Wrapper Classes

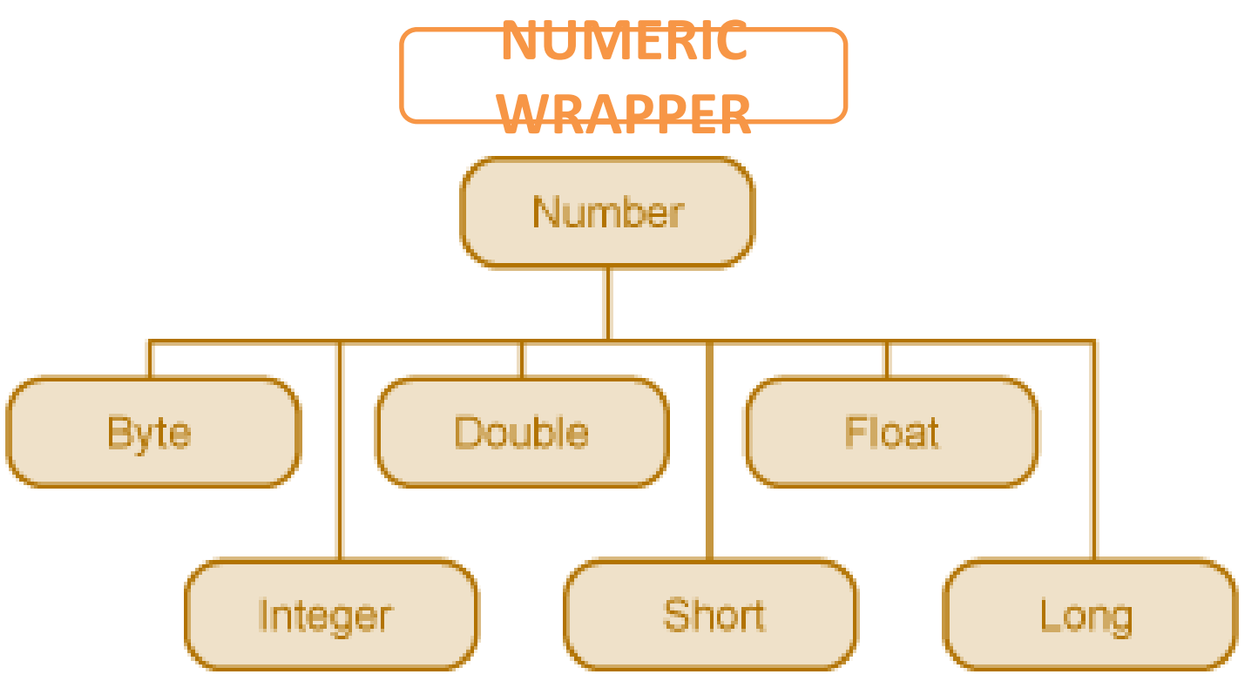
What are Wrapper Classes

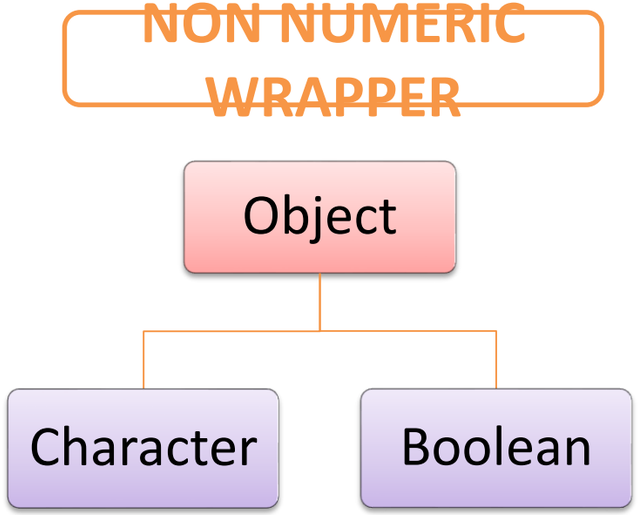
- What
  - Wraps a value of the primitive type in an object

- Why
  - Primitives can be included in activities reserved for objects like collections
  - To provide utility functions for primitives like
    - Converting primitives to and from String objects
  - To use constants MIN_VALUE and MAX_VALUE, that provide the upper and lower bounds of the data type

- Java provides wrapper classes for most of the primitive data types

Hierarchy of Wrapper Classes





- All of the numeric wrapper classes are sub classes of the abstract class Number

Constructors of Wrapper classes

```
//Character class provides one constructor
Character c = new Character('n');


//All other wrapper classes have overloaded //Constructors
Double d = new Double(2346.23);
Double d1 = new Double(“2346.23”);

Boolean b = new Boolean(true);
Boolean b = new Boolean("false");

Integer i1 = new Integer(“10”);
Integer i2 = new Integer(10);
```

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

Boxing and unBoxing

- Boxing
  - Automatic conversion of primitive datatypes in to its equivalent wrapper type

```
Integer i = 10;  //Boxing
```

- UnBoxing
  - Automatic conversion of wrapper in to its equivalent primitive data type

```
Integer i1 = new Integer("10");
int i2 = i1; //UnBoxing
```

- From java version 5 , Compiler does the boxing and unboxing automatically

Methods of Wrapper Classes

- parsexxx(..) - static method
  - parseInt(), parseLong(), ParseDouble() etc
  - Static method to convert String objects into the named primitive
  - Takes String as parameter and returns the primitive
  - Throws a NumberFormatException, if the String argument is not properly formed

```
//Parsing String to int
String s1 = "225";
int i = Integer.parseInt(s1);

//Parsing String to float
String s2 = "225.25";
float f1 = Float.parseFloat(s2);
```

Applicable to all wrapper classes except Character

Methods of Wrapper Classes

- valueOf(..) - static method
  - Converts Strings in to the underlying wrapped object
  - Wraps primitives into the corresponding wrapped object
  - Returns a wrapped object

```
      Integer i =
Integer.valueOf(“175”);
      Integer i =
Integer.valueOf(175);

      Double d1 =
Double.valueOf(“175.75”);
      Short  s1 =
Short.valueOf(“200”);
```

Methods of Wrapper Classes

- xxxValue() – non static method
  - Returns the numeric value in named primitive represented by the Wrapper object

```
Integer num = new Integer("10");
byte bNum = num.byteValue();
short sNum = num.shortValue();
```

- Assigning one numeric wrapper object to another

```
Integer num1 = new Integer("10");

Long num2 = num1;        // Compilation
Error

Long num2 = num1.longValue();
```

Methods of Wrapper Classes

- toString() – non static method
  - Returns a String representing the value represented by the Wrapper object

```
Integer num = new Integer("10");
String str = num.toString();
```

- toString() – static method
  - Returns a String representing the value represented by the Wrapper object or primitive

```
int num = 5;
String str = Integer.toString(num);
```

Methods of Wrapper Classes

- toHexString() – static method
  - Returns a string representing the hexadecimal value of the named primitive
  - Applicable to int, long, float and double

```
String str =
Integer.toHexString(225);
// value of str will be “e1”
```

- toBinaryString() – static method
  - Returns a string representing the hexadecimal value of the named primitive
  - Applicable to int and long

```
String str = Integer.toHexString(6);
// value of str will be “110”
```

Important methods of Character class

- Below are some important static methods of Character class
  - isUpperCase(ch)
  - isLowerCase(ch)
  - isDigit(ch)
  - isLetter(ch)
  - isWhitespace(ch)
  - toUpperCase(ch)
  - toLowerCase(ch)

```
Character.isUpperCase('A'); //
true

Character.toUpperCase('a'); //
A

Character.isDigit('1');
// true

Character.isLetter('1');
// false
```

Boxing and UnBoxing

- In pre–Java 5 version, to increment a Integer wrapper object

Unbox

```
Integer i = new Integer(222);        // wrapper
int x = i.intValue();          // unwrap it
```

Increment

```
x++;                                 // process it
```

Box

```
i = new Integer(x);            // re-wrap it
System.out.println(i);               // print it
```

AutoBoxing

- In Java 5 and above

```
Integer i = new Integer(222);    // wrapper
i++;                             // Auto box
```

- The code appears to be using the post increment operator on an object reference variable

- Behind the scenes, the compiler does the unboxing and reassignment