1. Wrapper Class

- Java is **object-oriented**, but primitive data types (`int`, `char`, `boolean`, etc.) are **not objects**.
- Wrapper classes **wrap primitive values into objects**, allowing them to be used in collections (like `ArrayList`) and APIs that require objects.

Example of Wrapper Classes:

- `int → Integer`
- `char → Character`
- `boolean → Boolean`
- `double → Double`
- `float → Float`

2. Autoboxing

Definition: Automatic conversion of **primitive → Wrapper class**.

- Done by Java compiler.
- Useful in Collections (`ArrayList`, `HashMap`) because they only work with objects, not primitives.

Example:

```
import java.util.*;

public class AutoBoxingExample {
    public static void main(String[] args) {
        int a = 10;
        Integer obj = a;  // Autoboxing: int → Integer

        ArrayList<Integer> list = new ArrayList<>();
        list.add(5);   // Autoboxing: int → Integer

        System.out.println("Integer object: " + obj);
        System.out.println("ArrayList: " + list);
    }
}
```

3. Unboxing

Definition: Automatic conversion of **Wrapper class → primitive**.

- Also handled by the compiler.

Example:

```
public class UnboxingExample {
    public static void main(String[] args) {
        Integer obj = 20;   // Autoboxing
        int b = obj;        // Unboxing: Integer → int

        System.out.println("Wrapper object: " + obj);
        System.out.println("Primitive value: " + b);
    }
}
```

4. Mixed Example

```
public class BoxingUnboxing {
    public static void main(String[] args) {
        // Autoboxing
        int x = 100;
        Integer i = x;

        // Unboxing
        Integer y = 200;
        int j = y;

        System.out.println("Autoboxed: " + i);
        System.out.println("Unboxed: " + j);
    }
}
```

Summary

- **Wrapper class** → Converts primitive to object.
- **Autoboxing** → primitive → Wrapper (automatic).
- **Unboxing** → Wrapper → primitive (automatic).