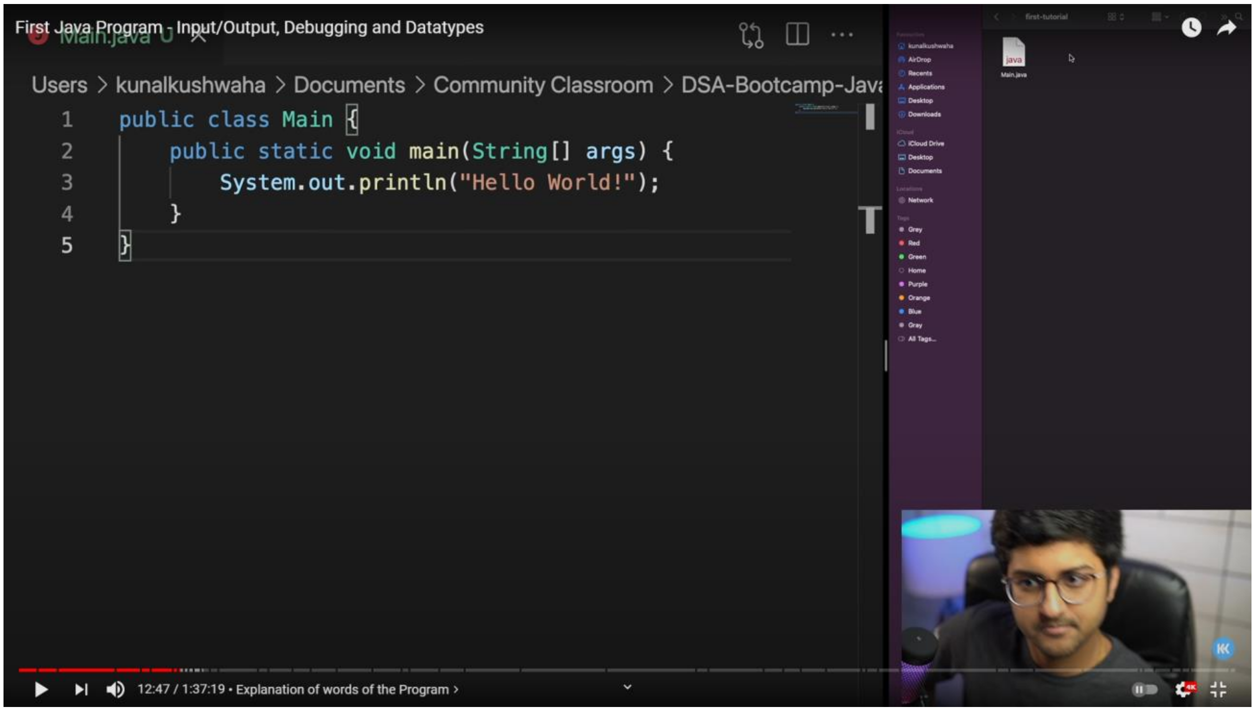
JIT:
- those methods that are repeated, JIT provides direct machine code so re-interpretation is not required.
- makes execution faster

Garbage collector

(How JVM works) Class Loader
- Loading:
  - reads .class file and generate binary data
  - an object of this class is created in heap
- Linking
  - JVM verifies the .class file
  - allocates memory for class variables & default values
  - replace symbolic references from the type with direct references
- Initialization
  all static variables are assigned with their values defined in the code and static block

JVM contains the Stack and Heap memory allocations.



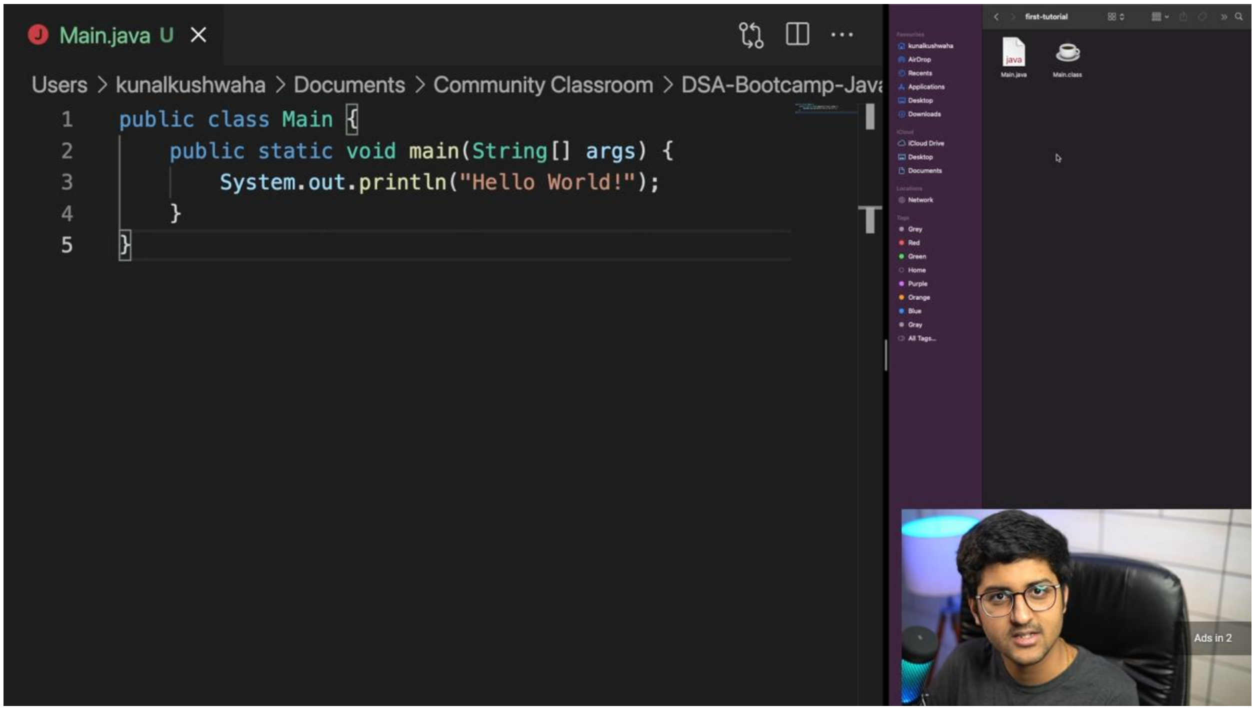

First Java Program - Input/Output, Debugging and Datatypes

Main.java U

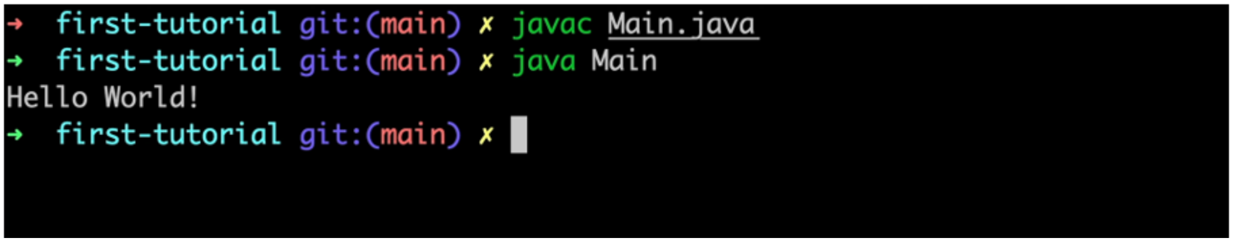
Users > kunalkushwaha > Documents > Community Classroom > DSA-Bootcamp-Java

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

12:47 / 1:37:19 • Explanation of words of the Program >

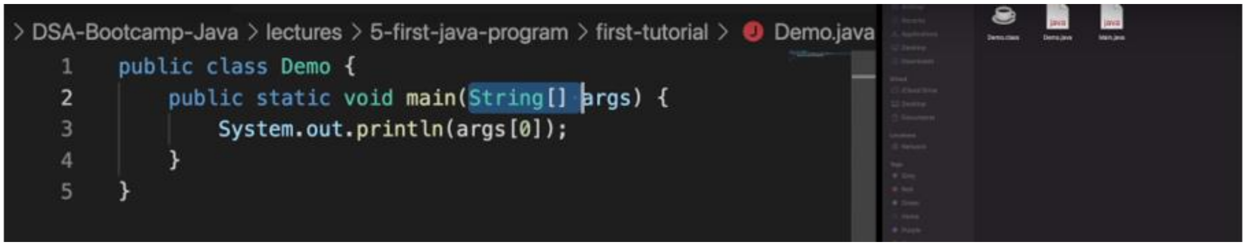


```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



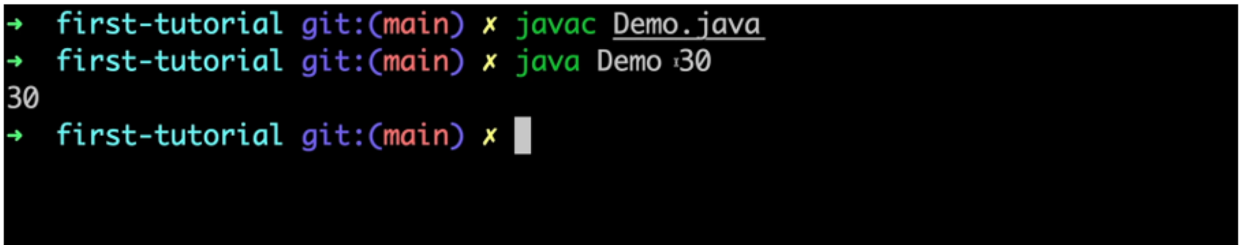
```
➜  first-tutorial git:(main) ✗ javac Main.java
➜  first-tutorial git:(main) ✗ java Main
Hello World!
➜  first-tutorial git:(main) ✗ 
```

Argument at index two



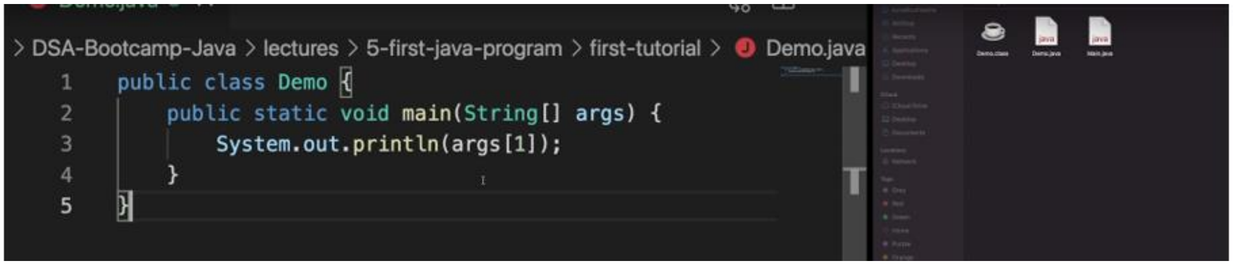
> DSA-Bootcamp-Java > lectures > 5-first-java-program > first-tutorial > Demo.java

```java
public class Demo {
    public static void main(String[] args) {
        System.out.println(args[0]);
    }
}
```

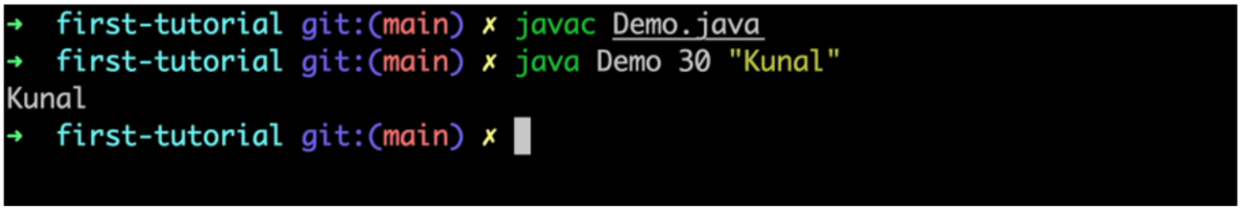


```
➜  first-tutorial git:(main) ✗ javac Demo.java
➜  first-tutorial git:(main) ✗ java Demo 30
30
➜  first-tutorial git:(main) ✗ 
```

Argument index one

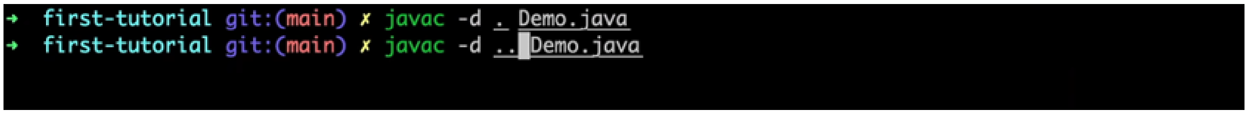

DSA-Bootcamp-Java > lectures > 5-first-java-program > first-tutorial > Demo.java
Demo.class
Demo.java
Main.java

```java
public class Demo {
    public static void main(String[] args) {
        System.out.println(args[1]);
    }
}
```



```
➜  first-tutorial git:(main) ✗ javac Demo.java
➜  first-tutorial git:(main) ✗ java Demo 30 "Kunal"
Kunal
➜  first-tutorial git:(main) ✗ 
```

Changing Location of Bytecode



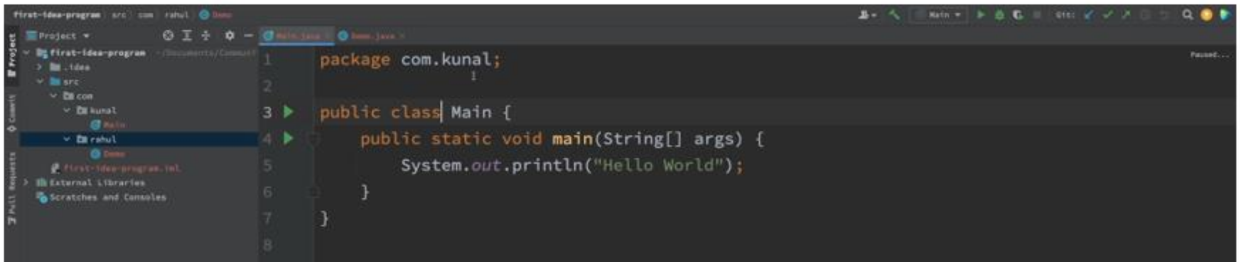
```
➜  first-tutorial git:(main) ✗ javac -d . Demo.java
➜  first-tutorial git:(main) ✗ javac -d .. Demo.java
```

-d means choosing which directory

. means present directory and .. means previous directory

What is package?

A **package** in **Java** is used to group related classes. Think of it as a folder in a file directory. We use **packages** to avoid name conflicts, and to write a better



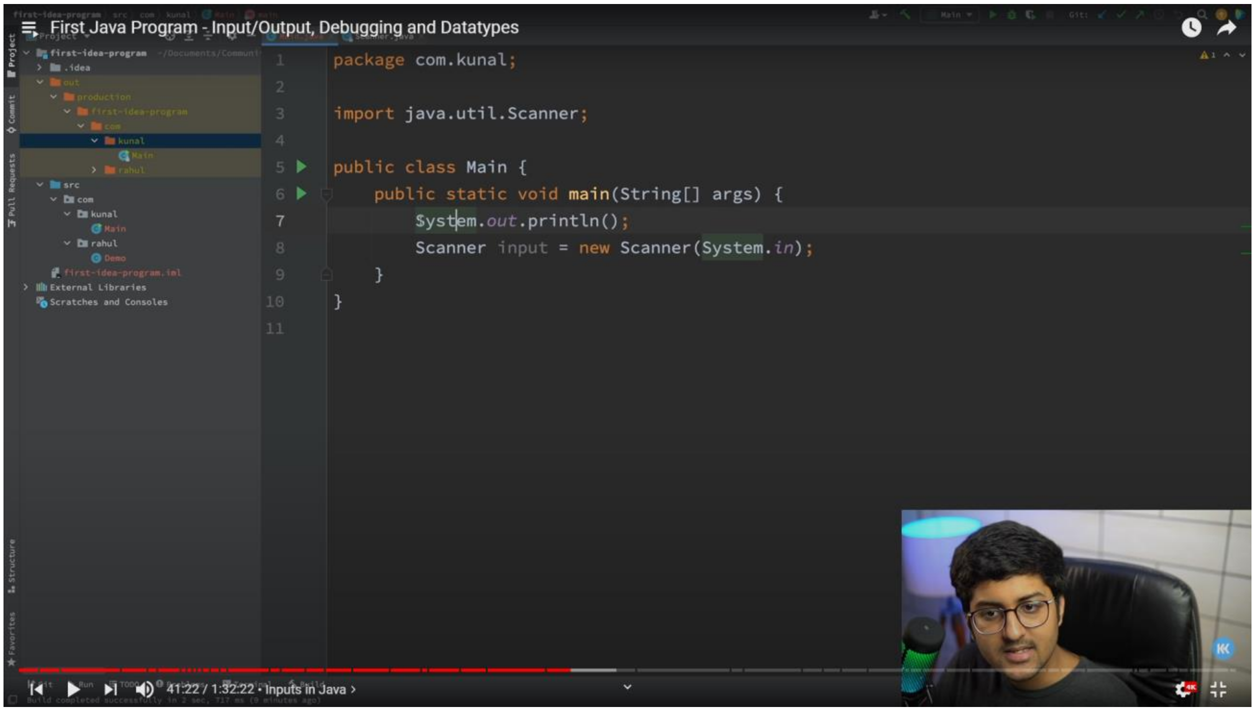
```java
package com.kunal;

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

System is a class defined by the java developer. Out is a variable declared for PrintStream which has println method in it.

Inputs in Java

import java.util.Scanner;

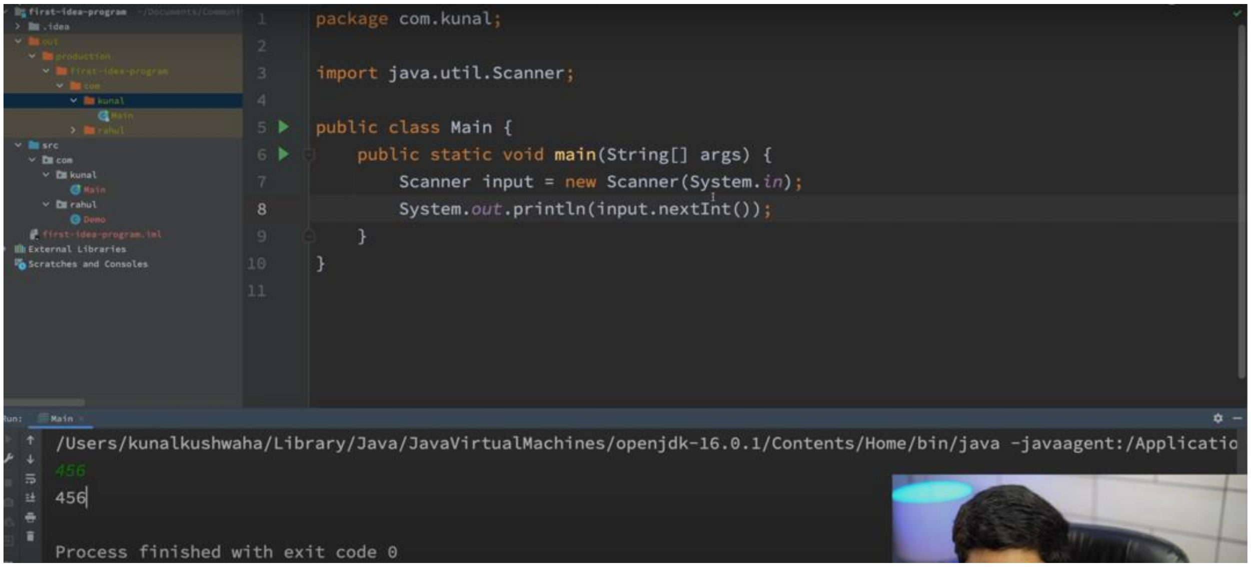
java.util is a package & Scanner is a class new keyword creates object


First Java Program - Input/Output, Debugging and Datatypes
41:22 / 1:32:22 • Inputs in Java ›

```java
package com.kunal;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println();
        Scanner input = new Scanner(System.in);
    }
}
```

System.in => (.in) means default standard input stream (i.e Keyboard)

System.out => (.out) means default standard output stream (i.e comandLine)

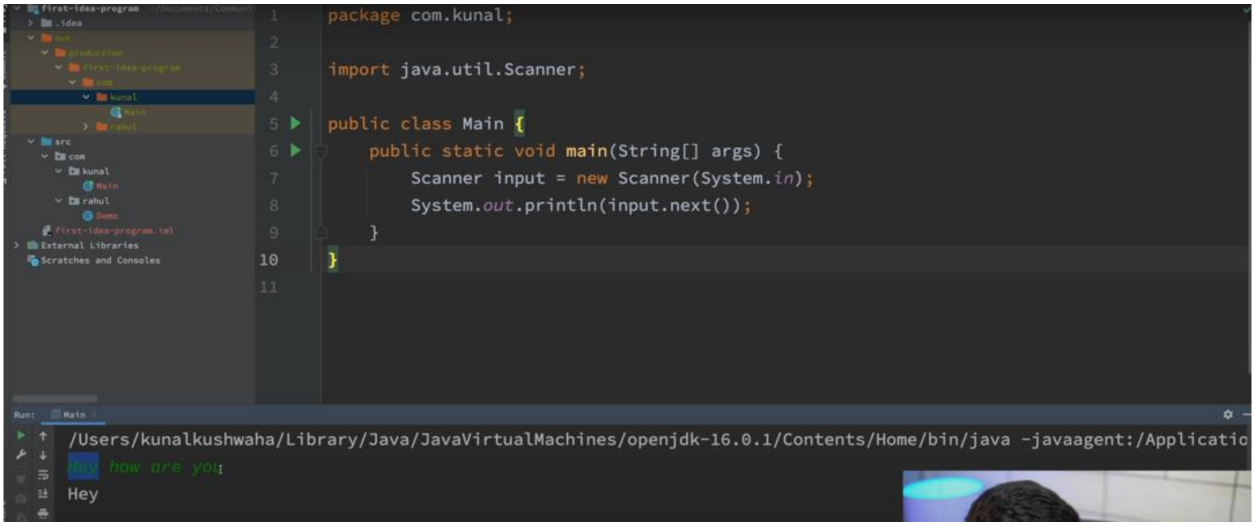


```java
package com.kunal;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println(input.nextInt());
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
456
456

Process finished with exit code 0
```

nextInt() is also in scanner class.

next()



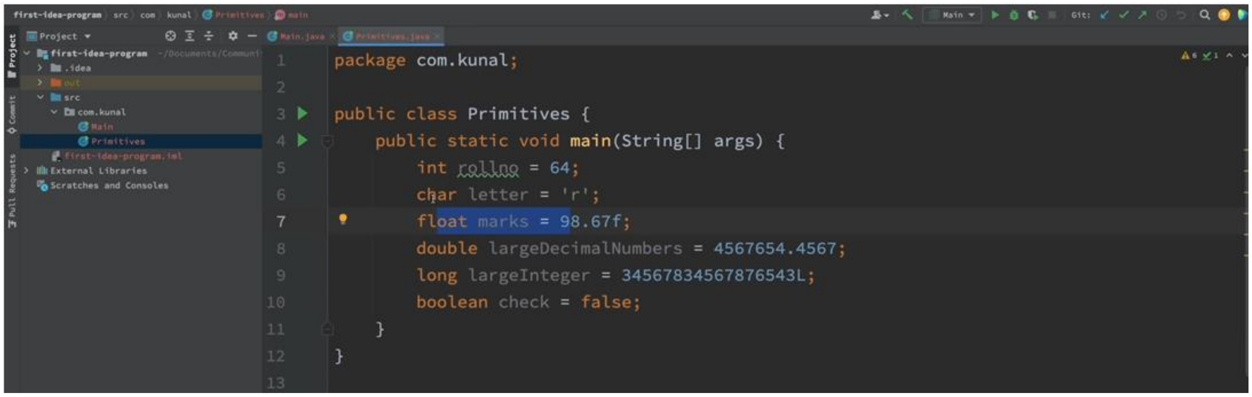
```
package com.kunal;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println(input.next());
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
Hey how are you
Hey
```

nextLine() will take entire line

Data type

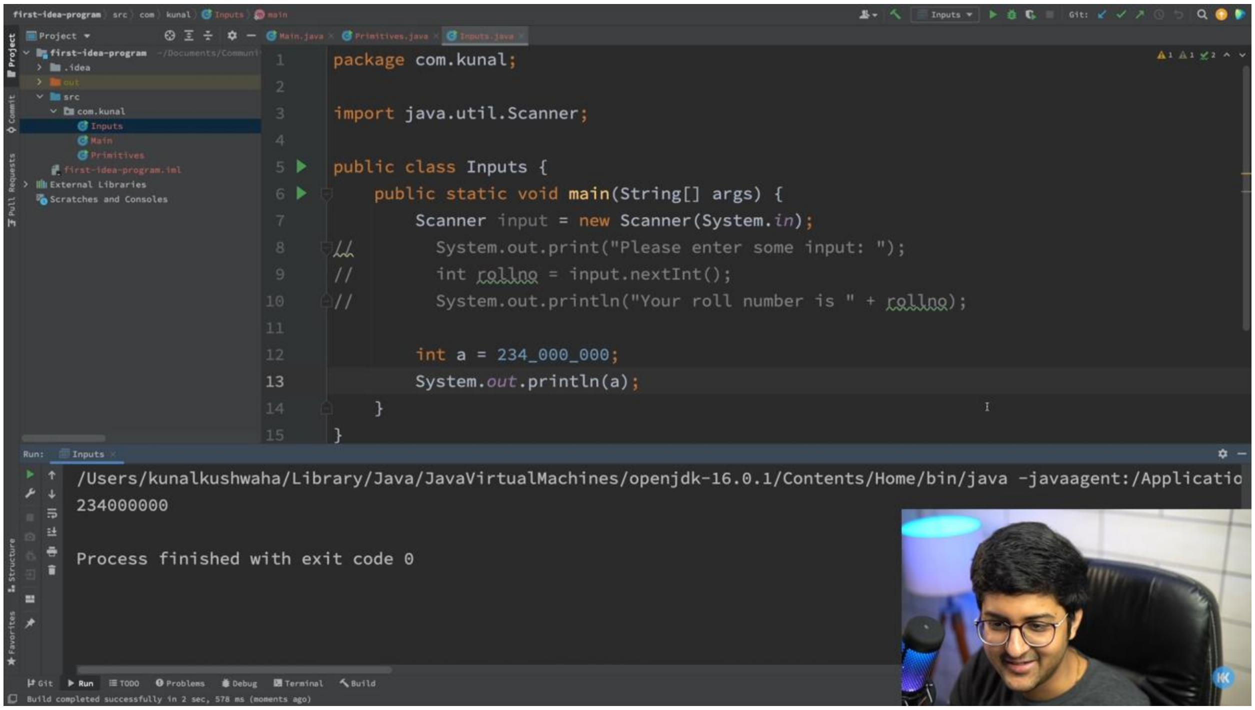
Primitive data type



```
package com.kunal;

public class Primitives {
    public static void main(String[] args) {
        int rollno = 64;
        char letter = 'r';
        float marks = 98.67f;
        double largeDecimalNumbers = 4567654.4567;
        long largeInteger = 34567834567876543L;
        boolean check = false;
    }
}
```

Int and double are default type so to use long and float we need to append l and f at the end of the variable value.

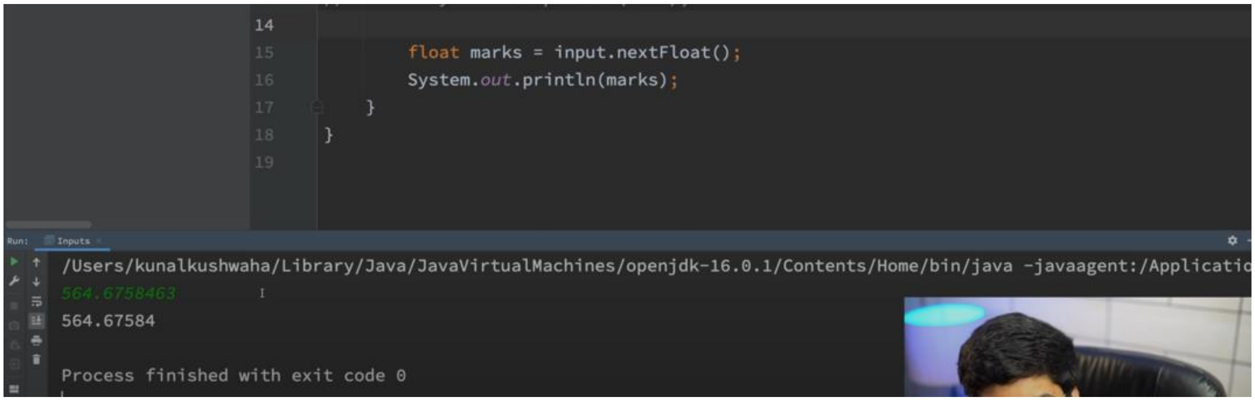
$ note to comment select the text the click ctrl+ /



```java
package com.kunal;

import java.util.Scanner;

public class Inputs {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
//          System.out.print("Please enter some input: ");
//          int rollno = input.nextInt();
//          System.out.println("Your roll number is " + rollno);

        int a = 234_000_000;
        System.out.println(a);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
234000000

Process finished with exit code 0
```

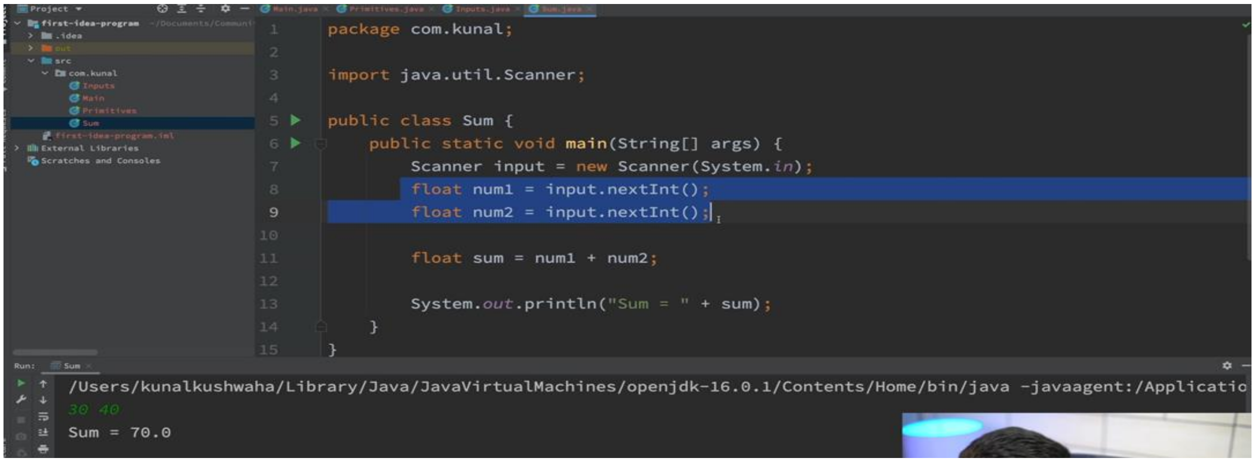
Build completed successfully in 2 sec, 578 ms (moments ago)



```java

        float marks = input.nextFloat();
        System.out.println(marks);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
564.6758463
564.67584

Process finished with exit code 0
```

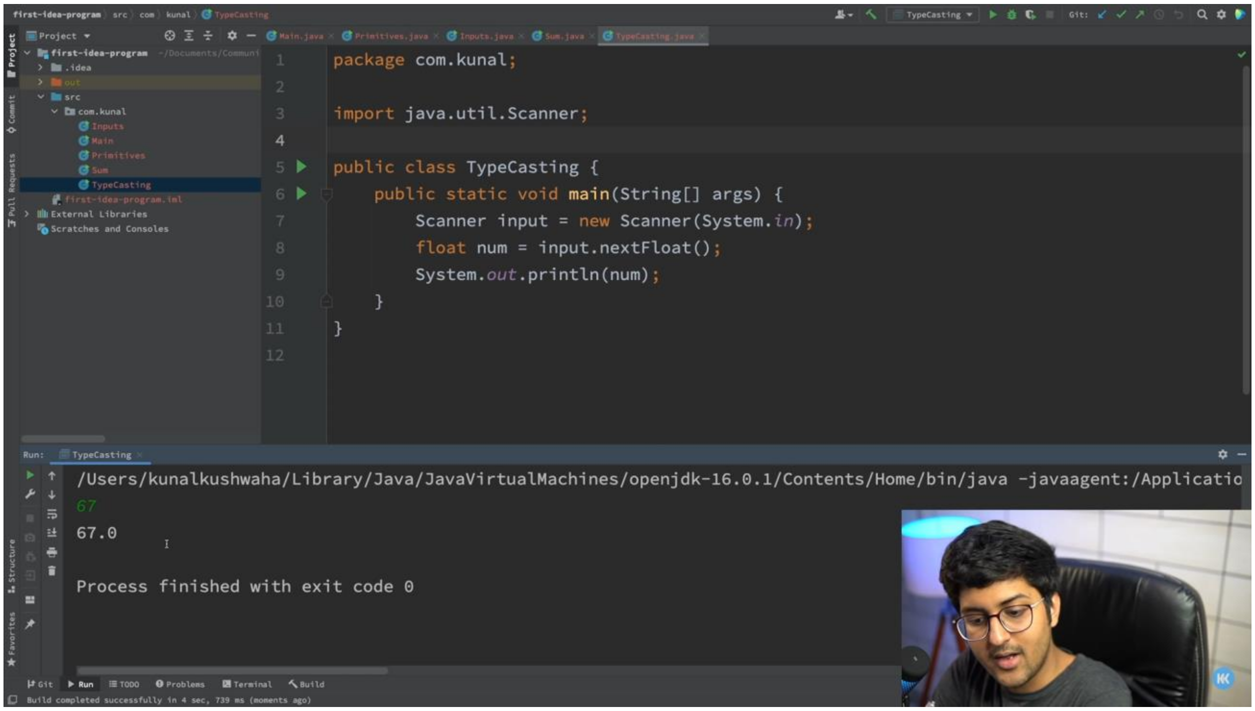
Sum of two numbers



```java
package com.kunal;

import java.util.Scanner;

public class Sum {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        float num1 = input.nextInt();
        float num2 = input.nextInt();

        float sum = num1 + num2;

        System.out.println("Sum = " + sum);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
30 40
Sum = 70.0
```

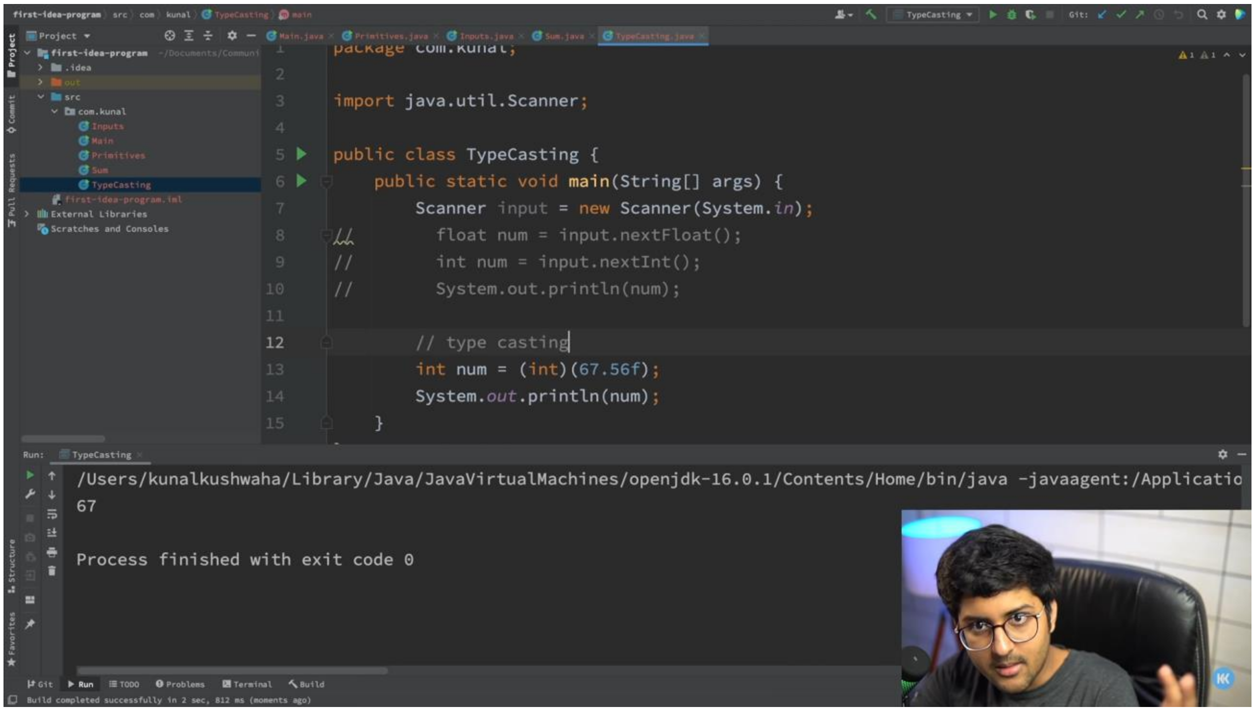
Type Conversation and Type casting



```java
package com.kunal;

import java.util.Scanner;

public class TypeCasting {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        float num = input.nextFloat();
        System.out.println(num);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
67
67.0

Process finished with exit code 0
```

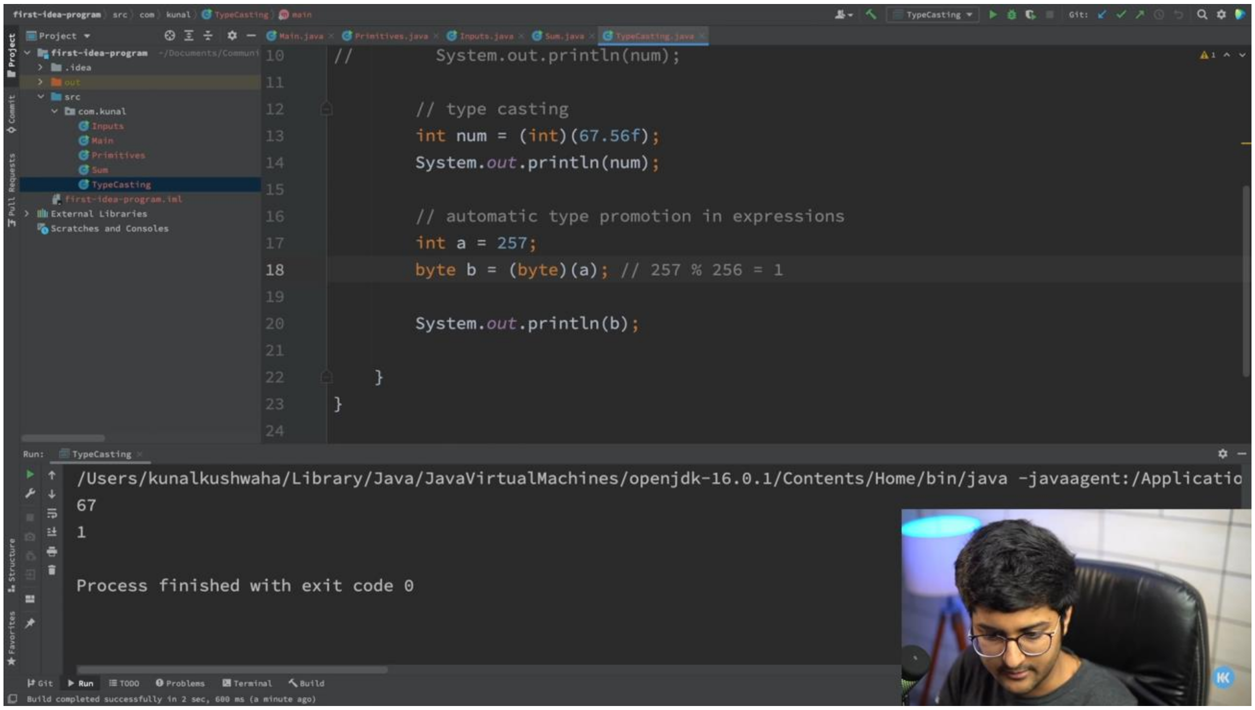
Int is converted in to float automatically



```java
package com.kunal;

import java.util.Scanner;

public class TypeCasting {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
//        float num = input.nextFloat();
//        int num = input.nextInt();
//        System.out.println(num);

        // type casting
        int num = (int)(67.56f);
        System.out.println(num);
    }
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
67

Process finished with exit code 0
```

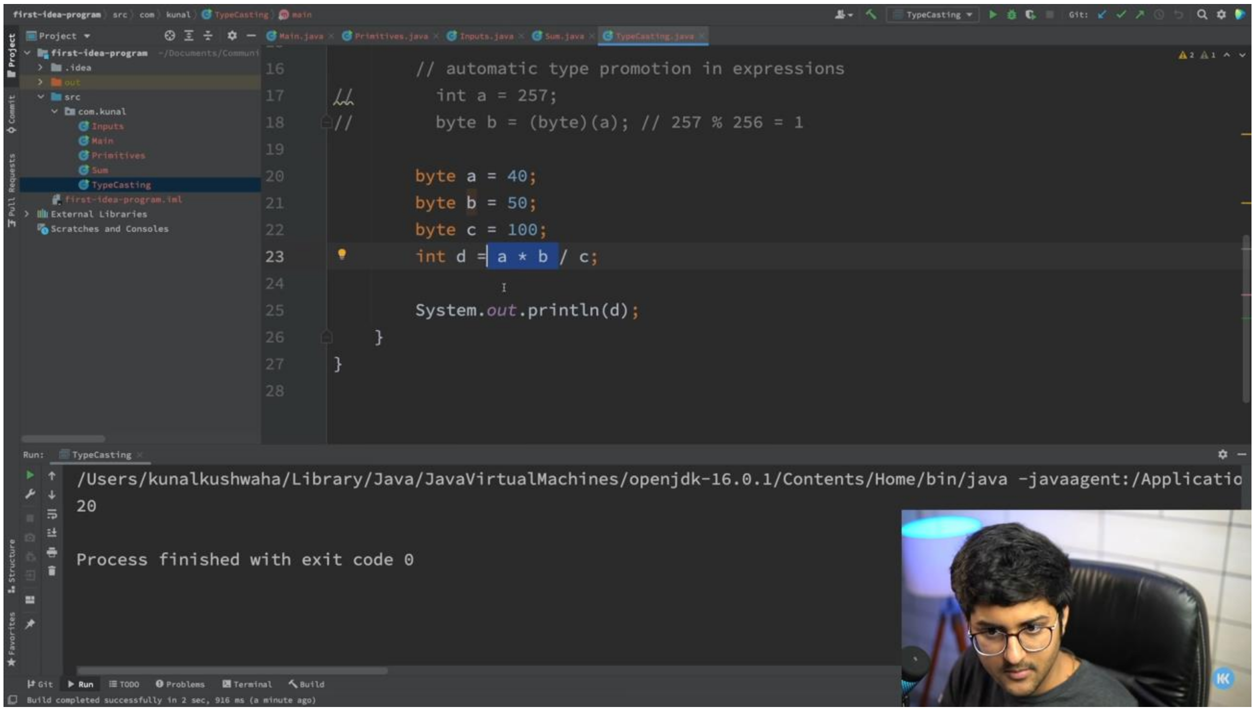
Automatic type promotion in expression



```
//        System.out.println(num);

        // type casting
        int num = (int)(67.56f);
        System.out.println(num);

        // automatic type promotion in expressions
        int a = 257;
        byte b = (byte)(a); // 257 % 256 = 1

        System.out.println(b);

    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
67
1

Process finished with exit code 0
```

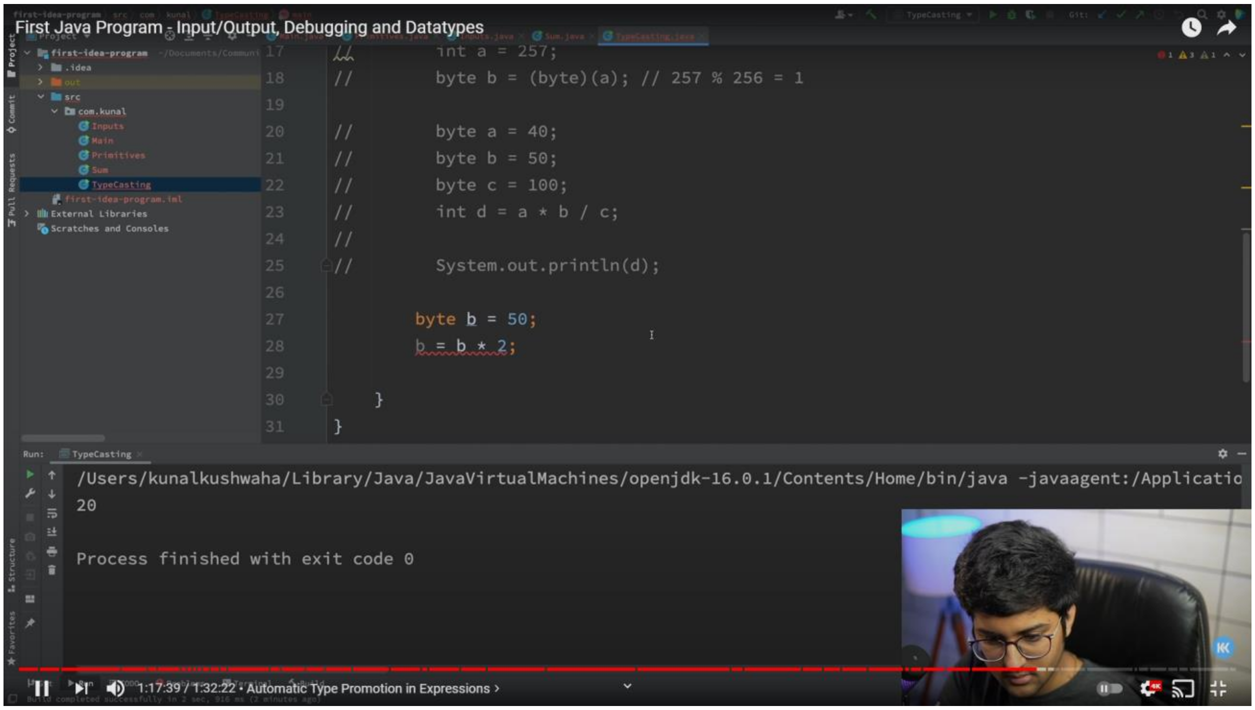
Because byte can store only up to 256 values



```
        // automatic type promotion in expressions
//        int a = 257;
//        byte b = (byte)(a); // 257 % 256 = 1

        byte a = 40;
        byte b = 50;
        byte c = 100;
        int d = a * b / c;

        System.out.println(d);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
20

Process finished with exit code 0
```

While solving expression byte is converted in to int and then expression is calculated.

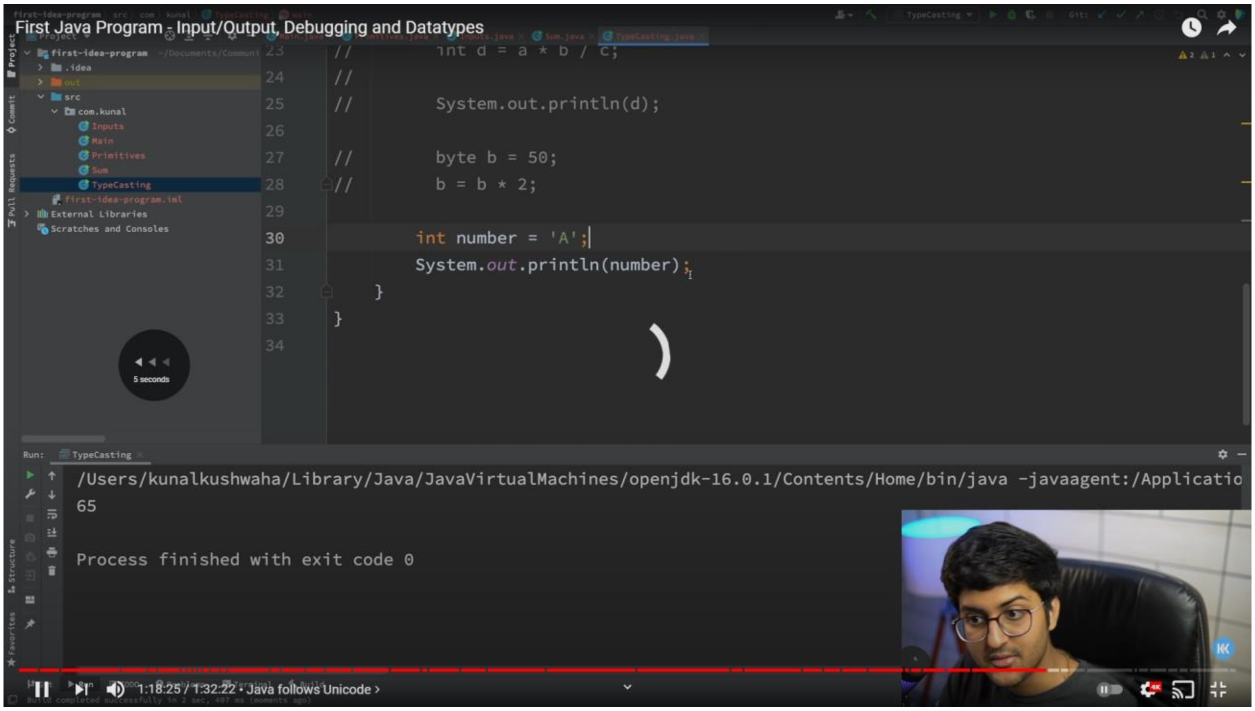


```
//        int a = 257;
//        byte b = (byte)(a); // 257 % 256 = 1

//        byte a = 40;
//        byte b = 50;
//        byte c = 100;
//        int d = a * b / c;
//
//        System.out.println(d);

        byte b = 50;
        b = b * 2;

    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
20

Process finished with exit code 0
```

Here b*2 can’t be calculated as it is assigned to byte value and it can’t be converted.

Automatic Type promotion



```
//        int d = a * b / c;
//
//        System.out.println(d);

//        byte b = 50;
//        b = b * 2;

        int number = 'A';
        System.out.println(number);
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
65

Process finished with exit code 0
```

For example, examine the following expression:

```
public class Main {
  public static void main(String[] argv) {
    byte a = 40;// j  a v  a  2s .c o m
    byte b = 50;
    byte c = 100;
    int d = a * b / c;
  }
}
```

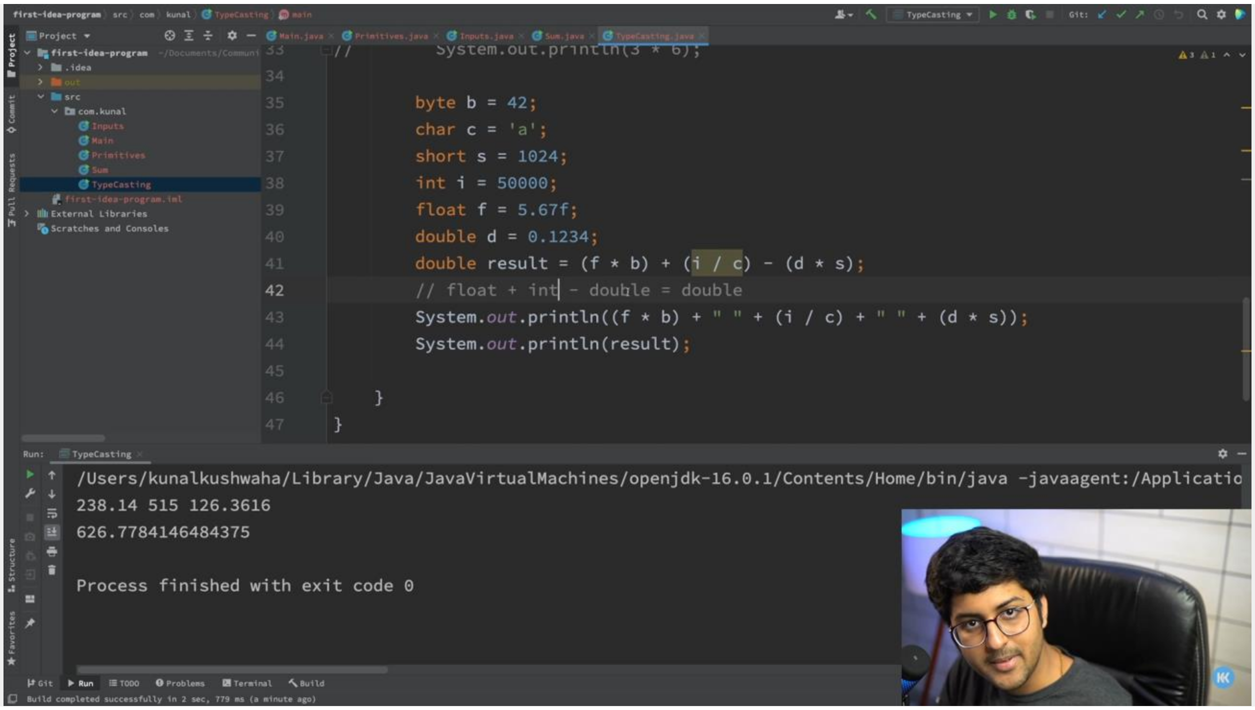
The result of `a * b` exceeds the range of `byte`. To handle this kind of problem, Java automatically promotes each byte or short operand to `int`. `a * b` is performed using integers.

Here are the Type Promotion Rules:

1. All `byte` and `short` values are promoted to `int`.
2. If one operand is a `long`, the whole expression is promoted to `long`.
3. If one operand is a `float`, the entire expression is promoted to `float`.
4. If any of the operands is `double`, the result is `double`.

In the following code, `f * b`, `b` is promoted to a `float` and the result of the subexpression is `float`.

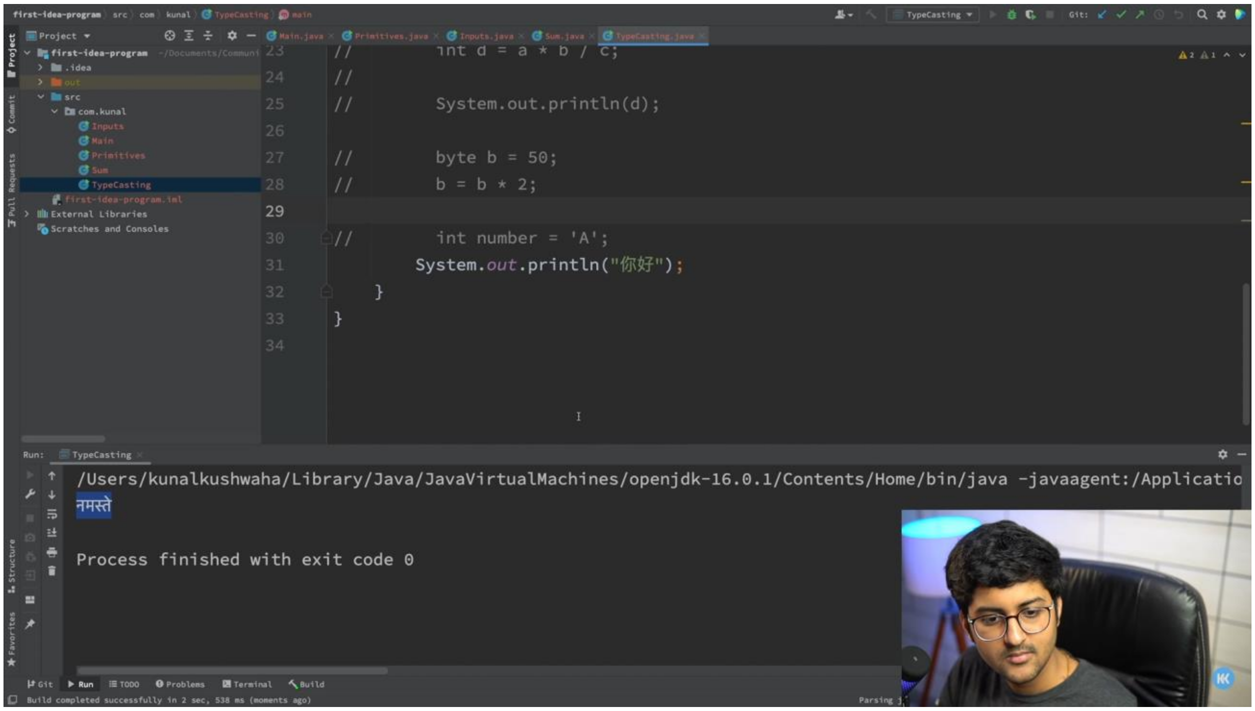
```
public class Main {
/* j a va2s .  co m*/
  public static void main(String args[]) {
    byte b = 4;
    float f = 5.5f;
    float result = (f * b);
    System.out.println("f * b = " + result);
  }
}
```



```
//        System.out.println(3 * 6);

        byte b = 42;
        char c = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = 0.1234;
        double result = (f * b) + (i / c) - (d * s);
        // float + int - double = double
        System.out.println((f * b) + " " + (i / c) + " " + (d * s));
        System.out.println(result);

    }
}
```

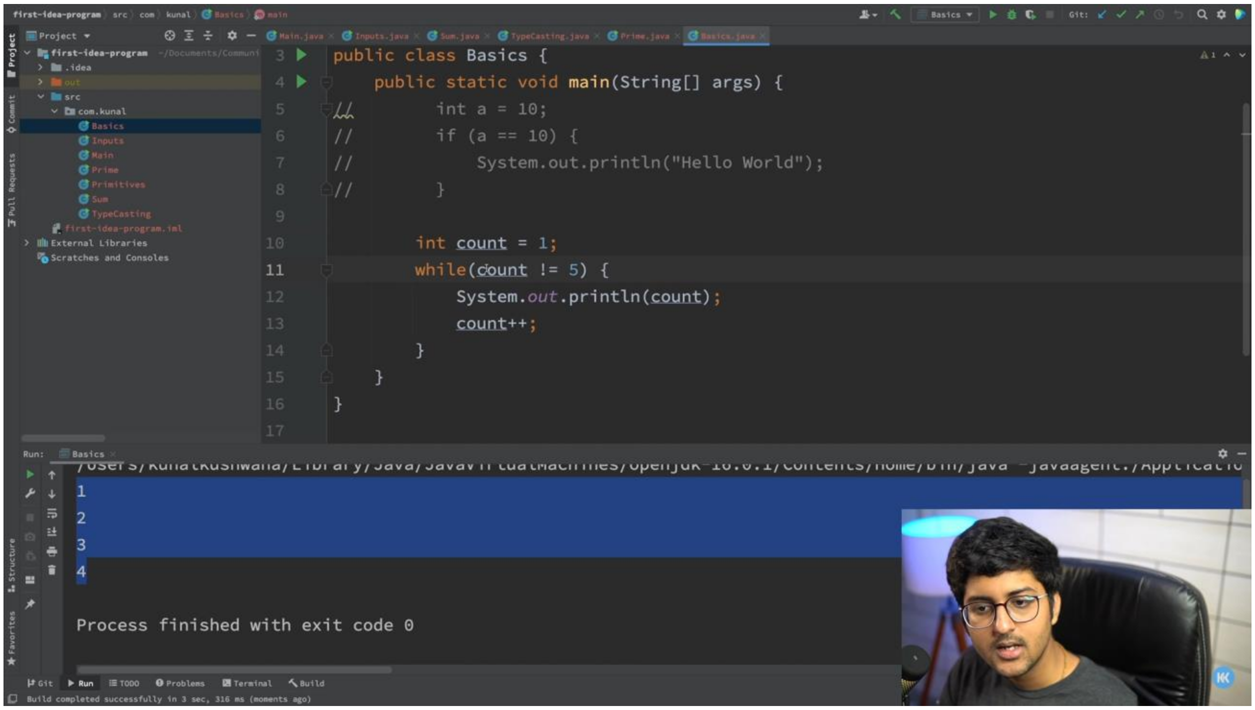
```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
238.14 515 126.3616
626.7784146484375

Process finished with exit code 0
```

Java follows unicode



```
//        int d = a * b / c;
//
//        System.out.println(d);

//        byte b = 50;
//        b = b * 2;

//        int number = 'A';
        System.out.println("你好");
    }
}
```

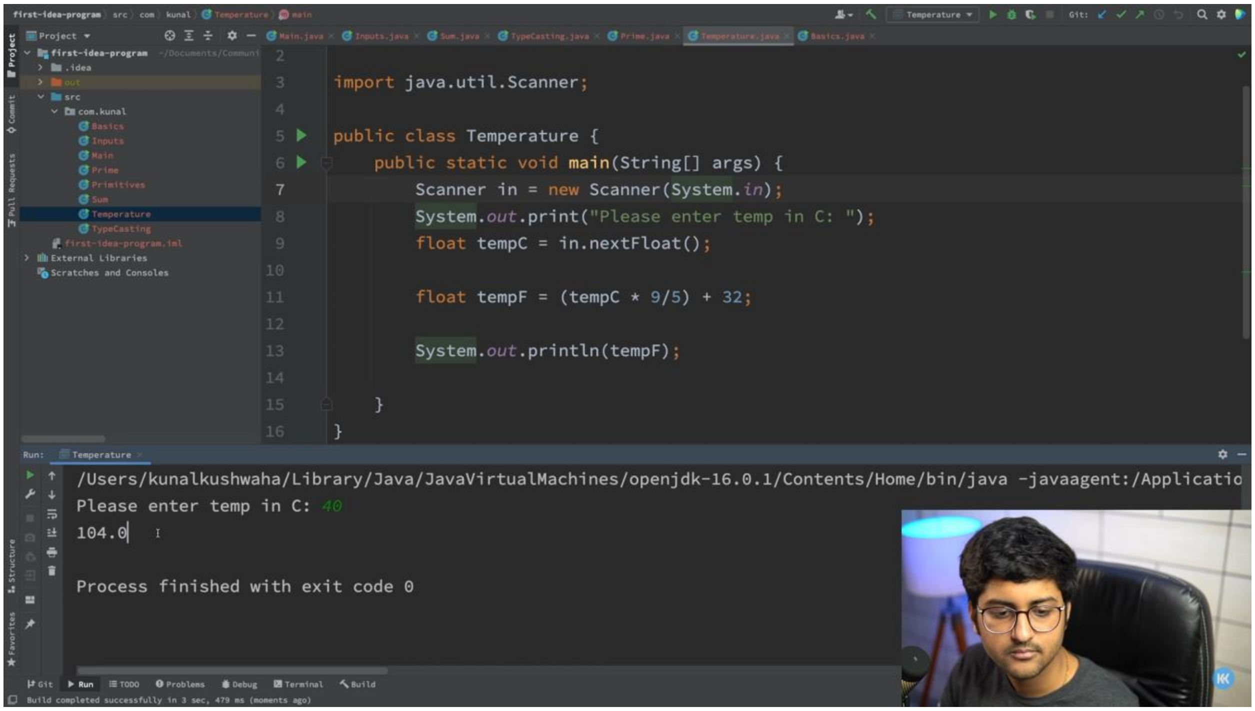
```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
नमस्ते

Process finished with exit code 0
```



```java
public class Basics {
    public static void main(String[] args) {
//        int a = 10;
//        if (a == 10) {
//            System.out.println("Hello World");
//        }

        int count = 1;
        while(count != 5) {
            System.out.println(count);
            count++;
        }
    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
1
2
3
4

Process finished with exit code 0
```

Build completed successfully in 3 sec, 316 ms (moments ago)



```java

import java.util.Scanner;

public class Temperature {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Please enter temp in C: ");
        float tempC = in.nextFloat();

        float tempF = (tempC * 9/5) + 32;

        System.out.println(tempF);

    }
}
```

```
/Users/kunalkushwaha/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -javaagent:/Applicatio
Please enter temp in C: 40
104.0

Process finished with exit code 0
```

Build completed successfully in 3 sec, 479 ms (moments ago)