WRAPPER CLASS IN JAVA

- Two concepts are there in the wrapper classes -
- Autoboxing
- Unboxing
- Autoboxing : It is an automatic method that converts primitive data type into an object of their corresponding wrapper class.

  Eg. int to Integer, double to Double, etc.
- Unboxing : It is just the reverse process of autoboxing. Converts object of a wrapper into its corresponding primitive data type.

  Eg. Integer to int, Float to float.

Need of Wrapper Class

- They convert primitive data types into objects. Objects are needed to modify the arguments passed into a method.
- The classes in java.util package handles only objects and hence wrapper classes help in this case also.
- It is useful in Hibernate because it doesn't support primitive data type.
- Data structures in the Collection framework, such as ArrayList and Vector, store only objects and not primitive types.

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

Practical code for conversion

- ```
  Character ch = 'a';
  char a = ch;    // unboxing - Character object to primitive
                  conversion
  ```

- ```
  Integer n = 28;
  int num = arrayList.get(0);  // unboxing because get method returns an Integer object
  ```

Strings

- Strings in java are like arrays, they are immutable (cannot grow) and hence for change, an entirely new string is created.
- Syntax : String name = " "; (String literal)

  String name = new String(" "); (dynamic allocation)
- String is created in String constant pool but when it created dynamically though new operator, they are assigned a new memory location in heap.
- StringBuilder and StringBuffer can be used for mutable sequences as String literals are immutable.

Some String methods

- .length(): Returns the number of characters in the String
- .charAt(i): Returns the character at i^{th} index
- .substring(i): Return the substring from the i^{th} index character to end
- .substring(i,j): Returns the substring from i to j-1 index.
- .concat(String): Concatenates specified string to the end of this string
- .indexOf(string): Returns the index within the string of the first occurrence of the specified string
- .lastIndexOf(String): Returns the index within the string of the last occurrence of the specified string.

Syntax for String methods

- `String s= "HelloJava"; // or String s= new String ("HelloJava");`
- `s.length()); // Returns the number of characters in the String.`
- `s.charAt(3)); // Returns the character at ith index.`
- `s.substring(3)); // Return the substring from the ith index character to end of string`
- `s.substring(2,5)); // substring from i to j-1 index.`

- ```
  String s1 = "Hello";
  String s2 = "Java";
  s1.concat(s2));  // Concatenates string2 to the end of string1.
  ```

- ```
  String s4 = "Learn java from";
  s4.indexOf("GFG"));  // Returns the index within the string of the first occurrence of the specified string.
  ```

- ```
  s4.indexOf('a',3));  // Returns the index within the string of the
                       // first occurrence of the specified string,
                       // starting at the specified index.
  ```

- ```
  "Neha".equals("neha");
   "Neha".equalsIgnoreCase("nEhA ");    // Checking equality of Strings
  ```

- ```
  s1.compareTo(s2);   // If ASCII difference is zero
  then the two strings are similar
  ```

- ```
  String word1 = "HelloMam";
   word1.toLowerCase());  // Converting cases
  ```

- ```
  String word2 = "Java Training";
  word2.toUpperCase());  // Converting cases
  ```

- ```
  String word4 = " Learn Java from Monica Gupta ";
  word4.trim());   // Trimming the word
  ```

- ```
  String str1 = "Hello lava World";
  str1.replace('l','j') ;  // Replacing characters
  ```
  ```