Type Casting & Data Conversion

What is Type Casting?

In real life, we convert things all the time.

Examples:

- Rupees to dollars
- Kilometers to meters

In Java, **type casting means converting one data type into another**.

It is mainly used when:

- We want to store one type of value into another type
- Java does not allow automatic conversion

Why Type Casting is Needed

Java is a **strictly typed language**.

This means:

- Every variable has a fixed data type
- Java does not allow unsafe conversions

Sometimes:

- A small value needs to go into a bigger type
- A bigger value needs to go into a smaller type

That is where type casting is required.

Simple Example Without Casting

```
int a = 10;

double b = a;
```

Explanation:

- int → smaller data type
- double → larger data type

Java allows this automatically.

This is called **implicit casting**.

Types of Type Casting in Java

Java supports **two main types** of type casting.

1. Implicit Type Casting
2. Explicit Type Casting

Implicit Type Casting (Automatic)

Implicit casting is done **automatically by Java**.

Rules:

- Small data type → Bigger data type
- No data loss

Example:

```
int a = 10;

double b = a;
```

Explanation:

- int → source type
- double → destination type
- Java converts 10 into 10.0

This process is also called **Widening**.

Widening Conversion Order

Common widening flow:

int → long → float → double

Java allows this safely.

Explicit Type Casting (Manual)

Explicit casting is done **manually by the programmer**.

Rules:

- Bigger data type → Smaller data type
- Data loss is possible

Example:

```
double x = 10.75;

int y = (int) x;
```

Explanation:

- (int) → casting operator

- Decimal part is removed
- Result becomes 10

This process is also called **Narrowing**.

Narrowing Conversion Order

Common narrowing flow:

double → float → long → int

Java does not allow this automatically.

Why Java Does Not Allow Automatic Narrowing

Because narrowing:

- Can lose data
- Can change the value

Example:

```
double d = 99.99;
int i = d;   // ❌ error
```

Java forces us to **explicitly mention casting** to avoid mistakes.

Another Simple Casting Example

```
char ch = 'A';
int value = ch;
```

Explanation:

- char stores character
- Internally, characters have numeric values
- 'A' becomes 65

Java converts character to number automatically.

Type Casting with int and byte

```
int a = 130;
byte b = (byte) a;
```

Explanation:

- byte range is limited
- Value changes due to overflow

This shows **why explicit casting must be used carefully**.

Data Conversion vs Type Casting

Type Casting:

- Happens between compatible data types
- Mostly with primitives

Data Conversion:

- Happens between different formats
- Example: number to string

Example:

```
int num = 10;
String s = String.valueOf(num);
```

Remember This

Small → Big => Automatic
Big → Small => Manual

If you remember this rule, type casting becomes easy.