Data types and Type Casting

Numeric Data Types – Overview

Type	Range	Size
byte	-128 to 127	8 bits
short	-32,768 to 32,767	16 bits
int	-2,147,483,648 to +2,147,483,647	32 bits
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	64 bits
float	±1.4E-45 to ±3.4E+38	32 bits
double	±4.9E-324 to ±1.8E+308	64 bits

Type Casting in Java

byte → short → int → long → float → double

- **Type casting** =
  unary operation that converts a value from one data type to another.
- Two types:
  - **Widening (implicit):** smaller → larger range (automatic).
  - **Narrowing (explicit):** larger → smaller range (manual).

Widening is safe; narrowing risks data loss.

Widening vs. Narrowing

Type	Direction	Automatic?	Risk	Example
Widening	Small → Large	Yes	None	int → double
Narrowing	Large → Small	No	Data loss	double → int

```
// Widening (implicit)
System.out.println((double)1 / 2);   // Output: 0.5


// Narrowing (explicit)
System.out.println((int)1.7);        // Output: 1
(truncated)
```

Syntax of Type Casting

Rules:

- Use (`DataType`) before variable or value.
- Does **not** modify the original variable.

Syntax examples:

```
(DataType)variableName;
(DataType)value;
```

Example :

```
double d = 4.5;
int i = (int)d;  // i = 4, d remains 4.5
```

Example task

Object: Write a program that converts an integer variable holding the value 123 into a floating-point variable and then back to an integer again. Print both converted values.

```
Original Integer Value: 123
Converted Float Value: 123.0
Converted Back Integer Value: 123
```

```java
public class TypeCastingExample {
    public static void main(String[] args) {
        // Initial integer value
        int originalValue = 123;

        // Casting integer to float
        float floatValue = (float) originalValue;

        // Casting float back to integer
        int convertedBackValue = (int) floatValue;

        // Printing results
        System.out.println("Original Integer Value: " +
originalValue);
        System.out.println("Converted Float Value: " +
floatValue);
        System.out.println("Converted Back Integer Value: "
+ convertedBackValue);
    }
}
```

Boolean Data Type

Booleans are fundamental for conditional logic

- Represents `true` or `false` values.
- Reserved words: `true`, `false` (cannot be identifiers).

Declaration

```
boolean lightsOn = true;
boolean isComplete = false;
```

Relational Operators (Comparison)

== (comparison) ≠ **=** (assignment)

Java Op.	Math Symbol	Name	Example (radius=5)
<	<	Less than	radius < 0
<=	≤	Less/equal	radius <= 0
>	>	Greater than	radius > 0
>=	≥	Greater/equal	radius >= 0
==	=	Equal to	radius == 0
!=	≠	Not equal	radius != 0