# Wrapper Class In Java

July 19, 2023
Wrapper Class in Java

By,
Muni Amala Telu

# Wrapper Class in Java:

A Wrapper class in Java is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and, in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

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

## 1.Character wrapper class:

The **Character** wrapper class in Java provides various methods to work with character values. Here are some commonly used methods of the **Character** class:

1. **isDigit(char ch)**: Returns true if the specified character is a digit (0-9), otherwise returns false.
2. **isLetter(char ch)**: Returns true if the specified character is a letter (a-z or A-Z), otherwise returns false.
3. **isWhitespace(char ch)**: Returns true if the specified character is a whitespace character (space, tab, newline, etc.), otherwise returns false.
4. **isUpperCase(char ch)**: Returns true if the specified character is an uppercase letter, otherwise returns false.
5. **isLowerCase(char ch)**: Returns true if the specified character is a lowercase letter, otherwise returns false.
6. **toUpperCase(char ch)**: Converts the specified character to uppercase.
7. **toLowerCase(char ch)**: Converts the specified character to lowercase.
8. **toString(char ch)**: Returns a string representation of the specified character.

8. **isVowel(char ch)**: Returns true if the specified character is a vowel (a, e, i, o, u, A, E, I, O, U), otherwise returns false.

9. **isConsonant(char ch**): Returns true if the specified character is a consonant (a letter that is not a vowel), otherwise returns false.

10. **compareTo(char x, char y):** Compares two characters numerically and returns the difference.

11. **equals(Object obj):** Compares this Character object to the specified object for equality.

These are just a few of the methods available in the Character class.

## 2.Byte Wrapper Class:

The Byte wrapper class in Java is a class that provides an object representation of the byte primitive type. It allows you to work with byte values as objects and provides various methods for performing operations on byte values. Here, some methods in Byte Wrapper Class are:

**1.Creating Byte Objects:**

- Byte(byte value): Constructs a Byte object with the specified byte value.
- Byte(String value): Constructs a Byte object from a string representation of a byte value.

**2. Value Retrieval:**

- byteValue(): Returns the byte value of the Byte object.

**3. Converting Bytes:**

- toString(): Converts the Byte object to a string representation.
- toHexString(byte value): Converts the specified byte value to a hexadecimal string representation.
- toOctalString(byte value): Converts the specified byte value to an octal string representation.
- toBinaryString(byte value): Converts the specified byte value to a binary string representation.

**4. Comparison and Equality:**

- equals(Object obj): Compares this Byte object to the specified object for equality.
- compareTo(Byte anotherByte): Compares two Byte objects numerically.

**5. Parsing and Conversion:**

- parseByte(String s): Parses the string argument as a signed decimal byte and returns the corresponding byte value.
- parseByte(String s, int radix): Parses the string argument as a byte with the specified radix and returns the corresponding byte value.

**6. Constants:**

- MIN_VALUE and MAX_VALUE: Constants representing the minimum and maximum values of the byte type.

## 3.Short Wrapper Class:

The Short wrapper class in Java provides an object representation of the short primitive type. It allows you to work with short values as objects and provides various methods for performing operations on short values.
Here some methods are:

1. **Creating Short Objects**:
   - Short(short value): Constructs a Short object with the specified short value.
   - Short(String value): Constructs a Short object from a string representation of a short value.
2. **Value Retrieval**:
   - shortValue(): Returns the short value of the Short object.
3. **Converting Shorts**:
   - toString(): Converts the Short object to a string representation.
   - toHexString(short value): Converts the specified short value to a hexadecimal string representation.
   - toOctalString(short value): Converts the specified short value to an octal string representation.
   - toBinaryString(short value): Converts the specified short value to a binary string representation.
4. **Comparison and Equality**:
   - equals(Object obj): Compares this Short object to the specified object for equality.
   - compareTo(Short anotherShort): Compares two Short objects numerically.

## 4. Integer wrapper class:

The Integer wrapper class in Java is a class that provides an object representation of the int primitive type. It allows you to work with int values as objects and provides various methods for performing operations on int values. The Integer class is part of the java.lang package and provides functionalities such as value manipulation, conversion, comparison, parsing, and more.
Some methods are:

1. **Creating Integer Objects:**
   - Integer(int value): Constructs an Integer object with the specified int value.
   - Integer(String value): Constructs an Integer object from a string representation of an int value.
2. **Value Retrieval:**
   - intValue(): Returns the int value of the Integer object.
3. **Converting Integers:**
   - toString(): Converts the Integer object to a string representation.

- toHexString(int value): Converts the specified int value to a hexadecimal string representation.
- toOctalString(int value): Converts the specified int value to an octal string representation.
- toBinaryString(int value): Converts the specified int value to a binary string representation.

4. **Comparison and Equality:**
   - equals(Object obj): Compares this Integer object to the specified object for equality.
   - compareTo(Integer anotherInteger): Compares two Integer objects numerically.

## 5. Long Wrapper Class:

The Long wrapper class in Java provides an object representation of the long primitive type. It allows you to work with long values as objects and provides various methods for performing operations on long values.

Here Some methods:

1. **Creating Long Objects:**
   - Long(long value): Constructs a Long object with the specified long value.
   - Long(String value): Constructs a Long object from a string representation of a long value.
2. **Value Retrieval:**
   - longValue(): Returns the long value of the Long object.
3. **Converting Longs:**
   - toString(): Converts the Long object to a string representation.
   - toHexString(long value): Converts the specified long value to a hexadecimal string representation.
   - toOctalString(long value): Converts the specified long value to an octal string representation.
   - toBinaryString(long value): Converts the specified long value to a binary string representation.
4. **Comparison and Equality:**
   - equals(Object obj): Compares this Long object to the specified object for equality.
   - compareTo(Long anotherLong): Compares two Long objects numerically.

## 6. Float Wrapper Class:

The Float wrapper class in Java provides an object representation of the float primitive type. It allows you to work with float values as objects and provides various methods for performing operations on float values.

Here are the key aspects and functionalities of the Float class:

1. **Creating Float Objects:**
   - Float(float value): Constructs a Float object with the specified float value.
   - Float(String value): Constructs a Float object from a string representation of a float value.

2. **Value Retrieval:**
    - floatValue(): Returns the float value of the Float object.
3. **Converting Floats:**
    - toString(): Converts the Float object to a string representation.
    - toHexString(float value): Converts the specified float value to a hexadecimal string representation.
4. **Comparison and Equality:**
    - equals(Object obj): Compares this Float object to the specified object for equality.
    - compareTo(Float anotherFloat): Compares two Float objects numerically.
5. **Parsing and Conversion:**
    - parseFloat(String s): Parses the string argument as a float value and returns the corresponding float value.
    - intValue(): Returns the float value as an int.
    - longValue(): Returns the float value as a long.
    - doubleValue(): Returns the float value as a double.
6. **Constants:**
    - MIN_VALUE and MAX_VALUE: Constants representing the minimum and maximum positive finite values of the float type.
    - POSITIVE_INFINITY, NEGATIVE_INFINITY, and NaN: Constants representing positive infinity, negative infinity, and not-a-number (NaN) values in float.

## 7. Double Wrapper Class:

The Double wrapper class in Java provides an object representation of the double primitive type. It allows you to work with double values as objects and provides various methods for performing operations on double values.
Here some methods are:

1. **Creating Double Objects**:
    - Double(double value): Constructs a Double object with the specified double value.
    - Double(String value): Constructs a Double object from a string representation of a double value.
2. **Value Retrieval**:
    - doubleValue(): Returns the double value of the Double object.
3. **Converting Doubles**:
    - toString(): Converts the Double object to a string representation.
    - toHexString(double value): Converts the specified double value to a hexadecimal string representation.

4. **Comparison and Equality**:
    - equals(Object obj): Compares this Double object to the specified object for equality.
    - compareTo(Double anotherDouble): Compares two Double objects numerically.
5. **Parsing and Conversion**:
    - parseDouble(String s): Parses the string argument as a double value and returns the corresponding double value.
    - intValue(): Returns the double value as an int.
    - longValue(): Returns the double value as a long.
    - floatValue(): Returns the double value as a float.

## 8. Boolean Wrapper Class:

The Boolean wrapper class in Java provides an object representation of the boolean primitive type. It allows you to work with boolean values as objects and provides various methods for performing operations on boolean values.

Here are the key aspects and functionalities of the Boolean class:

1. **Creating Boolean Objects:**
    - Boolean(boolean value): Constructs a Boolean object with the specified boolean value.
    - Boolean(String value): Constructs a Boolean object from a string representation of a boolean value.
2. **Value Retrieval:**
    - booleanValue(): Returns the boolean value of the Boolean object.
3. **Converting Booleans:**
    - toString(): Converts the Boolean object to a string representation.
4. **Comparison and Equality:**
    - equals(Object obj): Compares this Boolean object to the specified object for equality.
    - compareTo(Boolean anotherBoolean): Compares two Boolean objects.
5. **Parsing and Conversion:**
    - parseBoolean(String s): Parses the string argument as a boolean value and returns the corresponding boolean value.
6. **Constants:**
    - TRUE and FALSE: Constants representing the true and false values of the boolean type.