Objectives of wrapper classes:

- To wrap primitive in object form so that we can handle *primitives* as well just like objects.
- To define several utility methods which are required for *primitves*.

Constructors:

- Almost all wrapper classes contain two constructors, one can take corresponding *primitive* as argument and the other can take `String` as argument.

Example:

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

Double D = new Double(10.5);
Double D = new Double("10.5");
```

- If `String` argument doesn't represent a number then we will get `runtime exception` saying `NumberFormatException`.

Example:

```
Integer I = new Integer("ten");  => runtime exception:
NumberFormatException
```

- `Float` class contains three constructors with `float`, `double` and `String` arguments.

Example:

```
Float D = new Float(10.5f);
Float D = new Float("10.5f");
Float D = new Float(10.5);
Float D = new Float("10.5");
```

- `Character` class contains only one constructor which can take a `char` argument.

Example:

```

```

```
Character ch = new Character('a');  => invalid
Character ch = new Character("a");  => invalid
```

- `Boolean` class contains two constructors, one can take `primitive` as argument and the other can take `String` argument. If we pass boolean primitive as agrument, the only allowed values are: `true` or `false` where case and content are important.
- If we are passing `String` type as argument then case and content both are not important.
- If content is case insensitive `true` then it is treated as `true` otherwise it's treated as `false`.
  Note:
  - These constructors are ***deprecated***.

Example:

```
Boolean B = new Boolean(true);      => valid
Boolean B = new Boolean(false);     => valid
Boolean B = new Boolean(True);      => invalid
Boolean B = new Boolean(Ahmed);     => invalid
Boolean B = new Boolean("true");    => true
Boolean B = new Boolean("True");    => true
Boolean B = new Boolean("TRUE");    => true
Boolean B = new Boolean("Ahmed");   => false
Boolean B = new Boolean("AHMED");   => false
```

Wrapper classs	Corresponding constructor arguments
`Byte`	`byte` or `String`
`Short`	`short` or `String`
`Integer`	`int` or `String`
`Long`	`long` or `String`
`Float`	`Float` or `String` or `double`
`Double`	`double` or `String`
`Character`	`char` or ~~`String`~~
`Boolean`	`boolean` or `String` [***deprecated***]

Note: - In all wrapper classes `toString()` method is *overriden* to return content directly. - In all wrapper classes `equals()` method is *overrriden* for content comparison.

Utility methods:

- `valueOf()`
- `xxxValue()`

- `parseXxx()`
- `toString()`

`valueOf()`:

- We can use `valueOf()` method to create wrapper object for the given `primitive` or `String`.

Form 1:

- Every wrapper class except `Character` class contains `static valueOf()` method to create wrapper object for the given `String`.
  `public static wrapper valueOf(String s);`

Example:

```
Integer I = new Integer.valueOf("10");
Double D = new Double.valueOf("10.5");
Boolean B = new Boolean.valueOf("Ahmed");
```

Form 2:

- Every integral type wrapper class [`Byte`, `Short`,`Integer`, `Long`] contains the following `valueOf()` method to create wrapper object for the given specified radix string.
  `public static wrapper valueOf(String s, int radix);`
- The allowed range of radix is *2* to *36*

Example:

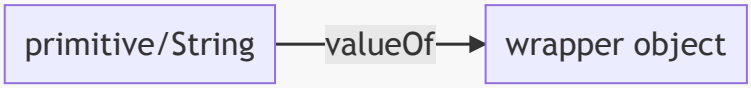
```
Integer I = new Integer.valueOf("100", 2);
System.out.println(I);                      => 4

Integer I = new Integer.valueOf("100", 4);
System.out.println(I);                      => 17

Integer I = new Integer.valueOf("100", 37); => runtime exception:
NumberFormant NumberFormatException: radix 37 greater than
Character.MAX_RADIX
```

Form 3:

- Every wrapper class including `Character` class contains a `static valueOf()` method to create wrapper object for the given `primitive`.
  `public static wrapper valueOf(primitive p);`

Example:

```
Integer I = new Integer.valueOf(10);
Character ch = new Character.valueOf('a');
Boolean B = new Boolean.valueOf(true);
```



xxxValue():

- We can use `xxxValue()` methods to get primitive for the given wrapper object.
- Every `Number` type wrapper class [`Byte`, `Short`, `Integer`, `Long`, `Float`, `Double`] contains the following six methods to get primitives for the given wrapper object.

```
public byte byteValue();
public short shortValue();
public int intValue();
public long longValue();
public float floatValue();
public double doubleValue();
```

Example:

```
Integer I = new Integer.valueOf(130);
System.out.println(I.byteValue());    =>  -126
System.out.println(I.shortValue());   =>  130
System.out.println(I.intValue());     =>  130
System.out.println(I.longValue());    =>  130
System.out.println(I.floatValue());   =>  130.0
System.out.println(I.doubleValue());  =>  130.0
```

- `Character` class contains `charValue()` method to get `char` primitive for the given `Character` object.

```
public char charValue();
```

Example:

```
Character ch = new Character.charValue('a');
char c = ch.charValue();
System.out.println(c);    => 'a'
```

- `Boolean` class contains `booleanValue()` method to get `boolean` primitive for the given `Boolean` object.

```
public boolean booleanValue();
```

Example:

```
Boolean B = new Boolean.charValue("Ahmed");
char b = B.booleanValue();
System.out.println(b);    => false
```

Note:

- In total *38 = (6 * 6 + 1 + 1)* `xxxValue()` are possible. 6 for the number one, 1 for character and one 1 for boolean.