JAVA

What you learn ?

Java Data Types – Type Casting	
➢ Data Types	➢ Unicode System
➢ Type Casting	➢ User Input

Data Types

A data type in programming is a classification of data that defines the type of values that a variable.
In Java, there are two categories of data types: **Primitive data types and Not-Primitive(reference) data types.**

- **Primitive data** types are simple and usually small in size, and **they are passed by value**, which means that when they are used as parameters in a method, the method receives a copy of the value rather than a reference to the original variable.

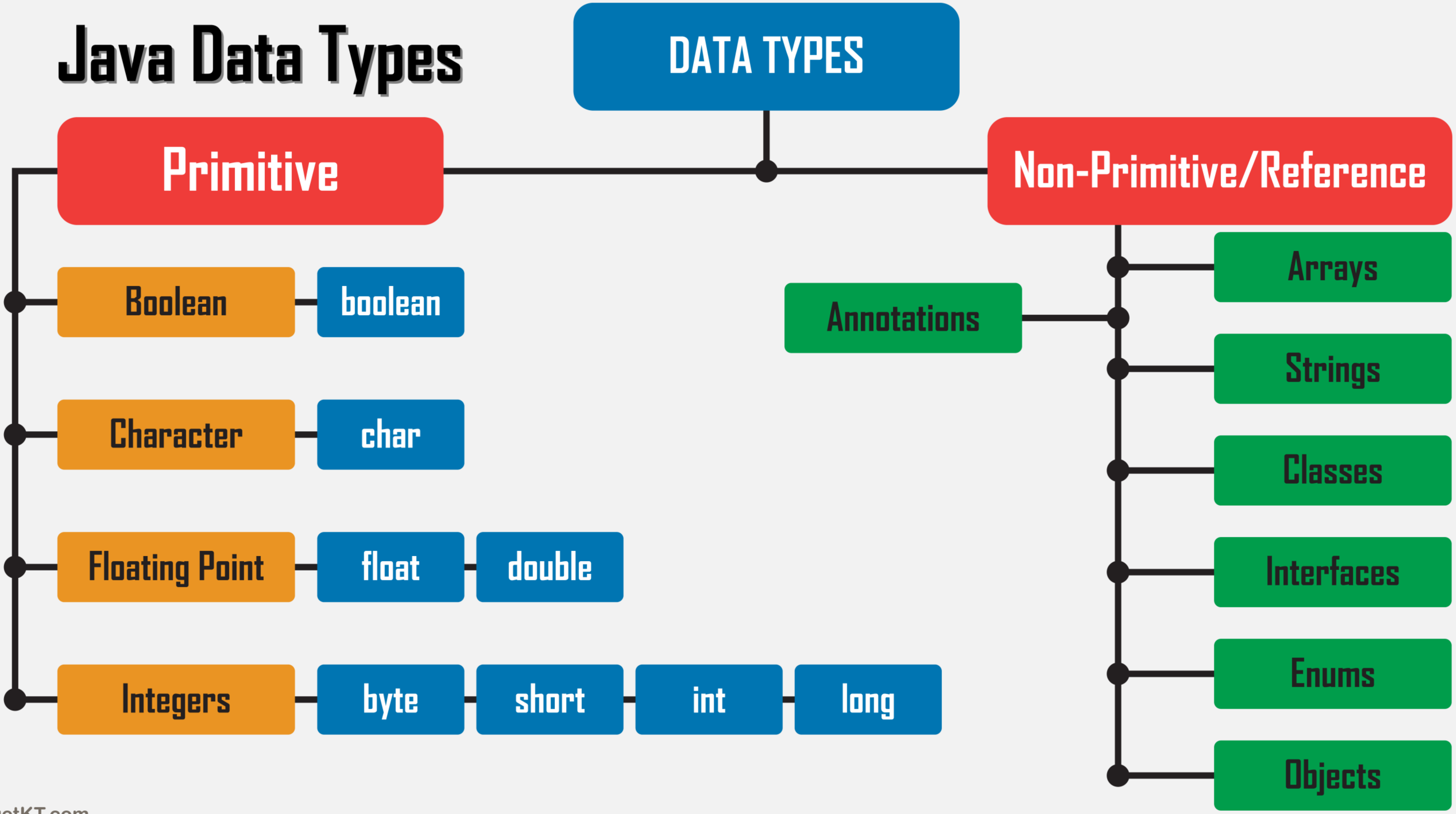
They are built-in to the programming language

Data Types

- **Non-Primitive (Reference) data types** are more complex data types that represent objects and other more complicated data structures. They are composed of smaller data types and are usually **created by the programmer**. Examples of reference data types in Java include String, Arrays, Classes, Interfaces, and Enumerations.

Reference data types are usually larger in size, and they are **passed by reference,** which means that when they are used as parameters in a method, the method receives a reference to the original variable rather than a copy of the value.

Java Data Types



Default value & size for Primitive Data Types

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Important about Char Data Type

Why char uses 2 byte in Java

- In Java, the char data type uses 2 bytes (16 bits) to represent a single Unicode character. This is because the **Unicode character set** contains a large number of characters, including non-Latin characters and special characters, that require more than 1 byte to be represented
- Because the char data type uses 2 bytes per character, it allows Java to represent a wide range of characters from the Unicode character set, including characters from non-Latin scripts such as Hindi,Chinese, Japanese, and Arabic.
- minimum value is '\u0000' (or 0)= NULL, and its maximum value is '\uffff' (or 65,535) = '?'.

https://home.unicode.org/

Why JAVA uses Unicode system?

Java uses the Unicode system because it provides a consistent way of representing characters from all languages, including those that require multiple bytes to represent them. **The Unicode standard assigns a unique code point to every character in every language**, making it possible for software developers to handle text in a variety of languages without having to worry about encoding issues.

The use of Unicode in Java is a key factor in making it a powerful and flexible language for developing applications that can work in a global environment.

Unicode system

Unicode is a character encoding standard that assigns a unique number, called a code point, to every character in the world's writing systems. The code points range from U+0000 to U+10FFFF, and each code point corresponds to a unique character or symbol.

However, **Unicode is not a specific encoding scheme for representing these code points as binary data.** There are several encoding schemes, such as UTF-8, UTF-16, and UTF-32, that can be used to represent Unicode code points as sequences of bytes.

Encoding Scheme

UTF-8: UTF-8 is a variable-length encoding scheme that uses between 1 and 4 bytes to represent each code point, depending on its value. It is the most commonly used encoding scheme on the web and in modern operating systems, and it is backward-compatible with ASCII. In UTF-8, ASCII characters are represented by a single byte, and non-ASCII characters are represented by multiple bytes.

UTF-16: UTF-16 is a fixed-length encoding scheme that uses 2 or 4 bytes to represent each code point, depending on its value. It is widely used in Windows operating systems and programming languages such as **Java** and C#. In UTF-16, code points in the Basic Multilingual Plane (BMP) are represented by 2 bytes, and code points outside the BMP are represented by a pair of 2-byte values called surrogate pairs.

Encoding Scheme

UTF-32: UTF-32 is a fixed-length encoding scheme that uses 4 bytes to represent each code point. It is not as commonly used as UTF-8 or UTF-16, but it is used in some specialized applications and programming languages such as Python. In UTF-32, each code point is represented by a single 32-bit value, which makes it the simplest encoding scheme in terms of byte order and byte alignment.

UTF-EBCDIC: UTF-EBCDIC is an encoding scheme that maps Unicode code points to EBCDIC characters, which are used in some mainframe computer systems.

Encoding Scheme

The choice of encoding scheme depends on the specific requirements of the application or system. UTF-8 is a good choice for web applications and text processing because of its compatibility with ASCII and its compact size for common characters. UTF-16 is a good choice for applications that need to handle characters outside the BMP, such as certain Asian languages. UTF-32 is a good choice for applications that require a fixed-size representation of each code point and need to perform random access or indexing of Unicode text.

Type Cast

- **Casting is process of changing one type value to another type in java**

1) Implicit Type Casting(Widening Casting) :

byte-->short-->int-->long-->float-->double

Implicit type casting, also known as widening or automatic type promotion, occurs when the conversion is done automatically by the compiler. This happens when a value of a smaller data type is assigned to a variable of a larger data type.

2) Explicit Type Casting(Narrowing Casting)

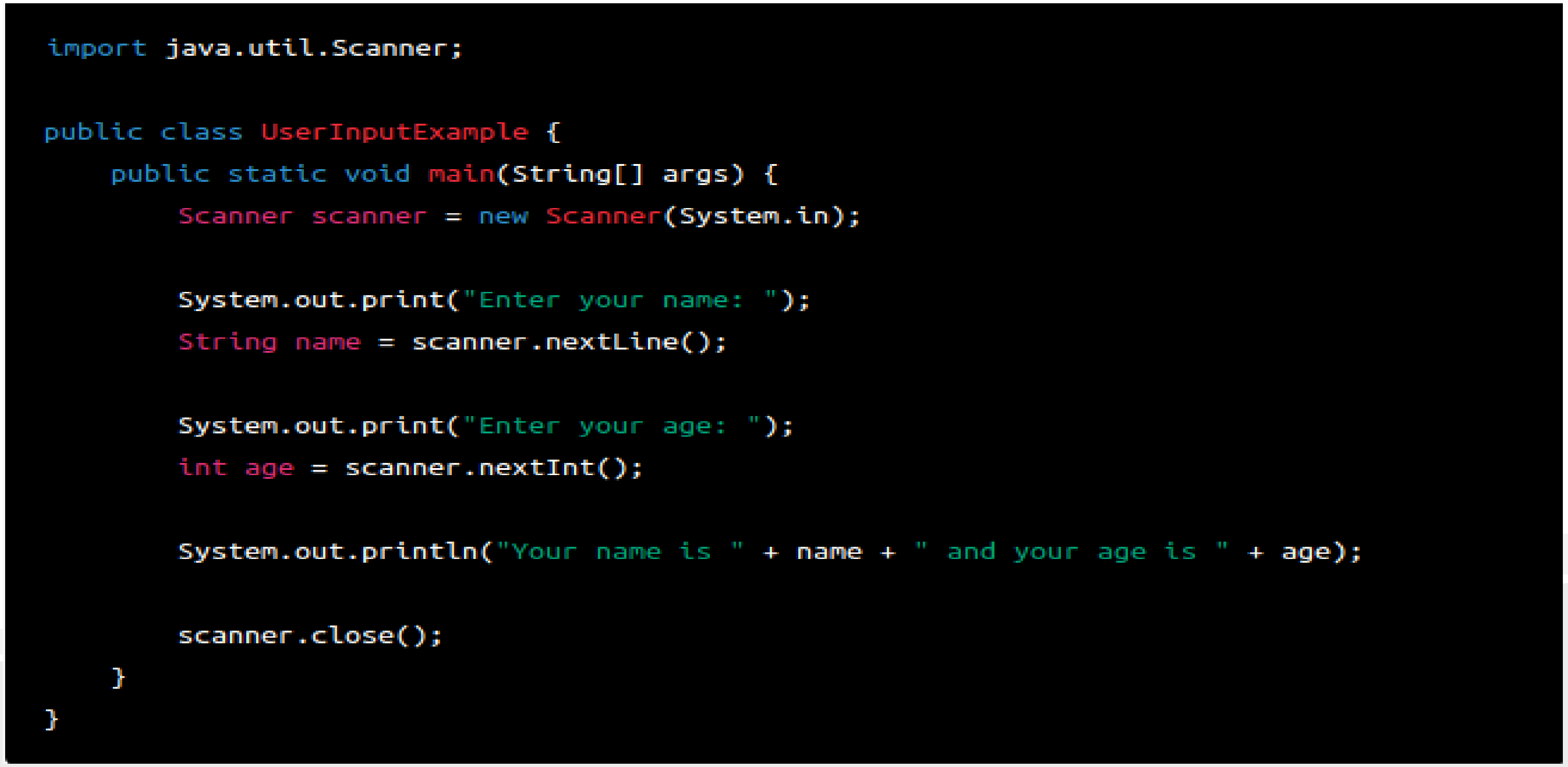
double-->float-->long-->int-->short-->byte

Explicit type casting, also known as narrowing or manual type conversion, occurs when the conversion is done manually by the programmer. This happens when a value of a larger data type is assigned to a variable of a smaller data type. In this case, the programmer needs to use a type casting operator to explicitly cast the value to the desired data type

Take user input in JAVA

➢ Java Scanner Class

In Java, you can take user input using the Scanner class, which is part of the **java.util** package. The **Scanner class** provides methods to read input from various sources, such as the keyboard or a file.



```
import java.util.Scanner;

public class UserInputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        System.out.println("Your name is " + name + " and your age is " + age);

        scanner.close();
    }
}
```

Take user input in JAVA

Methods of Scanner class:

Method	Description
int nextInt()	It is used to scan the next token of the input as an integer.
float nextFloat()	It is used to scan the next token of the input as a float.
double nextDouble()	It is used to scan the next token of the input as a double.
byte nextByte()	It is used to scan the next token of the input as a byte.
String nextLine()	Advances this scanner past the current line.
boolean nextBoolean()	It is used to scan the next token of the input into a boolean value.
long nextLong()	It is used to scan the next token of the input as a long.
short nextShort()	It is used to scan the next token of the input as a Short.
BigInteger nextBigInteger()	It is used to scan the next token of the input as a BigInteger.
BigDecimal nextBigDecimal()	It is used to scan the next token of the input as a BigDecimal.

Thanks

Anirudha Gaikwad