Type Casting in Java - Detailed Notes

1. Introduction

Type Casting in Java is the process of converting a variable from one data type to another.

There are two types of type casting:

1. Implicit Casting (Widening)

2. Explicit Casting (Narrowing)

2. Implicit Type Casting (Widening)

- Happens automatically.

- Smaller data types are promoted to larger data types.

- No data loss occurs.

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

Syntax: No special syntax required.

Example 1: byte to int

```
public class ImplicitCastingExample1 {
    public static void main(String[] args) {
        byte b = 10;
        int i = b;
        System.out.println("Byte value: " + b);
        System.out.println("Int value after casting: " + i);
    }
}
```

Output:

Byte value: 10

Int value after casting: 10

Example 2: int to double

```
public class ImplicitCastingExample2 {
    public static void main(String[] args) {
        int num = 25;
        double d = num;
        System.out.println("Int value: " + num);
```

Type Casting in Java - Detailed Notes

```
        System.out.println("Double value after casting: " + d);
    }
}
```

Output:

Int value: 25

Double value after casting: 25.0

3. Explicit Type Casting (Narrowing)

- Needs to be done manually by the programmer.

- Larger data types are converted to smaller types.

- May cause data loss or truncation.

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

Syntax: (targetType) variableName;

Example 1: double to int

```
public class ExplicitCastingExample1 {
    public static void main(String[] args) {
        double d = 9.7;
        int i = (int) d;
        System.out.println("Double value: " + d);
        System.out.println("Int value after casting: " + i);
    }
}
```

Output:

Double value: 9.7

Int value after casting: 9

Example 2: int to byte with data loss

```
public class ExplicitCastingExample2 {
    public static void main(String[] args) {
        int i = 130;
        byte b = (byte) i;
        System.out.println("Int value: " + i);
```

Type Casting in Java - Detailed Notes

```
        System.out.println("Byte value after casting: " + b);
    }
}
```

Output:

Int value: 130

Byte value after casting: -126 (Data Loss)

4. Data Loss During Casting

- When a larger value is cast into a smaller type, data may be lost or altered.

Example: long to int

```
public class DataLossExample {
    public static void main(String[] args) {
        long l = 2147483648L; // Beyond int range
        int i = (int) l;
        System.out.println("Long value: " + l);
        System.out.println("Int value after casting: " + i);
    }
}
```

Output:

Long value: 2147483648

Int value after casting: -2147483648 (Overflow)

5. Implicit vs Explicit Casting

Comparison between Implicit and Explicit Type Casting:

Aspect	Implicit Casting	Explicit Casting
Process	Automatic	Manual
Data Loss	No	Possible
Syntax	No cast operator	Requires cast operator
Direction	Smaller -> Larger	Larger -> Smaller

6. Important Points to Remember

Type Casting in Java - Detailed Notes

- Implicit casting is safe; explicit casting should be handled carefully.
- Always check for possible data loss when narrowing types.
- Casting between non-compatible types (like int to boolean) is not allowed.