Java Wrapper Classes, Autoboxing, Unboxing, and Collections

1. Introduction to Wrapper Classes

Java is an **object-oriented language**, but primitive data types like `int`, `char`, `float`, etc., are **not objects**. Java provides **wrapper classes** to convert these primitives into objects.

Primitive Type	Wrapper Class
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

Why Wrapper Classes?

- Used in **collections** (which only work with objects).
- Provides **utility methods** (like parsing, comparing).
- Supports **autoboxing/unboxing**.
- Null values (unlike primitives).

Example:

```
int x = 10;
Integer obj = Integer.valueOf(x); // Wrapping int into Integer
System.out.println(obj); // Output: 10
```

---

2. Autoboxing and Unboxing

Autoboxing:

Automatic conversion of a **primitive to its corresponding wrapper class**.

```
int a = 5;
Integer obj = a; // Autoboxing (int → Integer)
```

Unboxing:

Automatic conversion of a **wrapper class object to its corresponding primitive**.

```
Integer obj = 10;
int b = obj; // Unboxing (Integer → int)
```

Example:

```
public class AutoUnboxingExample {
    public static void main(String[] args) {
        Integer i = 100;  // Autoboxing
        int j = i;        // Unboxing
        System.out.println("Integer Object: " + i);
        System.out.println("Primitive int: " + j);
    }
}
```

---

3. Converting Between Wrapper and Primitive Types

Primitive to Wrapper

```
int x = 20;
Integer obj = Integer.valueOf(x); // Manual boxing
```

Wrapper to Primitive

```
Integer obj = 30;
int y = obj.intValue(); // Manual unboxing
```

□ Example:

```
public class ConversionExample {
    public static void main(String[] args) {
        double d = 12.34;
        Double dObj = Double.valueOf(d); // Primitive to wrapper

        double d2 = dObj.doubleValue(); // Wrapper to primitive

        System.out.println("Wrapper: " + dObj);
        System.out.println("Primitive: " + d2);
    }
}
```

---

4. Methods and Constructors of Wrapper Classes

Most wrapper classes are **immutable** and provide **static methods**.

□ Common Methods:

a. `parseXxx(String s)`

Converts string to primitive.

```
int i = Integer.parseInt("123"); // Output: 123
```

b. `valueOf(String s)`

Returns wrapper object.

```
Integer i = Integer.valueOf("123"); // Output: 123 (as Integer)
```

c. `xxxValue()`

Converts wrapper object to primitive.

```
Integer i = 42;
double d = i.doubleValue(); // Output: 42.0
```

□ Example:

```
public class WrapperMethods {
    public static void main(String[] args) {
        String str = "456";
        int i = Integer.parseInt(str); // parse string to int
        Integer obj = Integer.valueOf(str); // string to wrapper
        int j = obj.intValue(); // wrapper to primitive

        System.out.println("int: " + i + ", Integer: " + obj + ", int again:
" + j);
    }
}
```

5. Using Wrapper Classes in Collections and Generics

Collections like `ArrayList` **cannot store primitives** directly — they require objects.
Wrapper classes enable **primitive storage** using **autoboxing**.

□ Example with ArrayList:

```
import java.util.ArrayList;

public class WrapperInCollection {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();

        list.add(10);   // Autoboxing
        list.add(20);
        list.add(30);

        for (int i : list) { // Unboxing while retrieving
            System.out.println(i);
        }
    }
}
```

□ Example with Generics:

```
public class GenericBox<T> {
    T value;

    void set(T value) {
        this.value = value;
    }

    T get() {
        return value;
    }

    public static void main(String[] args) {
        GenericBox<Integer> intBox = new GenericBox<>();
        intBox.set(100); // Autoboxing

        int x = intBox.get(); // Unboxing
        System.out.println("Value: " + x);
    }
}
```

Real-Time Scenario

Scenario: You are building a marks management system that stores student scores.

```
import java.util.*;

class StudentMarks {
    private String name;
    private ArrayList<Integer> marks; // wrapper type for storing scores

    public StudentMarks(String name) {
        this.name = name;
        marks = new ArrayList<>();
    }

    public void addMark(int score) {
        marks.add(score); // autoboxing
    }

    public void printMarks() {
        System.out.println("Marks for " + name + ":");
        for (int score : marks) { // unboxing
            System.out.println(score);
        }
    }
}
```

Summary

Feature	Purpose	Example
Wrapper Class	Convert primitive → object	`Integer i = Integer.valueOf(10);`
Autoboxing	Automatic primitive → object	`Integer i = 5;`
Unboxing	Automatic object → primitive	`int j = i;`
`parseInt()`	Convert string → int	`int i = Integer.parseInt("123");`
`valueOf()`	Convert string → Integer	`Integer i = Integer.valueOf("123");`
`intValue()`	Integer → int	`int i = obj.intValue();`