Here are detailed notes based on the document "Wrapper Classes in Java":

Definition

- A **wrapper class** in Java wraps a **primitive data type** into an object.
- It provides a mechanism to convert:
  - Primitive values → Objects.
  - Objects → Primitive values.

Purpose

- Wrapper classes allow storage of primitive data as objects.
- They bridge the gap between primitive types and the **object-oriented** features of Java.

Examples

1. **Wrapping**:

   ```
   int x = 25;
   Integer iWrap = new Integer(x); // Wrapping primitive int into Integer object.
   ```

2. **Unwrapping**:

   ```
   int unWrapped = iWrap.intValue(); // Converting Integer object back to int.
   ```

Why Use Wrapper Classes?

- Many applications (e.g., business or web applications) require data sent to servers in object form.
- Primitive data types (e.g., `int`, `double`) cannot be sent directly as objects.
- Example: In e-commerce platforms, details like:
  - `name` → Object (`String`).
  - `credit card number` → Initially `int`, needs to be wrapped into an object (`Integer`).

Autoboxing and Unboxing

Introduced in Java 5.0:

1. **Autoboxing**:

   - Automatic conversion of primitive types to their wrapper class objects.
   - Example:

     ```
     Integer obj = 100; // Primitive int is automatically wrapped.
     ```

2. **Unboxing**:

   - Automatic conversion of wrapper objects back to primitive types.
   - Example:

     ```
     int a = obj; // Wrapper object is converted back to primitive int.
     ```

List of Wrapper Classes

Java provides eight wrapper classes for the eight primitive data types:

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

Key Characteristics

1. **Defined in** `java.lang` package.
2. **Immutable and final**:
   - Objects cannot be modified after creation.
3. **No-arg constructors**:
   - Wrapper classes do not have constructors without arguments.
4. **Character restrictions**:
   - `Character` class does not support a constructor with a `String` parameter.

Points to Remember

- Wrapper classes provide:
  - **Constructors**, **constants**, and **methods** for manipulating primitive data types.
- All wrapper classes are immutable and final.
- Useful for type conversions:
  - Example: `Integer.parseInt()` to convert `String` to `int`.

If you need these notes structured further or tailored to specific needs, let me know!