Java™

Wrapper class & Strings

Wrapper class

Introduction

- Wrapper classes provide a way of wrapping primitive values into objects, allowing primitives to be treated as objects.
- They provide a set of utility methods to manipulate and convert primitive data.
- All wrapper classes are part of the **java.lang** package, which is automatically imported.
- Direct type casting is not possible between different wrapper classes.
- Wrapper classes are immutable, meaning their values cannot be changed once created.
- Wrapper classes are mainly used when object representation of primitive data is required.
- All the primitive data types have their corresponding Wrapper classes. Wrapper classes are:
  - char - Character
  - byte - Byte
  - short - Short
  - long - Long
  - float - Float
  - Double - Double
  - Boolean - Boolean
  - Int - Integer

Situations when required

- When primitives are required to be added to a collection object.
- When we want to return a primitive from a method that returns an object.
- When converting primitives to and from String objects.
- When converting primitives & String objects to and from different bases like binary, octal & hexadecimal.

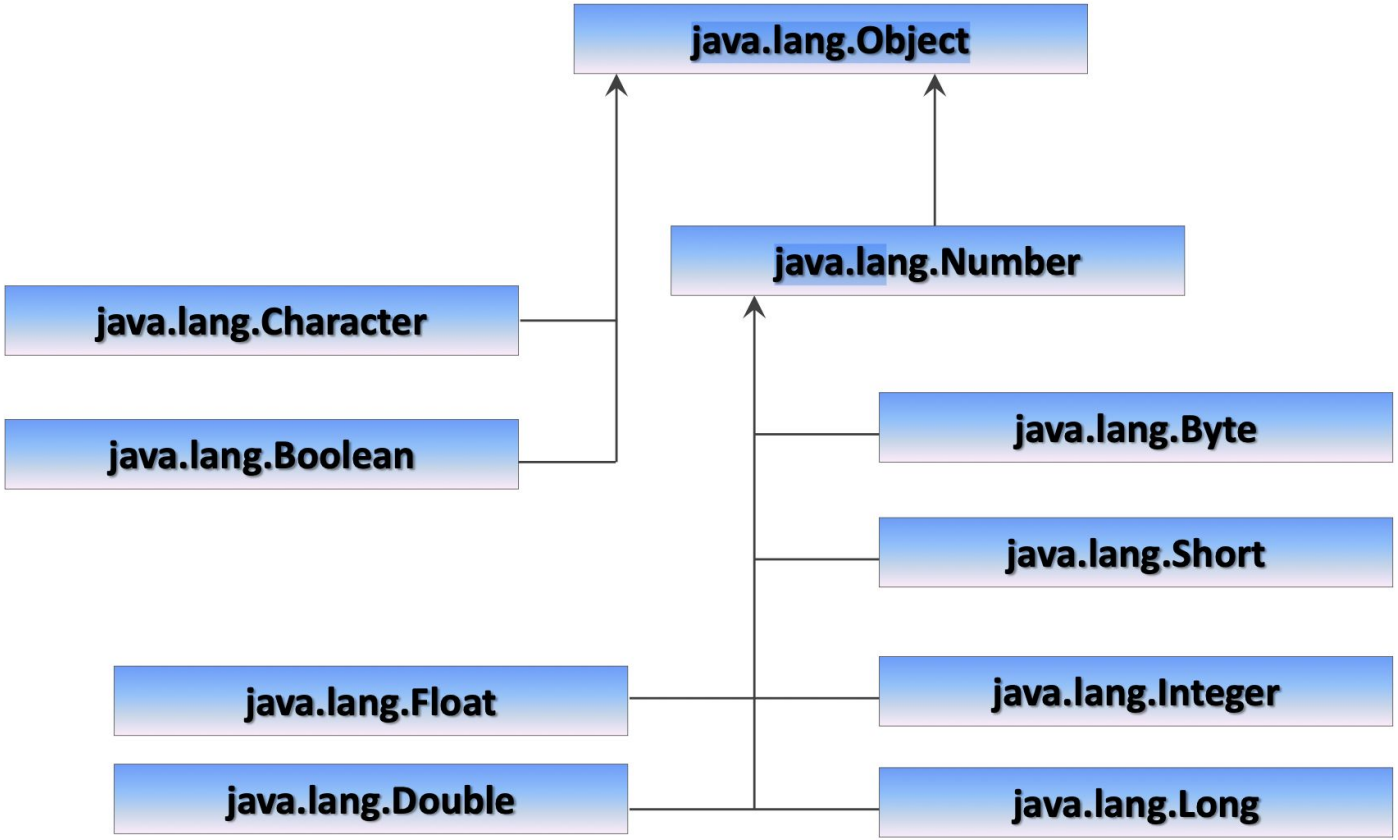
Wrapper Classes

Primitive	Wrapper Class	Constructor Arguments
boolean	Boolean	boolean or String
byte	Byte	byte or String
char	Character	char
double	Double	double or String
float	Float	float or String
int	Integer	int or String
long	Long	long or String
short	Short	short or String

Creating Objects

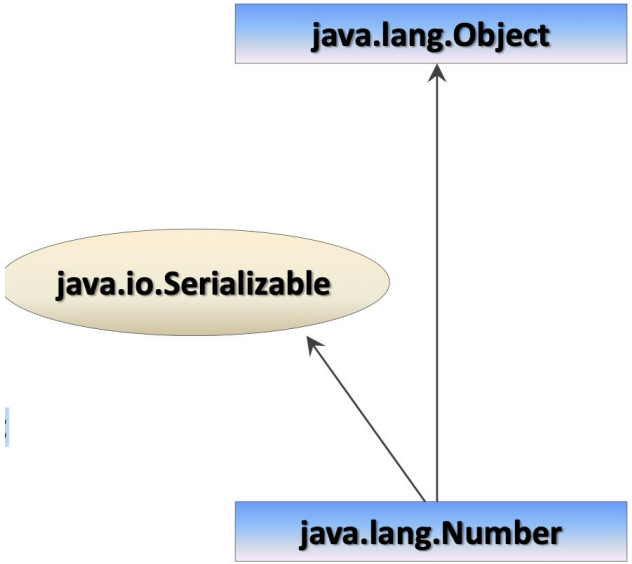
- Integer i1 = new Integer(5);
- Integer i2 = new Integer(“5”);
- Float f1 = new Float(5.6f);
- Float f2 = new Float(“5.6f”);
- Double d1 = new Double(6.7);
- Double d2 = new Double(“6.7”);
- Character c = new Character(‘a’);
- Boolean b1 = new Boolean(true);
- Boolean b2 = new Boolean(“true”);
- Boolean b3 = new Boolean(“TrUe”);

Class Hierarchy



java.lang.Number

- public java.lang.Number();
- public abstract int intValue();
- public abstract long longValue();
- public abstract float floatValue();
- public abstract double doubleValue();
- public byte byteValue();
- public short shortValue();



AutoBoxing & AutoUnboxing

- Integer i = 67;
  Automatically translated into following code by the compiler.
  Integer i = new Integer(67); // so called **AutoBoxing**.

- int a = i + 10;
  Automatically translated into following code by the compiler.
  int a = i.intValue() + 10; // so called **AutoUnboxing.**

Object Creation Methods

Method	Description	Example
`valueOf(String s)`	Converts a String to a wrapper object.	`Integer i = Integer.valueOf("123");`
`valueOf(primitive p)`	Converts a primitive to a wrapper object.	`Double d = Double.valueOf(45.67);`

Conversion Methods

Method	Description	Example
`byteValue()`	Converts to byte.	`byte b = i.byteValue();`
`shortValue()`	Converts to short.	`short s = i.shortValue();`
`intValue()`	Converts to int.	`int num = d.intValue();`
`longValue()`	Converts to long.	`long l = i.longValue();`
`floatValue()`	Converts to float.	`float f = d.floatValue();`
`doubleValue()`	Converts to double.	`double d = i.doubleValue();`

Parsing Methods

Method	Description	Example
`parseInt(String s)`	Parses String to int.	`int n = Integer.parseInt("123");`
`parseDouble(String s)`	Parses String to double.	`double d = Double.parseDouble("45.67");`
`parseBoolean(String s)`	Parses String to boolean.	`boolean b = Boolean.parseBoolean("true");`

```
Integer x = 100;
Integer y = 100;
System.out.println(x == y);
x = 128;
y = 128;
System.out.println(x == y);
```

Character Class Methods

Method	Description	Example
`isDigit(char ch)`	Checks if the character is a digit.	`boolean result = Character.isDigit('5');`
`isLetter(char ch)`	Checks if the character is a letter.	`boolean result = Character.isLetter('A');`
`isUpperCase(char ch)`	Checks if the character is uppercase.	`boolean result = Character.isUpperCase('A');`
`isLowerCase(char ch)`	Checks if the character is lowercase.	`boolean result = Character.isLowerCase('a');`
`toUpperCase(char ch)`	Converts a character to uppercase.	`char upper = Character.toUpperCase('a');`
`toLowerCase(char ch)`	Converts a character to lowercase.	`char lower = Character.toLowerCase('A');`

Utility Methods

Method	Description	Example
`toString()`	Converts value to String.	`String s = i.toString();`
`compareTo(Wrapper w)`	Compares two wrapper objects.	`int result = i1.compareTo(i2);`
`equals(Object obj)`	Compares the wrapper object to another object.	`boolean isEqual = i.equals(100);`
`hashCode()`	Returns the hash code of the object.	`int hash = i.hashCode();`
`compare(x, y)`	Compares two primitive values.	`int result = Integer.compare(10, 20);`
`max(x, y)`	Returns the maximum of two values.	`int maxVal = Integer.max(10, 20);`
`min(x, y)`	Returns the minimum of two values.	`int minVal = Integer.min(10, 20);`

String

- A String in Java is a sequence/array of characters.
- Strings are represented by the String class, which is present in the java.lang package.
- In Java, String objects are immutable, meaning once a String is created, its value cannot be changed.
- Java provides special support for Strings, allowing them to be created using double quotes (" ").
- Strings are widely used for text processing, user input, file handling, and data exchange.

Syntax: <String_Type> <string_variable> = “<sequence_of_string>”;

- There are two ways to create a string:

1. String literal

```
String customerName = "CDAC Patna";
```

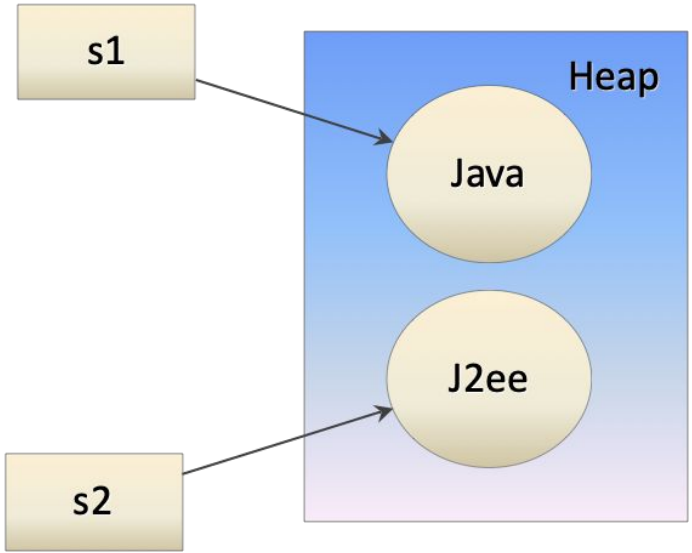
2. Using new() keyword

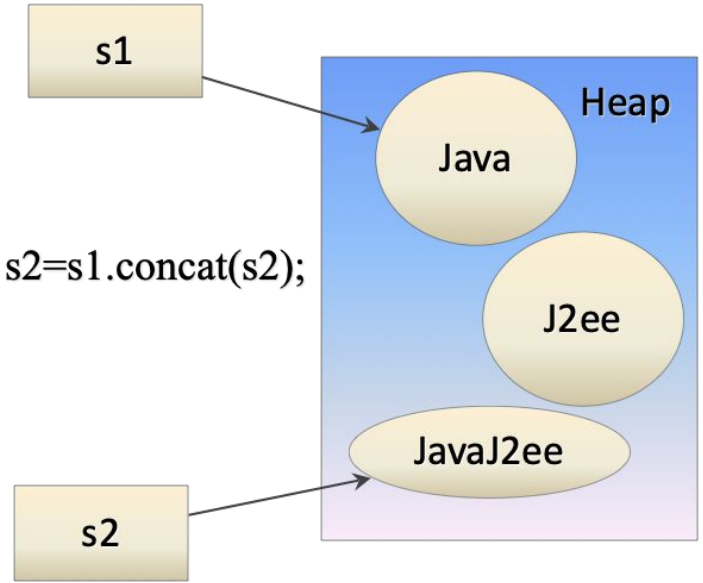
```
String customerName = new String("CDAC Patna");
```

Understanding the Immutability

String s1 = new String("Java");

String s2 = new String("J2ee");





Creation of String in Java

1. String str;

   str=”Java Programming”;

2. String str=”Java Programming”;

3. String str=new String(“Java Programming”);

4. char[] charArray={‘J’, ‘a’, ‘v’, ‘a’, ‘ ‘, ‘P’, ‘r’, ’o’, ’g’, ’r’, ’a’, ’m’, ’m’, ’i’, ‘n’, ’g’};

   String str2=new String(charArray);

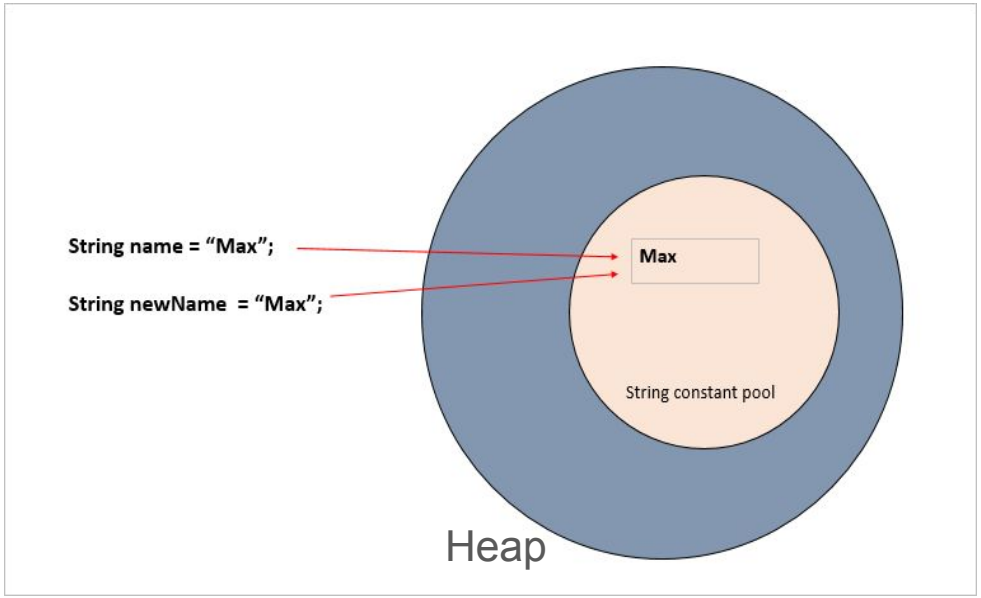
5. String str2=new String(charArray, 0, 4);

   Starting from 0th position 4 characters from the char[]

Memory allotment of String:

String literal:

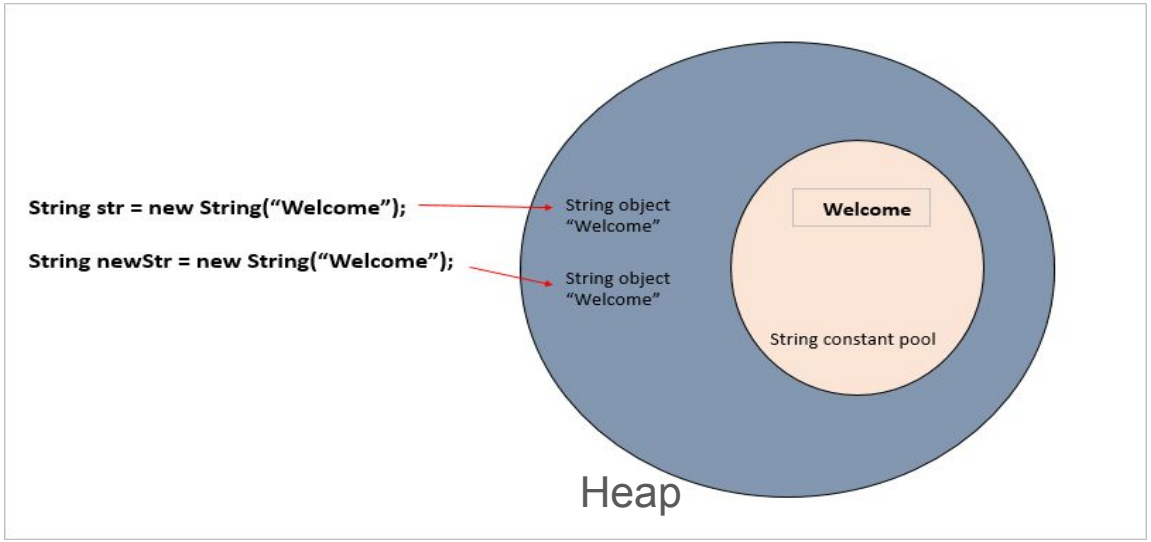
Every time you create a string literal, the JVM checks the **String constant pool**. String constant pool in Java is a pool of Strings stored in Heap memory. If the string exists in the pool, then a reference to the existing literal is returned. If the string is not found, then a new instance is created and placed in the pool.



Memory allotment of String contd.

String Object:

Strings behave a bit differently when a new instance of String class is created. When you create a string using the new() keyword, JVM places the literal in the constant pool and also creates a new string object in heap memory. The reference variable points to the object in the heap memory. the string pool is reserved for compile-time constants, not runtime results.



Comparing Strings

```
String str1 = “Java Program”;

String str2  = “Java Program”;

String str3 = new String(“Java Program”);

String str4 = new String(“Java Program”);

System.out.println(str1==str2);

System.out.println(str1.equals(str2));

System.out.println(str3 == str4));

System.out.println(str3.equals(str4));
```

```
str1==str3

str1.equals(str3)

String word2 =  “Program”;

System.out.println(str1 == "Java Program");

System.out.println(str1 == ("Java "+"Program"));

System.out.println(str1 == ("Java  "+word2));
```

TryOut: Strings Methods

String Method	Description
int length()	returns number of characters in a string
String concat(String s)	concatenates or joins two strings and returns third string as the result
boolean equals(String s)	checks case sensitive equality of the string
boolean equalsIgnoreCase(String s)	checks case insensitive equality of the string
String toLowerCase()	returns a string that contains all the characters of the source string converted to lower case
String toUpperCase()	returns a string that contains all the characters of the source string converted to upper case
char charAt(int index)	returns a char value at the given index
String substring(int beginIndex, [endIndex])	returns substring from beginIndex to endIndex and if endIndex is not mentioned then returns substring from beginIndex to end of string
boolean contains(CharSequence s)	returns true if the character sequence is present in the string else returns false
String replace(char old, char new)	replaces all the occurrences of the specified character with the new character

StringBuffer

- StringBuffer object is mutable i.e. it represents a String that can be changed without creating a new object.
- StringBuffer's capacity can be dynamically increased even though its initial capacity is specified.
- The API's of StringBuffer is synchronized
- StringBuffer is a final class and it is available in **java.lang package**
- Some of the methods of StringBuffer class are **append(), insert(), delete(), reverse()**.

```
StringBuffer sb = new StringBuffer();
```

TryOut: StringBuffer Methods

- append()
- insert()
- delete()
- reverse()
- replace()

StringBuilder

- StringBuilder is the **non-Synchronized** version of the StringBuffer class.
- It is used when we do not require **thread-safety.**
- String and StringBuffer classes are slower than StringBuilder
- It contains all the methods of the StringBuffer class
- The StringBuilder in Java provides a mutable sequence of characters. The StringBuilder is used as an alternative to the String class in Java because of its mutability.
- Some of the methods of StringBuilder class are **append(), insert(), delete(), reverse()**.

```
StringBuilder sb = new StringBuilder();
```

Constructors for StringBuilder:

- StringBuilder( ) - This constructs an empty string builder and with an initial capacity of 16 characters.
- StringBuilder(int capacity) - This constructs an empty string builder and with an initial capacity specified by the capacity argument.
- StringBuilder(String str) - This constructs a string builder and with the contents of the specified string.

TryOut: StringBuilder Methods

Method	Description
StringBuilder append(String str)	appends the specified string with the string
StringBuilder insert(int offset, String str)	inserts the given string at the specified position
StringBuilder reverse()	reverses the given string
char charAt(int index)	returns the character present at the specified position
StringBuilder delete(int start, int end)	deletes the string starting from start to one less than the end

Assignments

1. Write a Java program to concatenate a given string to the end of another string.
    a. Sample Output:
    b. String 1: This is STT
    c. String 2: Java class
    d. The concatenated string: This is STT Java class.
2. Write a Java program to compare a given string to another string, ignoring case considerations.
    a. Sample Output:
    b. "CDAC Patna" equals "Cdac Patna"? True
3. Write a Java program to replace the substring "Java" with "Python" in the string "Java is cool" using StringBuilder.
4. Write a Java program to convert all the characters in a string to Lowercase.
5. Write a Java program to convert string "123" into an Integer Class Object using Wrapper class.
6. Write a program using the Character wrapper class to: A. Check if a character is a digit. B. Convert a character to uppercase. C. Check if a character is a letter.

Exercise

➔ Remove all the white spaces from the string passed to the method and return the modified string.

Sample Input	Expected Output
"Hello How are you "	HelloHowareyou
"J ava pro gramming"	Javaprogramming

Thank You