

by Kunal Sir

Assignment : Type Casting

Learning Objectives

By completing this assignment, students will be able to:

- Understand what type casting is in Java
 - Differentiate between implicit (widening) and explicit (narrowing) type casting
 - Predict output of Java programs involving type casting
 - Avoid common mistakes related to data loss and casting
-



CJC
Complete Java Classes

Part A: Tricky MCQs (15 Questions)

1. What will be the output?

```
int a = 10;  
double b = a;  
System.out.println(b);
```

- A) 10
- B) 10.0
- C) Compilation error
- D) Runtime error

Answer: B) 10.0

- ✓ `int` is implicitly converted to `double` → widening casting
- ✓ Output shows decimal format

2. Which type of casting is happening below?

```
float f = 25;
```

- A) Explicit casting
- B) Narrowing casting
- C) Implicit casting
- D) Invalid casting

Answer: C) Implicit casting

- ✓ `int` value 25 is automatically converted to `float`
-

3. What will be the output?

```
double d = 9.8;  
int i = (int) d;  
System.out.println(i);
```

- A) 9.8
- B) 10
- C) 9
- D) Compilation error

Answer: C) 9

- ✓ Explicit casting removes decimal part
- ✓ Java **does not round**, it **truncates**

4. Which of the following conversions may cause data loss?

- A) int → long
- B) float → double
- C) double → int
- D) char → int

Answer: C) double → int

- ✓ Narrowing conversion
- ✓ Decimal part is lost → **data loss possible**

5. What will be the output?

```
char ch = 'A';  
int x = ch;  
System.out.println(x);
```

- A) A
- B) 65
- C) Compilation error
- D) Runtime error

Answer: B) 65

✓ 'A' ASCII value = **65**

✓ char → int happens implicitly

6. Which casting is done automatically by JVM?

- A) Explicit
- B) Narrowing
- C) Manual
- D) Implicit

Answer: D) Implicit

✓ JVM performs widening conversion automatically

7. What will be printed?

```
int a = 130;  
byte b = (byte) a;  
System.out.println(b);
```

- A) 130
- B) -126
- C) 127
- D) Compilation error

Answer: B)-126

- ✓ byte range = -128 to 127
- ✓ 130 exceeds range → **overflow**

8. Which statement is TRUE?

- A) Explicit casting is always safe
- B) Implicit casting may cause data loss
- C) Explicit casting can cause data loss
- D) Java does not support type casting

Answer: C) Explicit casting can cause data loss

- ✓ Narrowing conversions are unsafe

9. What happens if explicit casting is not done where required?

- A) Program runs normally
- B) Runtime error
- C) Logical error
- D) Compilation error

Answer: D) Compilation error

✓ Java forces explicit casting for narrowing

10. What will be the output?

```
int a = 5;  
int b = 2;  
double c = a / b;  
System.out.println(c);
```

- A) 2.5
- B) 2
- C) 2.0
- D) Compilation error

10. Answer: C) 2.0

✓ $a / b = 5 / 2 \rightarrow \text{integer division} = 2$

✓ Stored in double $\rightarrow 2.0$

11. To get correct decimal output in Q10, what should be done?

- A) Cast a to double
- B) Cast b to int
- C) Use long
- D) No change needed

Answer: A) Cast a to double

✓ `(double) a / b` → correct decimal division

12. Which conversion is NOT allowed implicitly?

- A) `int` → `double`
- B) `byte` → `int`
- C) `double` → `float`
- D) `char` → `int`

Answer: C) `double` → `float`

✓ Narrowing conversion

✓ Needs explicit casting

13. What will be the output?

```
float f = 7.9f;  
int i = (int)(f + 0.5);  
System.out.println(i);
```

- A) 7
- B) 7.9
- C) 8
- D) Compilation error

Answer: C) 8

✓ $7.9 + 0.5 = 8.4$

✓ Casting removes decimal → 8

14. Which data type has the largest range?

- A) int
- B) float
- C) double
- D) byte

Answer: C) double

✓ double has the **largest range**

by Kunal Sir

15. Type casting is mainly used to:

- A) Increase execution speed
- B) Reduce code length
- C) Convert one data type to another
- D) Avoid syntax errors

Answer: C) Convert one data type to another

✓ Main purpose of type casting



Part B: Problem Statements (15 Problems)

1. Write a program to demonstrate implicit type casting from int to double.

```
class Demo1 {  
    public static void main(String[] args) {  
        int a = 10;  
        double b = a;  
        System.out.println(b);  
    }  
}
```

-
2. Write a program to convert a double value into int using explicit casting and display both values.

```
class Demo2 {  
    public static void main(String[] args) {  
        double d = 9.8;  
        int i = (int) d;  
        System.out.println(d);  
        System.out.println(i);  
    }  
}
```

-
3. Take an integer value and store it into a float variable using implicit casting. Print the result.

```
class Demo3 {  
    public static void main(String[] args) {  
        int a = 20;  
        float f = a;  
        System.out.println(f);  
    }  
}
```

4. Write a program that shows data loss during explicit type casting (double to int).

```
class Demo4 {  
    public static void main(String[] args) {  
        double d = 12.7;  
        int i = (int) d;  
        System.out.println(i);  
    }  
}
```

5. Convert a char value into its ASCII value using type casting.

```
class Demo5 {  
    public static void main(String[] args) {  
        char ch = 'A';  
        int ascii = ch;  
        System.out.println(ascii);  
    }  
}
```

6. Write a program where division of two integers gives incorrect output. Then fix it using type casting.

```
class Demo6 {  
    public static void main(String[] args) {  
        int a = 5, b = 2;  
        double result = (double) a / b;  
        System.out.println(result);  
    }  
}
```

7. Demonstrate narrowing conversion from int to byte and print the result.

```
class Demo7 {  
    public static void main(String[] args) {  
        int a = 130;  
        byte b = (byte) a;  
        System.out.println(b);  
    }  
}
```

8. Write a program to convert float to int and explain the output using comments.

```
class Demo8 {  
    public static void main(String[] args) {  
        float f = 8.9f;  
        int i = (int) f;  
        System.out.println(i); // decimal removed  
    }  
}
```

9. Take user input as double and convert it into int. Display both values.

```
import java.util.Scanner;  
  
class Demo9 {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        double d = sc.nextDouble();  
        int i = (int) d;  
        System.out.println(d);  
        System.out.println(i);  
    }  
}
```

10. Write a program that adds an int and a double and stores the result in a double variable.

```
class Demo10 {  
    public static void main(String[] args) {  
        int a = 10;  
        double b = 2.5;  
        double sum = a + b;  
        System.out.println(sum);  
    }  
}
```

11. Write a program to show implicit casting in expressions (int + double).

```
class Demo11 {  
    public static void main(String[] args) {  
        int a = 5;  
        double b = 4.5;  
        System.out.println(a + b);  
    }  
}
```

12. Convert a large int value into byte and observe the output.

```
class Demo12 {  
    public static void main(String[] args) {  
        int a = 200;  
        byte b = (byte) a;  
        System.out.println(b);  
    }  
}
```

13. Write a program that demonstrates why explicit casting is mandatory in narrowing conversion.

```
class Demo13 {  
    public static void main(String[] args) {  
        double d = 10.5;  
        int i = (int) d;  
        System.out.println(i);  
    }  
}
```

14. Create a program that converts marks (double) into integer percentage using type casting.

```
class Demo14 {  
    public static void main(String[] args) {  
        double marks = 89.7;  
        int percentage = (int) marks;  
        System.out.println(percentage);  
    }  
}
```

Complete Java Classes

by Kunal Sir

15. Write a Java program that explains (using comments) the difference between implicit and explicit type casting with examples.

```
class Demo15 {  
    public static void main(String[] args) {  
  
        // Implicit casting  
        int a = 10;  
        double d = a;  
  
        // Explicit casting  
        double x = 9.8;  
        int y = (int) x;  
  
        System.out.println(d);  
        System.out.println(y);  
    }  
}
```

