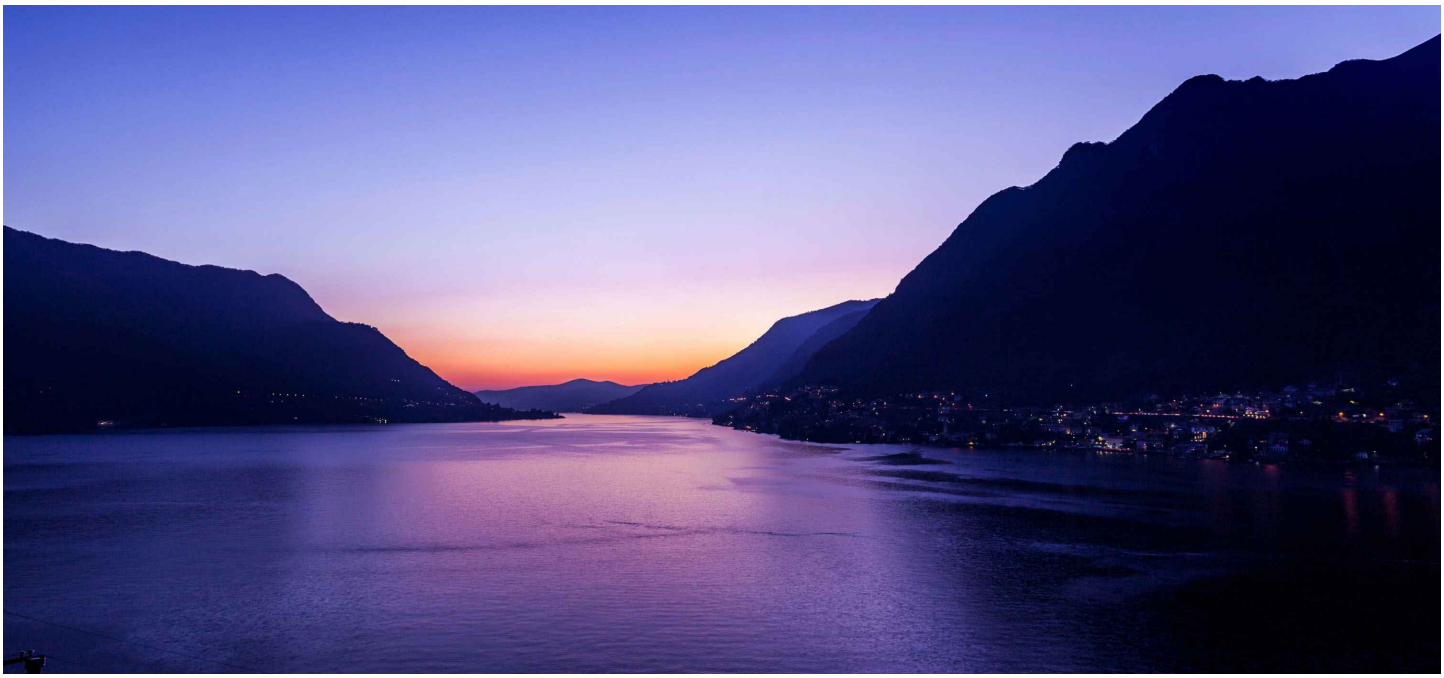

Primitives Basics

Author: Vincent Lau

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

In Java, `primitive` data types are the most basic data types that are not objects. They represent simple values and have corresponding wrapper classes for object-oriented operations.

Declarations & Implementations

Here are the primitive data types along with examples for each of them:

1. `byte` : Used to store small whole numbers. Range: -128 to 127.

```
// byte
byte num = 10;
```

2. `short` : Used to store small whole numbers. Range: -32,768 to 32,767.

```
// short
short count = 1000;
```

3. `int` : Used to store whole numbers (integers) without decimal places. Range: -2147483648 to 2147483647

```
// int
int age = 25;
```

4. `long` : Used to store large whole numbers. Range: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

```
// long
long population = 7894561230L;
```

5. `double` : Used to store floating-point numbers in decimal places.

```
// double
double salary = 2500.50;
```

6. `float` : Used to store floating-point numbers with decimal places, but less precise than `double` .

```
// float
```

```
// suffix can be f or F, norm to use f
float temperature = 36.5f;
```

7. `char` : Used to store a single character.
   - `char` is a ***16-bit unsigned integer*** representing a **Unicode-encoded character**. Unlike double-quoted String, *char* variables should be enclosed in **single quotes**.
   - char's **range is from 0 to 65,535**. It can be used to represent a **subset** of Unicode characters within that range.
   - For `Unicode` values, it refers to symbols, different human language characters, etc.

```
// char
char c = 'a';
char c = 65;
```

8. `boolean` : Used to store either `true` or `false` values.

```
// boolean: true or false;
boolean isPassed = true;
boolean isGenius = false;
```

Characteristics & Usages

These primitive data types are used to store different kinds of values in Java, depending on the requirements of your program. They are **lightweight** and **efficient** to work with since they are directly stored in memory.

Primitives are also used extensively in Java programs for **storing and manipulating simple data**, performing **arithmetic operations, and making logical decisions**. They are fundamental to the Java language and provide the foundation for more complex data types and operations.

Initialization

```
class Example {
    public static void main(String[] args) {
        int number; // Local variable without initialization
        System.out.println(number); // Compile error: Variable 'number' might not have been initialized
```

```
    }
}

```

Literal notation

In Java, when you specify a numeric literal without any suffix, the compiler interprets it as an `int` by default. This is known as integer literal notation.

For example, the value `1` is treated as an `int` unless otherwise specified. You can assign it to an `int` variable directly, like `int number = 1;`, without any issue.

Similarly, if you specify a decimal number without any suffix, such as `0.0`, it is interpreted as a `double` by default. The `double` type provides a higher precision for floating-point numbers compared to `float`.

To specify a numeric literal as a `float`, you need to add the `f` or `F` suffix to the value. For example, `0.0f` or `0.0F` indicates a `float` literally.

Here are some examples to illustrate these concepts:

```
int number = 1; // `1` is treated as an `int` by default
double decimal = 10.0; // `10.0` is treated as a `double` by default
float floatValue = 0.0f; // `0.0f` explicitly specifies a `float` literal
long bigNumber = 10000000000L; // `10000000000L` indicates a `long` literal
byte smallNumber = 10; // `10` fits within the range of a `byte` and is
  treated as an `int`
```

It's important to be aware of these default interpretations and suffixes to ensure that the literals are correctly represented with the desired data types.

Note: In Java, the `double` type is used more commonly than `float` due to its higher precision, unless there is a specific need for `float` in certain scenarios, such as conserving memory or when dealing with APIs that specifically require `float` values.