

# Git & Github

# Create GitHub Repository

- Open <https://github.com/>
- Create or log in to an account
- Make repository
- Download Git <https://git-scm.com/install/windows>
- Open a terminal or CMD and verify the version `[git --version]`
- Configure Git `[git config --global user.email "your-email@gmail.com"]`

# Clone Repository

- Copy repository URL
- Clone repository [git clone URL]
- Make change in readme.md file
- Push changes
  - [git add .]
  - [git commit -m “msg”]
  - git push

# Java Main Function

# Syntax

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

# Variable

- What is a variable
- A variable is like a small box in memory where you keep some information
- Syntax: [data type] [variable name] = [value]
  1. `int age = 25;`
  2. `String name = "Dheeraj";`
  3. `double price = 99.50;`

# Gradle Build Tool

What is Gradle?

- Build tool for Java
- Compile and run your code
- Download and manage dependencies
- Run tests
- Package project & deploy

# Create Gradle Project

- Create a new project in IntelliJ IDEA with the name “java\_training”
- Choose gradle



# Upload Project to Github

- Create Gradle Java project (Gradle, grovy)
- Create a repository with the same folder name on [GitHub.com](https://github.com)
  1. enter repository name
  2. click on create repository
- Sync local folder with github repository
  - A. [git init]
  - B. [git remote add origin URL]
  - C. [git branch -M main]
  - D. [git add .]
  - E. [git commit -m "first commit"]
  - F. [git push -u origin main]

## Working on a remote repository

1. Created github repository
2. `[git clone URL]`
3. we created file
4. `[git add .]`
5. `[git commit - m "msg"]`
6. `[git push]`

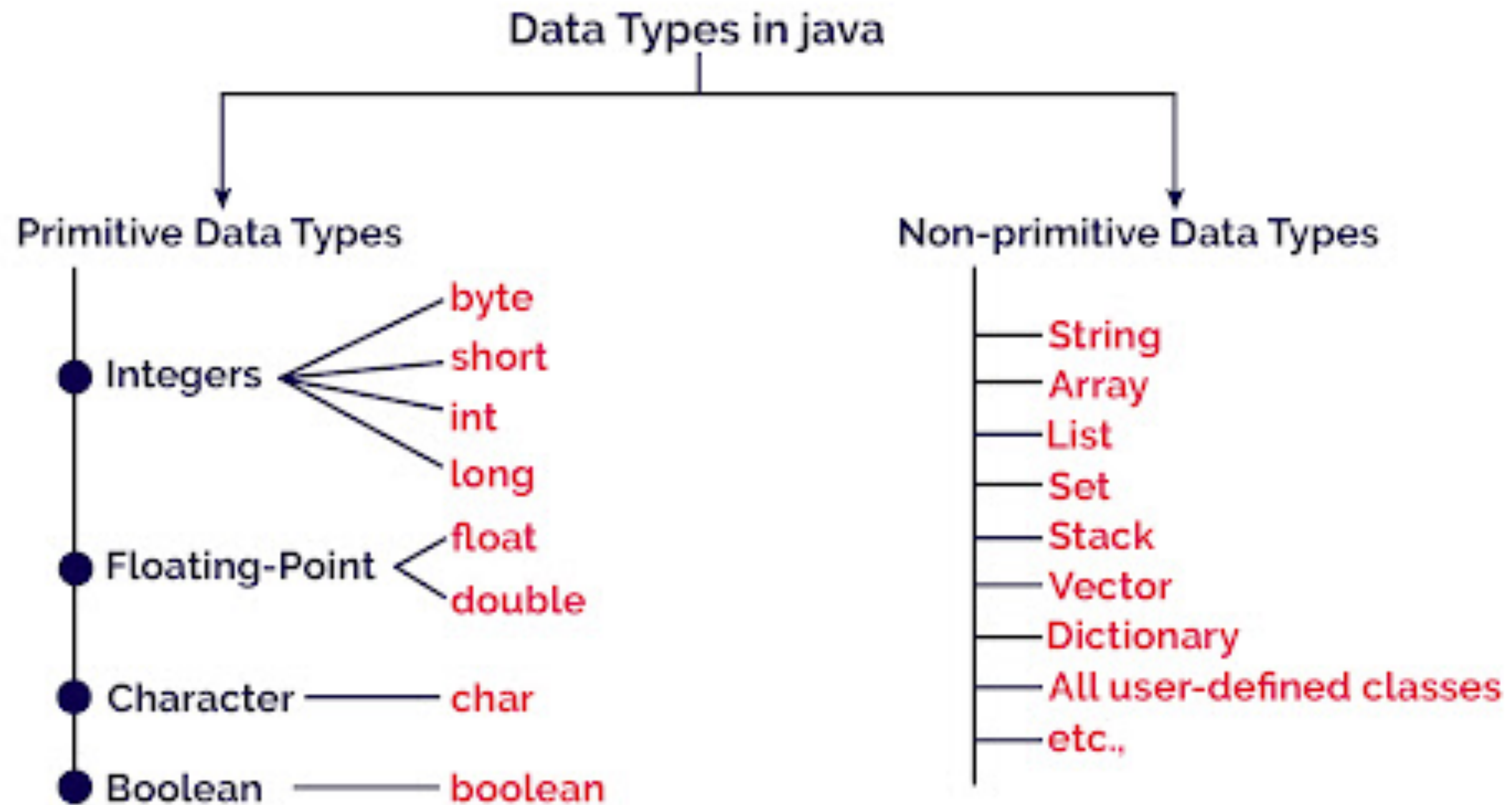
## Pushing local folder on github repository

1. Created local folder (gradle java project)
2. Created GitHub repository
3. `[git init]`
4. `[git remote add origin URL]`
5. `[git branch -M main]`
6. `[git add .]`
7. `[git commit -m "msg"]`
8. `[git push -u origin main]`

# Data Type

- The data type will tell what kind of value a variable will store
- Java has 2 categories of data types
  1. Primitive data type
  2. Non-Primitive data type

# Data Type Categories



# Primitive Data Type

<b>byte</b>	1 byte	10	Small numbers (-128 to 127)
<b>short</b>	2 bytes	200	Medium numbers
<b>int</b>	4 bytes	10,000	Normal integer numbers
<b>long</b>	8 bytes	100000L	Very large numbers
<b>float</b>	4 bytes	10.5f	Decimal (less accurate)
<b>double</b>	8 bytes	10.567	Decimal (more accurate)
<b>char</b>	2 bytes	'A'	Single character
<b>boolean</b>	1 bit	true/false	Conditions (true/false)

# Quiz

- <https://create.kahoot.it/share/basic-of-data-type-quiz/1e950c60-1c4b-4d8a-8022-8dc2082a2d3c>

# Non-Primitive Data Type

<b>String</b>	<code>String name = "Dheeraj";</code>	Sequence of characters
<b>Array</b>	<code>int[] arr = {1,2,3};</code>	Stores multiple values of same type
<b>Class</b>	<code>class Student {}</code>	Blueprint for creating objects
<b>Object</b>	<code>Student s = new Student();</code>	Instance created from class
<b>Interface</b>	<code>interface Animal {}</code>	Contains abstract methods
<b>Enum</b>	<code>enum Day { MON, TUE }</code>	Collection of constants

# Primitive vs Non Primitive Data Type

- Primitive data stores the actual value directly.
- Non-primitive data types store the reference (address) of the value in memory.
- Primitive data type stored in stack or heap?



# Quiz

- <https://create.kahoot.it/details/421632aa-a66e-4d29-9556-ea9e7ac552a1>

# Operator

- Operators are **symbols** that perform **actions on variables or values**
- Example: +, -, \*, /, ==, <, &&, etc.
- Java has 7 types of operators.

# Type of Operator

1. Arithmetic Operator
2. Assignment Operator
3. Relational Operator
4. Logical Operator
5. Unary Operator
6. Bitwise Operator
7. Ternary Operator

# 1. Arithmetic Operator

Used for basic math

+	Addition	$a + b$
-	Subtraction	$a - b$
*	Multiplication	$a * b$
/	Division	$a / b$
%	Modulus (remainder)	$10 \% 3 = 1$

# 2. Assignment Operators

Used to assign values

=	Assign	x = 10
+=	Add & assign	x += 5 (x = x + 5)
-=	Sub & assign	x -= 2
*=	Multiply & assign	x *= 3
/=	Divide & assign	x /= 2

# 3. Relational Operators

Used to compare values, and it return true/false

==

Equal

!=

Not equal

>

Greater than

<

Less than

>=

Greater or equal

<=

Less or equal

# 4. Logical Operators

Used with boolean values

&&

AND (both conditions must be true)

||

OR (at least one condition must be true)

!

NOT (reverses result)

# 5. Unary Operators

Operate on one operand.

+

Positive

-

Negative

++

Increment

--

Decrement



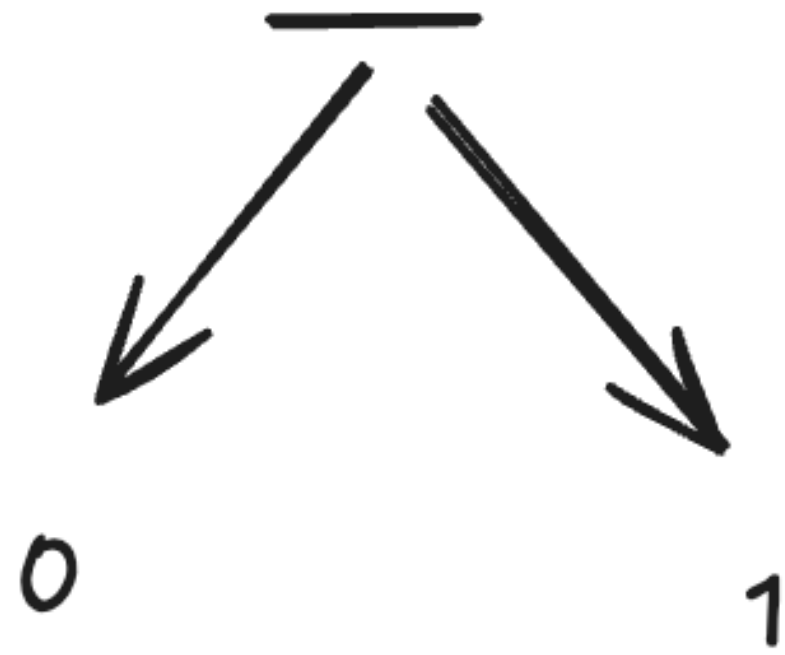
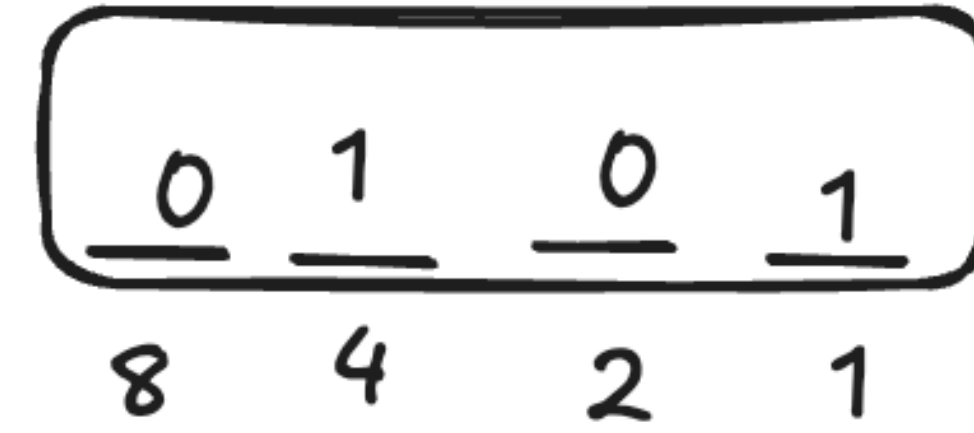
# 6. Bitwise Operators

These operators work on **bits (0 and 1)** at the binary level.

Operator	Name	Meaning
&	AND	1 if both bits are 1
	,	OR
^	XOR	1 if bits are different
~	NOT	Flips bits (1→0, 0→1)
<<	Left Shift	Multiply by 2
>>	Right Shift	Divide by 2
>>>	Unsigned Right Shift	Shift without keeping sign

$\sim a = -(a + 1)$

int a = 5;



$2^{\text{position of bit}}$

- `int a = 5;`
- In java int is 32 bit & long is 64 bit
- \_\_\_\_\_ 0 1 1 0
- 0101

# Who computer store negative no

- Hint: 2's complement
- **Two's complement** is a way of representing **negative numbers** in binary in computers.
  - A. Invert bits [this is 1's complement]
  - B. Add 1 [final 2's complement result]

# 7. Ternary Operator

`condition ? value_if_true : value_if_false;`

# Handle Input in Java

- Scanner is a built-in Java class used to **take input from the user** (keyboard input)
- Import scanner class
  - **[import java.util.Scanner;]**

# Method of Scanner Class

## Common Scanner Methods

Method	Purpose	Example
<code>nextInt()</code>	Reads integer	<code>int a = sc.nextInt();</code>
<code>nextFloat()</code>	Reads float value	<code>float f = sc.nextFloat();</code>
<code>nextDouble()</code>	Reads double value	<code>double d = sc.nextDouble();</code>
<code>next()</code>	Reads one word	<code>String s = sc.next();</code>
<code>nextLine()</code>	Reads full line	<code>String s = sc.nextLine();</code>
<code>nextBoolean()</code>	true/false input	<code>boolean b = sc.nextBoolean();</code>

# Control Flow

- Control flow determines the order in which statements execute in a Java program.
- Control flow are used
  - Decide which code to run
  - Repeat actions
  - Skip parts depending on conditions
- 3 types of control flow statements
  1. Decision-making statements (if-else, switch)
  2. Looping statements (for, while, do while)
  3. Branching statements (break, continue)



# Develop a Calculator