Doubt Class with Lakshay

Special class

Love Babbar • Mar 2, 2024

Arrays in Java

- Love Babbar

What is an Array ?

In Java, an array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created, and it cannot be changed once created. Each item in an array is called an element, and each element is accessed by its numerical index.

Creation of Array:

- Declaration
- Allocation
- Initialisation

Accessing Elements

- Index is the answer

Common Exceptions

For-each Loop:

```
// print array elements

class Main {
  public static void main(String[] args) {

    // create an array
    int[] numbers = {3, 9, 5, -5};

    // for each loop
    for (int number: numbers) {
      System.out.println(number);
    }
  }
}
```

Let's Practice

- Find Sum of all values in array
- Find Multiplication of all values in array
- Find minimum value in array
- Find maximum value in array

Multi-dimensional Array

In Java, a multi-dimensional array is essentially an array of arrays. The most common type of multi-dimensional array is the two-dimensional array, which can be thought of as a grid or table of rows and columns. However, Java supports arrays of any number of dimensions.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

It's an Array of arrays

Creation of 2D Arrays

- Definition
- Initialisation

Accessing Elements in 2D Arrays

- Again, Index is the answer

Let's Practice

Methods in Java

- Love Babbar

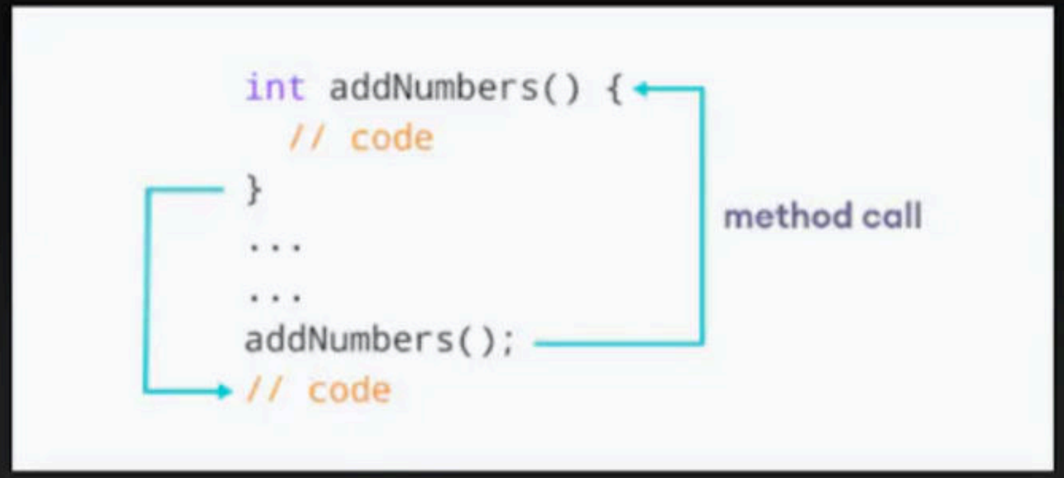
What is a Function / Method ?

A method is a block of code that performs a specific task.

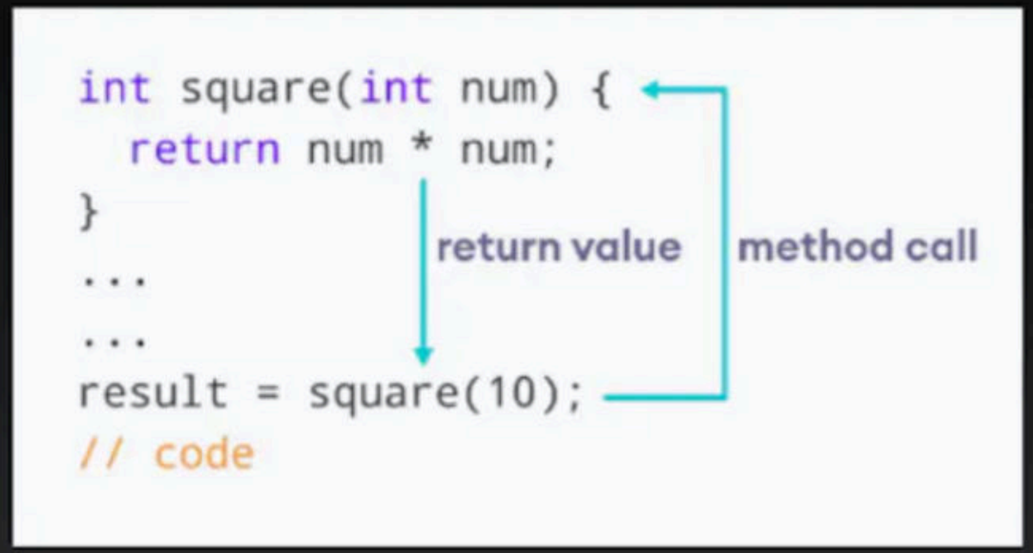
Function Declaration:

```
modifier static returnType nameOfMethod (parameter1, parameter2, ...) {
  // method body
}
```

Function Call:



Return Keyword:



Method Parameters:

```
// method with two parameters
int addNumbers(int a, int b) {
  // code
}

// method with no parameter
int addNumbers(){
  // code
}
```

Function Call Stack:

Let's Practice

- Write a Function to print your name "N" times
- Write a function to print the sum from 1 to N
- WAF to return the average of 2 numbers
- WAF to return the minimum of 2 numbers
- WAF to return the maximum of 2 numbers
- WAF to return the absolute value of a number
- WAF to return the exponent of a number
- WAF to return a random value between 1 to N

Function and Arrays

- WAF to print the array
- WAF to return the sum of all elements in the array
- WAF to double the values present inside the array

Strings in Java

- Love Babbar

What is a String ?

In Java, a String is a sequence of characters used to store and manipulate text. It is one of the most fundamental data types in Java and is part of the java.lang package, which is automatically imported into every Java program

Creation of String

- String Literal
- New Keyword

String Pool

Java maintains a special area in the Java heap called the string pool. This is where all string literals are stored. When you create a string using string literals, Java checks the pool first. If the string already exists, it returns the reference to the same instance, helping to save memory.

Immutable Strings

Strings in Java are immutable, which means once a String object is created, its value cannot be changed. If you perform any operations that seem to modify a String, what actually happens is a new String object is created.

Let's Practice

Comparing Strings in Java

- “==“ : The == operator in Java checks for reference equality, meaning it determines whether two string references point to the same object in memory, not comparing their contents.
- .equals( ): The .equals() method compares the content of two strings, returning true if the sequence of characters in both strings is the same.
- .equalsIgnoreCase( ): The .equalsIgnoreCase() method is similar to .equals() but ignores case differences, meaning it checks if two strings are equal irrespective of the case (uppercase or lowercase) of the characters.

String Input in Java

- nextLine( )
- next( )

Java String Methods:

- .length(): Returns the length of the string.
- .charAt(int index): Returns the character at the specified index.
- .substring(int beginIndex, int endIndex): Returns a substring from the specified beginIndex to endIndex.
- .contains(CharSequence s): Returns true if the sequence of char values is found in this string.
- .equals(Object another): Compares this string to the specified object.
- .equalsIgnoreCase(String anotherString): Compares this String to another String, ignoring case considerations.
- .toUpperCase(): Converts all the characters in this String to uppercase.
- .toLowerCase(): Converts all the characters in this String to lowercase.
- .trim(): Returns a copy of the string, with leading and trailing whitespace omitted.
- .split(String regex): Splits this string around matches of the given regular expression.

Java String Methods:

- .startsWith(String prefix): Checks if the string starts with the specified prefix.
- .endsWith(String suffix): Determines if the string ends with the specified suffix.
- .valueOf(any type): Converts different data types (int, long, float, double, boolean, char, char array, Object) into a string representation.
- .replace(char oldChar, char newChar): Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
- .toCharArray(): Converts the string to a new character array.
- .isEmpty(): Checks if the string is empty (length is 0).
- .isBlank(): Checks if the string is blank (empty or contains only white space characters).

Let's Practice

Strings and Functions:

Let's Practice

Java Exception Handling

- Love Babbar

What is Exception ?

In Java, Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program's instructions. Exceptions can be caught and handled by the program.

Exception occurs due to:

- Invalid user input
- Device failure
- Loss of network connection
- Physical limitations (out-of-disk memory)
- Code errors
- Opening an unavailable file

Runtime Exceptions

It happens due to programming errors aka unchecked exceptions

- RuntimeException is the superclass of all classes that exceptions are thrown during the normal operation of the Java VM (Virtual Machine). The RuntimeException and its subclasses are unchecked exceptions. The most common exceptions are NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException, InvalidArgumentException etc.

Java IOException exceptions

- The Java IOException is a checked exception that must be handled at compilation time.
- IOException is the base class for such exceptions which are thrown while accessing data from files, directories and streams. It represents a class of exceptions that are thrown when an I/O error occurs.

Try-catch block

- The ***try*** statement allows you to define a block of code to be tested for errors while it is being executed.
- The ***catch*** statement allows you to define a block of code to be executed, if an error occurs in the try block.

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

Try-catch-finally block

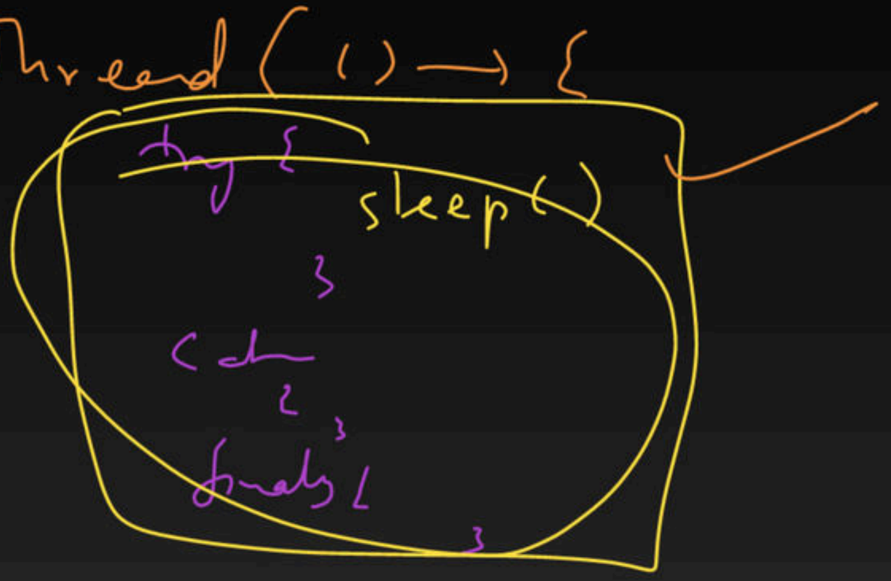
The try, catch, and finally blocks are important components of the Java programming language. They are used to handle errors and exceptions in Java. The three blocks are used in the following way:

- The *try* block is used to specify a block of code that may throw an exception.
- The *catch* block is used to handle the exception if it is thrown.
- The *finally* block is used to execute the code after the try and catch blocks have been executed.

```java
public class Main {
    public static void main(String[] args) {
        try {
            // Code that may throw an exception
        } catch (Exception e) {
            // Code to handle the exception
        } finally {
            // Code that is always executed
        }
    }
}
```

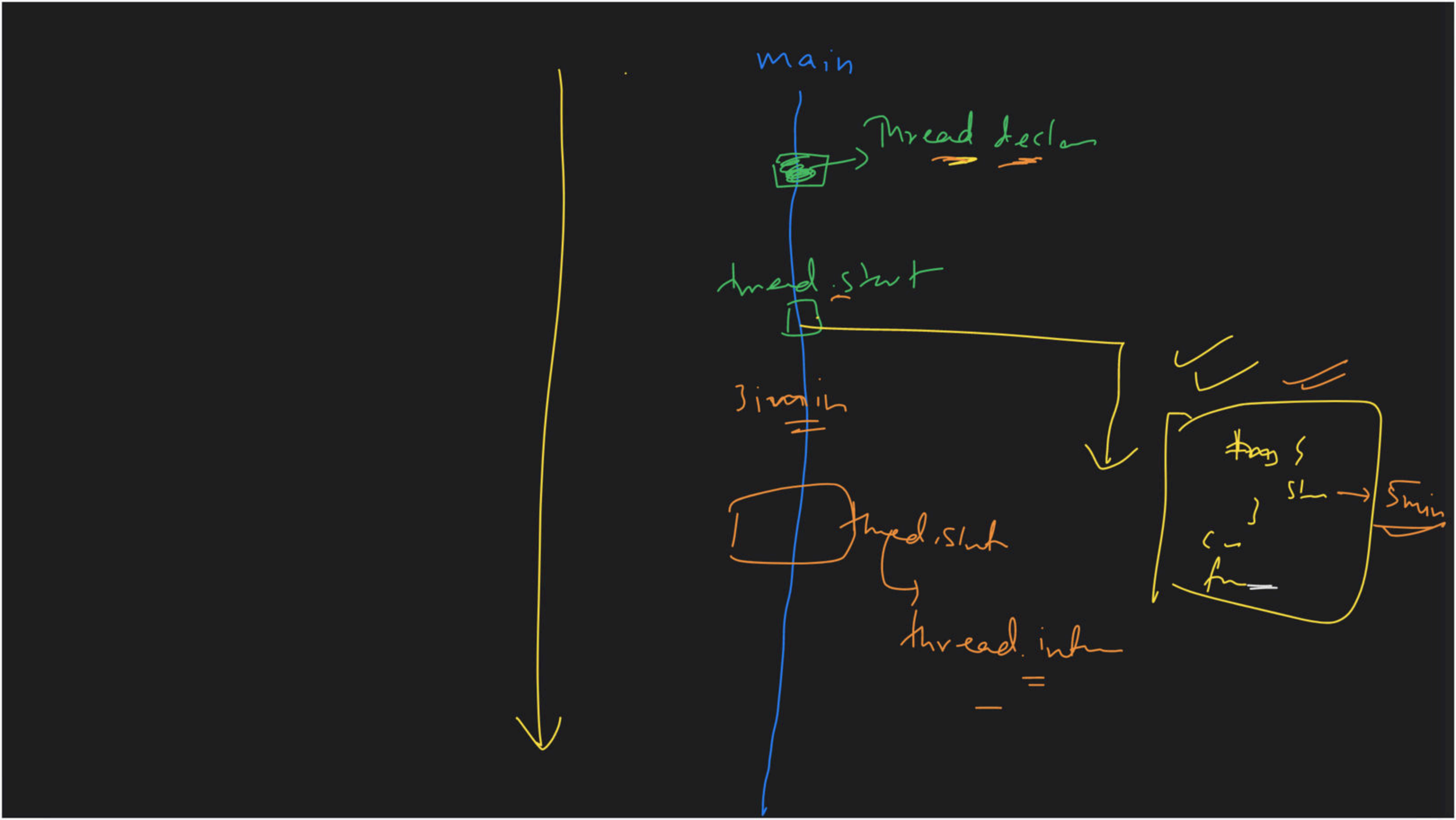
Is there any case where finally does not execute ?

Thread thread = new Thread(() -> {



});

thread.start();


main
Thread declar
thread.start
Join
thread.start
thread.inter

Throws keyword

We use the throws keyword in the method declaration to declare the type of exceptions that might occur within it.

Its syntax is:

```
accessModifier returnType methodName() throws ExceptionType1, ExceptionType2 … {
  // code
}
```

```
import java.io.*;
class Main {
  public static void findFile() throws NullPointerException, IOException,

    // code that may produce NullPointerException
    … … …

    // code that may produce IOException
    … … …

    // code that may produce InvalidClassException
    … … …
  }

  public static void main(String[] args) {
    try{
      findFile();
    } catch(IOException e1){
      System.out.println(e1.getMessage());
    } catch(InvalidClassException e2){
      System.out.println(e2.getMessage());
    }
  }
}
```

Throw Keyword

The throw keyword is used to explicitly throw a single exception. When an exception is thrown, the flow of program execution transfers from the try block to the catch block. We use the throw keyword within a method.

Its syntax is:

```
class Main {
  public static void divideByZero() {
    throw new ArithmeticException("Trying to divide by 0");
  }

  public static void main(String[] args) {
    divideByZero();
  }
}
```

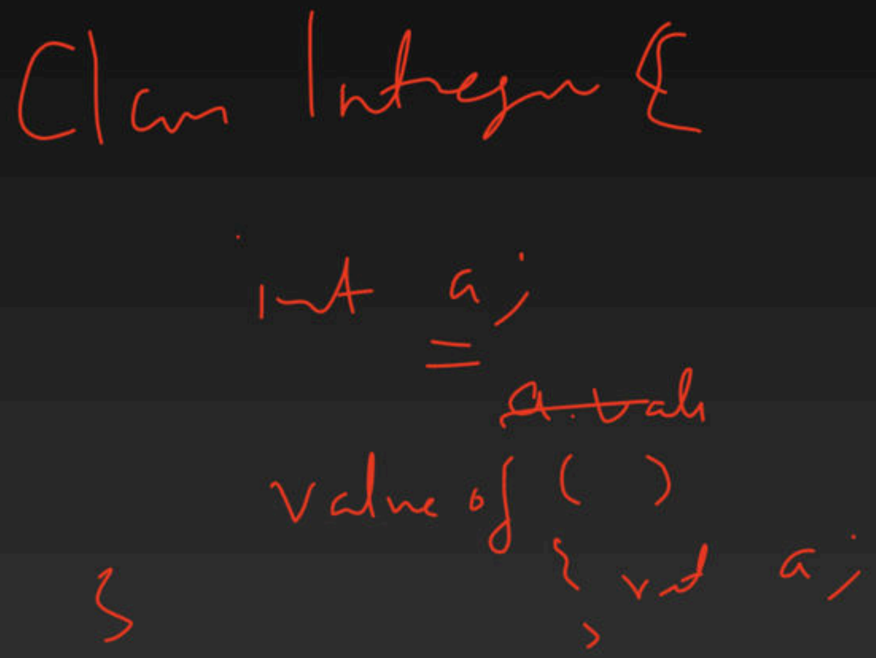
Wrapper Classes

- Love Babbar

Java Wrapper Class

The wrapper classes in Java are used to convert primitive types (int, char, float, etc) into corresponding objects.

- Each of the 8 primitive types has corresponding wrapper classes.



Primitive Type	Wrapper Class
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Need of Wrapper Class ?

- The classes in java.util package handles only objects and hence wrapper classes help in this case also.
- Data structures in the Collection framework, such as ArrayList and Vector, store only objects (reference types) and not primitive types.
- An object is needed to support synchronization in multithreading.

Example 1: Primitive Types to Wrapper Objects

```
class Main {
  public static void main(String[] args) {

    // create primitive types
    int a = 5;
    double b = 5.65;

    //converts into wrapper objects
    Integer aObj = Integer.valueOf(a);
    Double bObj = Double.valueOf(b);

    if(aObj instanceof Integer) {
      System.out.println("An object of Integer is created.");
    }

    if(bObj instanceof Double) {
      System.out.println("An object of Double is created.");
    }
  }
}
```

Example 2: Wrapper Objects into Primitive Types


```
class Main {
  public static void main(String[] args) {

    // creates objects of wrapper class
    Integer aObj = Integer.valueOf(23);
    Double bObj = Double.valueOf(5.55);

    // converts into primitive types
    int a = aObj.intValue();
    double b = bObj.doubleValue();

    System.out.println("The value of a: " + a);
    System.out.println("The value of b: " + b);
  }
}
```

Advantages of Wrapper Class

- In Java, sometimes we might need to use objects instead of primitive data types. For example, while working with collections.In such cases, wrapper classes help us to use primitive data types as objects.
- We can store the null value in wrapper objects. For example,

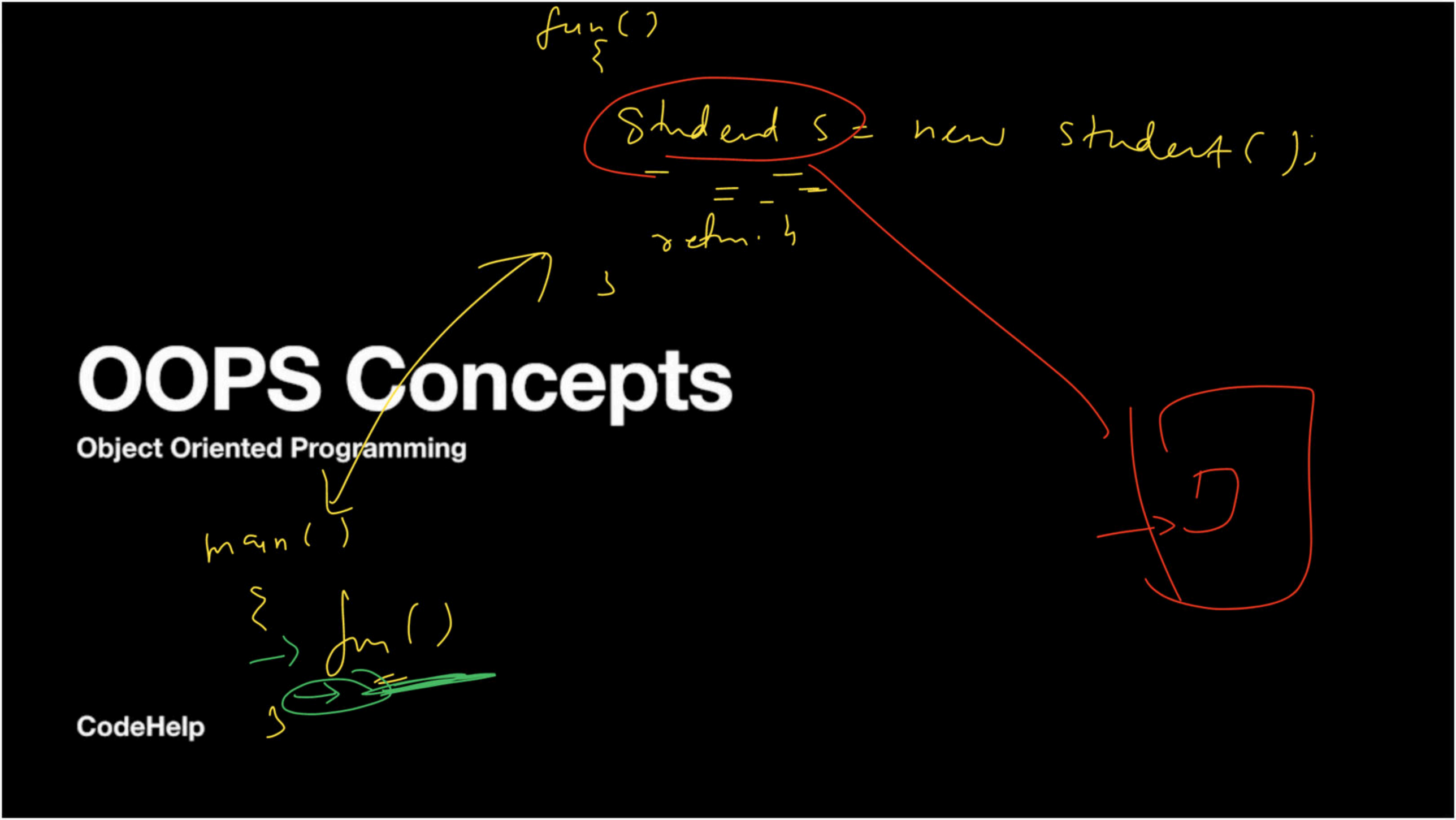
```
// error
ArrayList<int> list = new ArrayList<>();

// runs perfectly
ArrayList<Integer> list = new ArrayList<>();
```

```
// generates an error
int a = null;

// runs perfectly
Integer a = null;
```

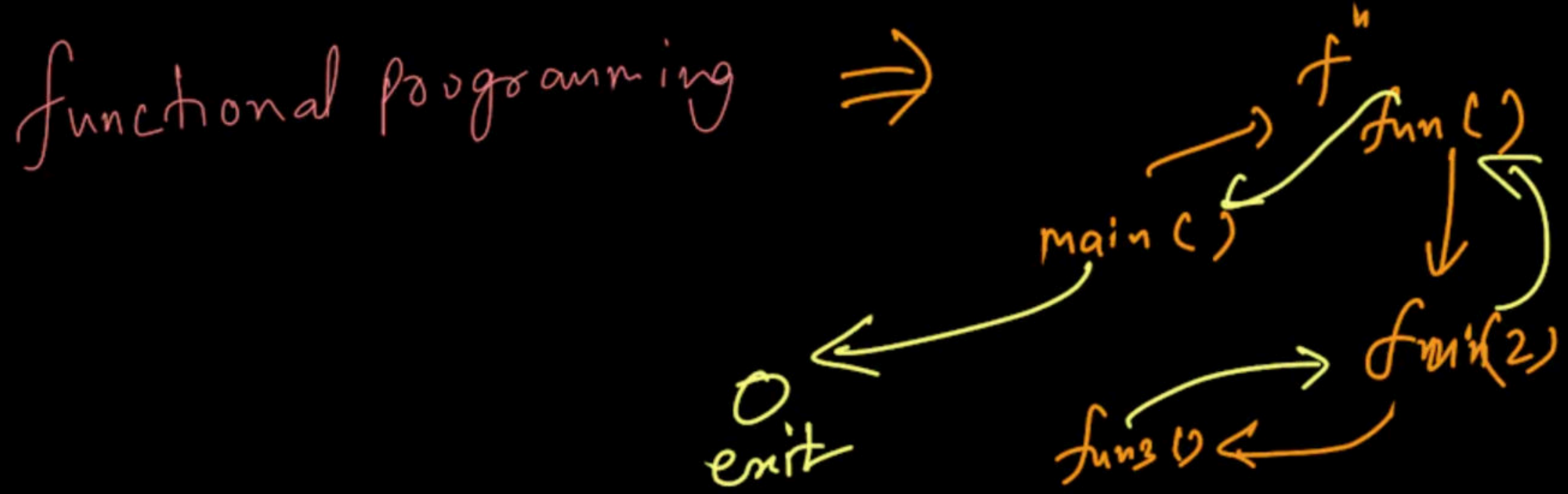
Note: Primitive types are more efficient than corresponding objects. Hence, when efficiency is the requirement, it is always recommended primitive types.


fun()
{
Student S = new Student();
}
main()
{
fun()
}
OOPS Concepts
Object Oriented Programming
CodeHelp

Why OOPS?

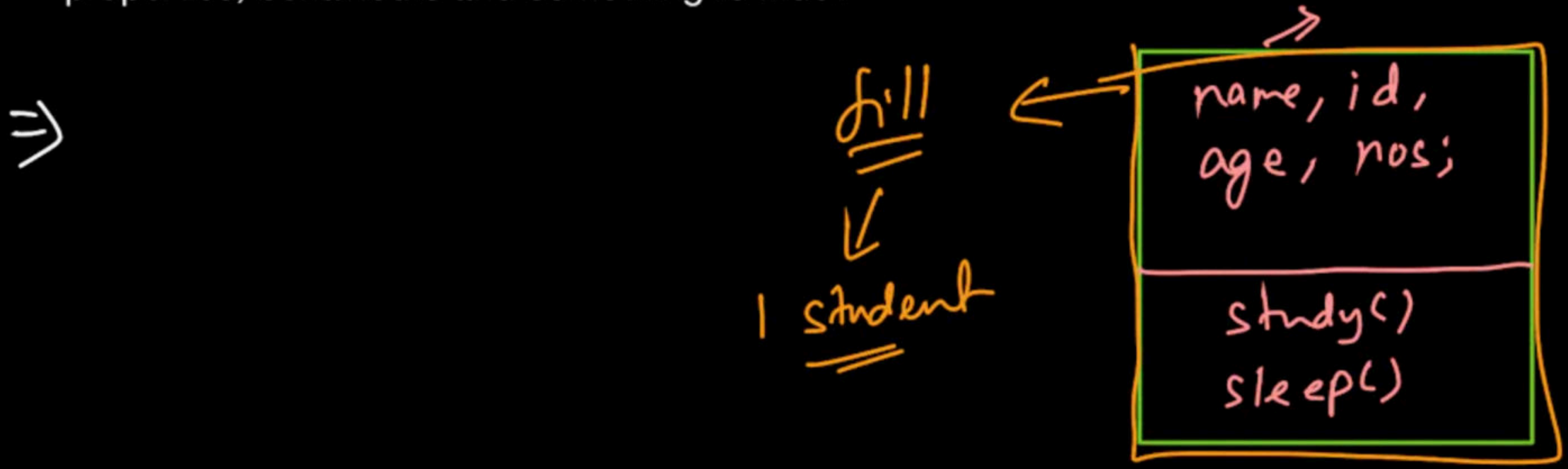
1. OOPS was developed because of limitations were discovered in earlier approaches to programming.

2. To appreciate what OOPS, Let's understand the limitations of earlier approach.



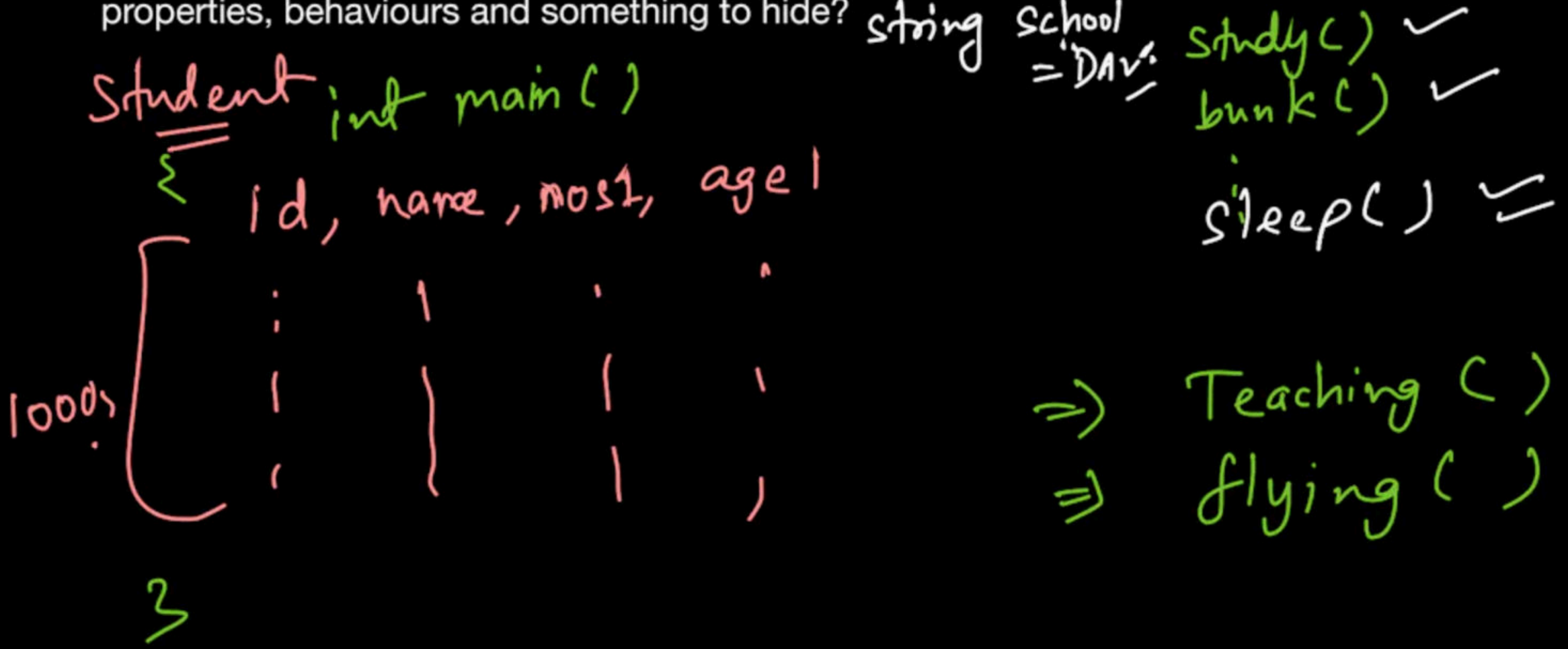
Multiple Students Example - Messy Code

1. How will you model a program having 100s of students, and each students have their own properties, behaviours and something to hide?



Multiple Students Example - Messy Code

1. How will you model a program having 100s of students, and each students have their own properties, behaviours and something to hide?

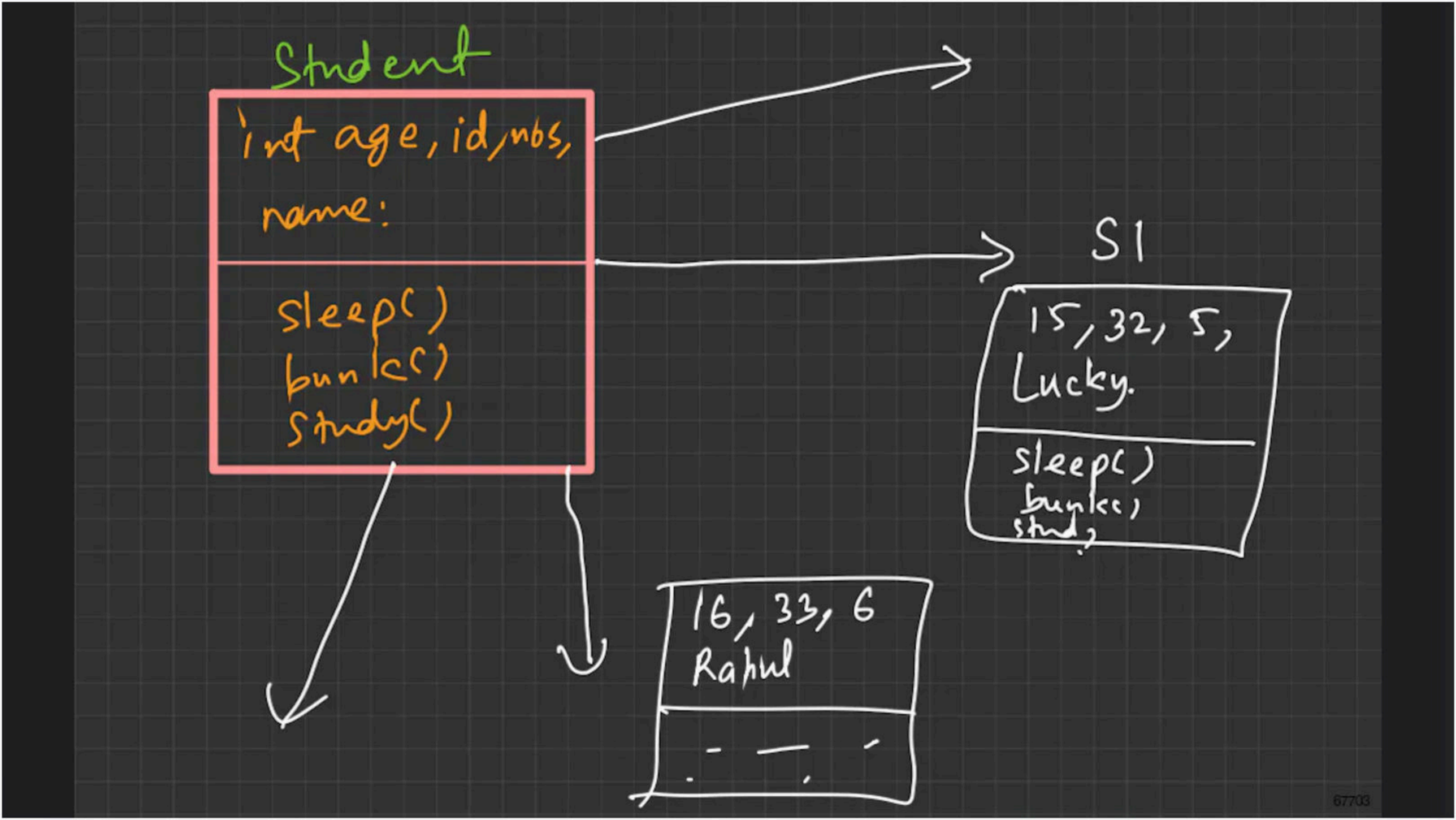


Drawbacks of Functional / Procedural Programming

1. Does not model real world problem very well.
2. If a new data is added, all the functions needs to be modified to access the Data.
3. Global data is accessible to all the functions.
4. No clear boundaries and well definition of code.
5. No Modularity: Functional programs can become monolithic and difficult to maintain as they grow in complexity.

What is OOP?

1. Programming is used to solve real-world problems, how can we model real-world systems with programming languages.
2. A Programming Style, involves dividing a program into pieces of objects that can communicate with each other.
3. Objects based coding style, in which each object (aka, real-world entity) has its own attributes and behaviour.
4. Fundamental Idea is to combine into single unit, both data and behaviour, that will promote Modularity.
5. **OOP promotes modularity** by encapsulating data and behavior within objects. This modular approach enhances code reusability and maintainability, as objects can be reused in different parts of the program.
6. OOP is LIFE (We'll understand this on-the-go)

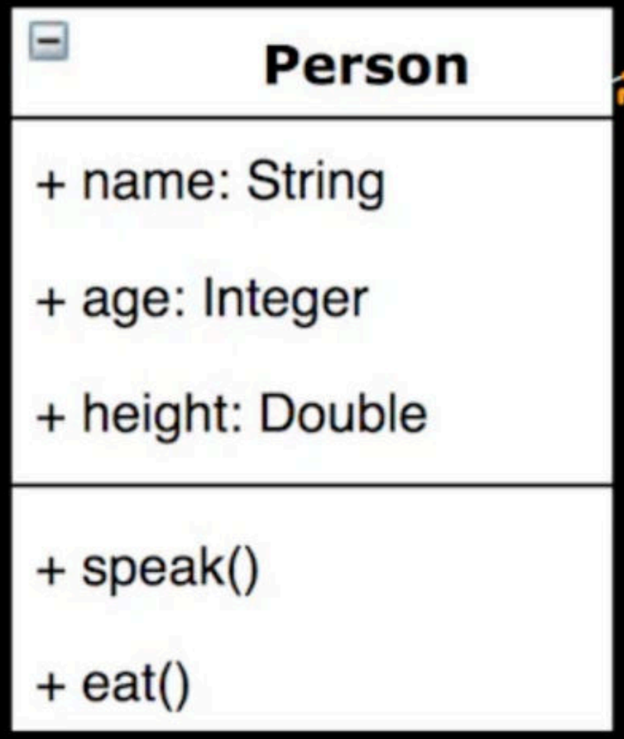

Student
int age, id, nos,
name:
sleep()
bunk()
study()
S1
15, 32, 5,
Lucky.
sleep()
bunk()
stud()
16, 33, 6
Rahul

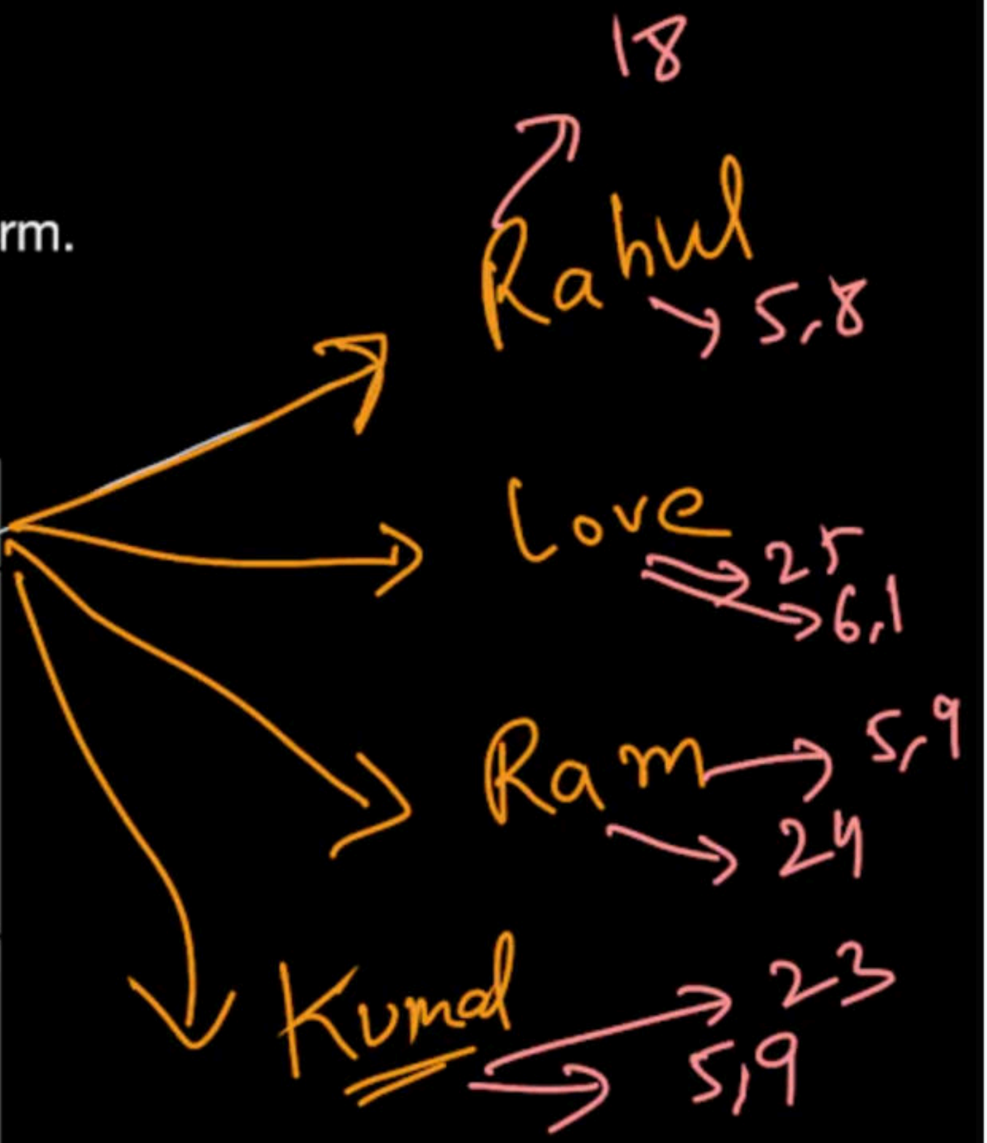
Objects and Classes.

1. Real world entities like cars, person, students, building etc., they all have some state and behaviour.
2. For e.g., a Student named Rahul, is an real world entity, in programming, he is an object.
3. What defines, how would an object look like?, there must be a Blueprint i.e., Class.
4. Hence, Object is an instance of a Class.

Attributes and Behaviour

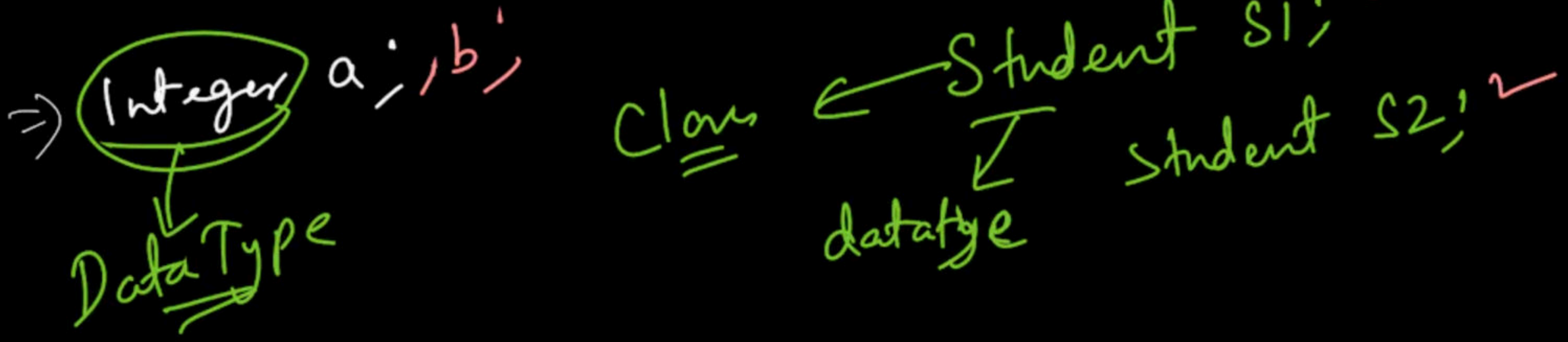
1. Attributes are state, properties of an object.
2. Behaviour is methods / functions that an object can perform.





So...

1. Object Oriented Programming (OOP) is a programming paradigm focused on implementing real-world objects.
2. The identification of code objects similar to real-life objects and structuring code using classes and objects signifies the use of OOP principles.
3. Classes and objects serve as the fundamental building blocks of the OOP concept.
4. Major Object Oriented languages include C++, Java, and Javascript.



So...

1. Object Oriented Programming (OOP) is a programming paradigm focused on implementing real-world objects.
2. The identification of code objects similar to real-life objects and structuring code using classes and objects signifies the use of OOP principles.
3. Classes and objects serve as the fundamental building blocks of the OOP concept.
4. Major Object Oriented languages include C++, Java, and Javascript.