Java Wrapper Classes

In Java, **wrapper classes** are used to **wrap primitive data types** into objects. This is useful when you need to use primitives in places that require objects, such as **collections (e.g., ArrayList)**, or for **utility methods** that operate on objects.

🔹 List of Java Wrapper Classes

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

🔹 Why Use Wrapper Classes?

1. **Collections require objects**, not primitives (e.g., `ArrayList<int>` is not allowed, but `ArrayList<Integer>` is).
2. **Utility methods** in `java.lang` like `Integer.parseInt()` work with wrapper classes.
3. **Serialization** requires objects.
4. **Null values** can only be represented using objects.
5. Used in **generics**, where primitives can't be used.

🔹 Autoboxing and Unboxing

Java automatically converts between primitive types and wrapper classes:

- **Autoboxing** – converting a primitive to an object:

```java
int a = 10;
Integer obj = a; // autoboxing
```

- **Unboxing** – converting an object to a primitive:

```java
Integer obj = 20;
```

```
int b = obj; // unboxing
```

🔹 Example: Using Wrapper Class in ArrayList

```
java

import java.util.*;

public class WrapperExample {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);   // autoboxes int to Integer
        list.add(20);

        for (Integer num : list) {
            System.out.println(num);  // unboxing happens automatically
        }
    }
}
```

🔹 Commonly Used Methods

Each wrapper class provides useful methods:

```
java

Integer.parseInt("123");          // Converts String to int
Double.parseDouble("12.34");      // Converts String to double
Boolean.parseBoolean("true");     // Converts String to boolean

Integer.toString(123);            // Converts int to String
Boolean.toString(true);           // Converts boolean to String
```

🔚 Summary

- Wrapper classes convert primitives into objects.
- Necessary for working with Java Collections and object-only APIs.
- Java supports autoboxing and unboxing for convenience.

Would you like to practice with some interview-style questions or coding examples?