

# Data Types, Variables and Operators

Storing and manipulating data in programs

# Learning objectives

---

- ▶ Variables
- ▶ Data types
- ▶ Operators
- ▶ Constants (the final keyword)
- ▶ Type conversion

# Variables and data types

---

- ▶ Variables are used to store and manipulate data in a program
- ▶ A variable is a named memory location that holds a value of a specific data type (like a String of text, a character, an integer number, etc)

# Demo 1 - Variables and data types

---

- ▶ Declaring variables
- ▶ Initializing variables (by assigning values to them)

# Variable declaration

---

- ▶ Java is a strongly typed language, all variables have a data type
- ▶ The data type of a variable is decided when it is created and it can never change
- ▶ Normally, a variable is declared (created) by specifying its data type followed by the name of the variable, for example: `int number;`

# Variable initialization

- ▶ After declaring a variable, it can be initialized by being assigned a value
- ▶ For example, first a declaration like this: `int age;`
- ▶ Then an assignment of a value: `age = 27;`
- ▶ The `=` is an assignment operator and not an equals operator

# Assignment operators

Operator	Meaning	Example int a = 10; int b = 5;
=	Assignment	a = b; (a is equal to 5)
+=	Add and assignment	a += b; (same as: a = a + b; // a is 15)
-=	Subtract and assignment	a -= b; (same as: a = a - b; // a is 5)
*=	Multiply and assignment	a *= b; (same as: a = a * b; // a is 50)
/=	Divide and assignment	a /= b; (same as: a = a / b; // a is 2)
%=	Modulus and assignment	a %= b; (same as: a = a % b; // a is 0)

# Java naming conventions for variables

---

- ▶ Variables: camelCase starting with a lowercase letter, just like methods
- ▶ For example: name, age, firstName, homeAddress



# Primitive types

Type	Size	Default	Description
byte	1 bytes	0	Integer, representing -128 to +127
short	2 bytes	0	Integer, representing -32768 to +32767
int	4 bytes	0	Integer, representing -2 <sup>31</sup> to 2 <sup>31</sup> -1
long	8 bytes	0	Integer, representing -2 <sup>63</sup> to 2 <sup>63</sup> -1
float	4 bytes	0.0	Floating point number
double	8 bytes	0.0	Floating point number (higher precision)
boolean	1 bit	false	Boolean, representing true or false
char	2 bytes	'\u0000'	Represents a 16 bit Unicode character

# Non-primitive types

---

- ▶ There are also non-primitive types like objects
- ▶ A String for example represents a string of text and is an object
- ▶ This will be covered in other modules in the course

# Constants

---

- ▶ A constant is a variable with a value that can never change
- ▶ Constants are declared with the final keyword, for example: `final int pi;`
- ▶ Once a value is assigned to a constant it can never change
- ▶ Naming convention for constants: all uppercase with possible underlines
- ▶ For example: `NAME`, `AGE`, `FIRST_NAME`, `HOME_ADDRESS`

# Demo 2 - Primitive types and Constants

---

- ▶ Declaring and using primitive types
- ▶ Declaring and using constants

# Type conversion

---

- ▶ Type conversion is the process of changing the data type of a value
- ▶ This is necessary when performing operations involving variables of different data types
- ▶ There are two types of type conversion:
  - ▶ Implicit type conversion - Occurs automatically, no risk of losing data
  - ▶ Explicit type conversion - Not automatic, may result in losing data

# Implicit type conversion

---

- ▶ Occurs automatically when a value of a smaller data type is assigned to a variable of a larger data type
- ▶ The conversion is considered safe since there is no risk of losing data
- ▶ For example, assigning an **int** to a **double** is an implicit conversation

# Explicit type conversion (casting)

- ▶ Explicit type conversion (also known as casting) is not automatic but requires an explicit casting to the target data type
- ▶ It is required when a value of a larger data type is assigned to a variable of a smaller data type
- ▶ This conversion is not considered safe since there is a risk of losing data
- ▶ For example, assigning a **double** x to an **int** y requires an explicit cast like this: `int y = (int)x;`

# Arithmetic operators

Operator	Meaning	Example int a = 10; int b = 5;
+	Addition	a + b will give 15
-	Subtraction	a - b will give 5
*	Multiplication	a * b will give 50
/	Division	a / b will give 2
%	Modulus	a % b will give 0
++	Increment	a++ will give 11
--	Decrement	a-- will give 9



# String concatenation operator

- ▶ The plus sign (+) is also used as the String concatenation operator
- ▶ String concatenation means combining two Strings
- ▶ The plus sign used in a combination with at least one String is not an arithmetic addition but a String concatenation
- ▶ For example: `String greeting = "Hello " + "World!";`

# Demo 3 - Type conversion

---

- ▶ Implicit type conversion
- ▶ Explicit type conversion
- ▶ Arithmetic operators and type conversion

# Exercise 1 - Variables and types

- ▶ Put this code inside a main method:
- ▶ It will not work
- ▶ Can you change it to make it compile?
- ▶ Can you make it compile while keeping the types of the three variables?
- ▶ Hint: Use casting to be able to put the result in the circumference variable
- ▶ Did you get the result 300 or 314? Why could the result be 300?

```
int diameter = 100;  
double pi = 3.14;  
  
int circumference = diameter * pi;  
System.out.println(circumference);
```

# Exercise 2 - Variables and types

- ▶ Create a float variable with the value 0
- ▶ Then add 0.1 to this variable ten times
- ▶ Then print the variable
- ▶ You probably would expect the result to be 1.0 but it isn't!
- ▶ Will it work better if using a variable of type double? Try it out!
- ▶ (We will be looking for an alternative solution later on in the course)

```
float amount = 0;  
amount += 0.1f; // repeat ten times  
  
System.out.println(amount);
```

# Exercise 3 - Variables and types

- ▶ Which code examples are valid?
- ▶ First think about it and then try it out in IntelliJ

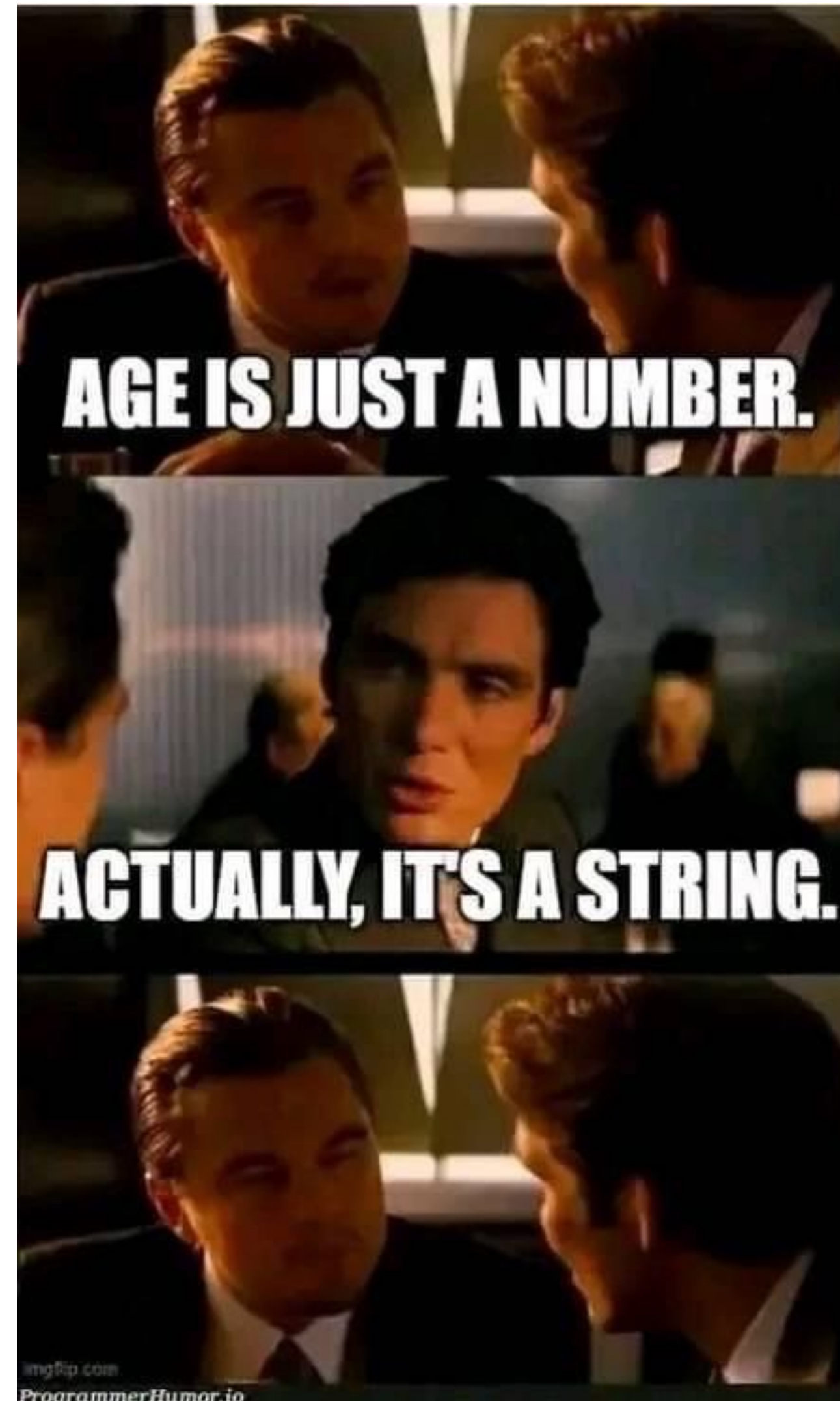
```
// 1  
final int CONSTANT_1;  
CONSTANT_1 = 1;
```

```
// 2  
int final CONSTANT_2;
```

```
// 3  
final int TEST_3 = 3;
```

```
// 4  
final int TEST_4 = 4;  
TEST_4 = 0;
```

# Data types



# Learning objectives

---

- ▶ Variables
- ▶ Data types
- ▶ Operators
- ▶ Constants (the final keyword)
- ▶ Type conversion