```
JDK8-features
+++++++++++++
 1. Functional interface
 2. Lambda Expression,Method reference, Constructor reference
 3. default and static methods in interface
 4. Inbuilt functional interface
      a. Runnable
      b. Predicate
      c. Function
      d. Supplier
      e. Consumer
 5. Optional API
 6. StringJoiner
 7. JodaAPI



Wrapper classes
+++++++++++++++
 What is the need of Wrapper class, when we already had primitive types like
int,float,double,....
Ans. To wrap primitives into Object form, so that we can handle primitives also
just like Objects.
     To define several utility methods which are requried for primitives.

int a= 10;//primtive data
System.out.println(a);
System.out.println(a.toString());//CE

Constructors
===========
 Almost all the Wrapper class have 2 constructors
   a. one taking primitive type.
   b. one taking String type.

eg: Integer i=new Integer(10);
    Integer i=new Integer("10");

    Double d=new Double(10.5);
    Double d=new Double("10.5");

Note: If String argument is not properly defined then it would result in
RunTimeException called "NumberformatException".
      eg:: Integer i=new Integer("ten");//RE:NumberFormatException

Note: In wrapper class, toString() is overriden to print the content of the object
which is similar to "String" class.
@Override
public java.lang.String toString(){
      //override to print the data present in the object
}

Wrapper class and its associated constructor
   Byte   => byte and String
   Short  => short and String
  Integer => int and String
  Long    => long and String
  **Float => float ,String and double
   Double => double and String
 **Character=> character
 ***Boolean  => boolean and String

eg::
 1) Float f=new Float (10.5f);
 2) Float f=new Float ("10.5f");
 3) Float f=new Float(10.5);
 4) Float f=new Float ("10.5");

eg::
 1) Charcter c=new Character('a');
 2) Character c=new Character("a"); //invalid

eg::
Boolean b=new Boolean(true);
Boolean b=new Boolean(false);
Boolean b1=new Boolean(True);//C.E
Boolean b=new Boolean(False);//C.E
Boolean b=new Boolean(TRUE);//C.E

eg::
 Boolean b1=new Boolean("true");
 Boolean b2=new Boolean("True");
 Boolean b3=new Boolean("false");
 Boolean b4=new Boolean("False");
 Boolean b5=new Boolean("nitin");
 Boolean b6=new Boolean("TRUE");
 System.out.println(b1);//true
 System.out.println(b2);//true
 System.out.println(b3);//false
 System.out.println(b4);//false
 System.out.println(b5);//false
 System.out.println(b6);//true


eg::
Boolean b7 = new Boolean("yes");
Boolean b8 = new Boolean("no");

System.out.println(b7);//false
System.out.println(b8);//false
System.out.println(b7 == b8);//reference comparison    :: false
System.out.println(b7.equals(b8));//content comparison :: true

eg::
Integer i2 = new Integer(10);
System.out.println(i1);
System.out.println(i1.equals(i2));
Output
10
true
```

Note: In case of Boolean constructor, boolean value be treated as true w.r.t to case insensitive
      part of "true",for all others it would be treated as "false".

Note: If we are passing String argument then case is not important and content is not important.
      If the content is case insensitive String of true then it is treated as true

```
in all other cases it is treated as false.

Note: In case of Wrapper class,toString() is overriden to print the data.
      In case of Wrapper class,equals() is overriden to check the content.
      Just like String class, Wrapper classes are also treated as "Immutable
class".


Wrapper class utiltiy methods
=============================
1. valueOf() method.
2. XXXValue() method.
3. parseXxx() method.
4. toString() method.


valueOf() method
================
 To create a wrapper object from primitive type or String we use valueOf().
 It is alternative to constructor of Wrapper class, not suggestable to use.
 Every Wrapper class,except character class contain static valueOf() to create a
Wrapper Object.

eg#1.
Integer i=Integer.valueOf("10");
Double  d=Double.valueOf("10.5");
Boolean b=Boolean.valueOf("nitin");
   System.out.println(i);
   System.out.println(d);
   System.out.println(b);

eg#2.
     public static valueOf(String s,int radix)
                                 |=> binary : 2(0,1)
                                 |=> octal  : 8(0-7)
                                 |=> decimal : 10(0-9)
                                 |=> hexadecimal : 16(0-9,a,b,c,d,e,f)
                                 |=> base : 36(0-9,a-z)

Integer i1=Integer.valueOf("1111");
    System.out.println(i1);//1111
Integer i2=Integer.valueOf("1111",2);
    System.out.println(i2);//15
Integer i3=Integer.valueOf("ten");
    System.out.println(i3);//RE:NumberFormatException
Integer i4=Integer.valueOf("1111",37);
    System.out.println(i4);//RE:NumberFormatException

eg#3.
    public static valueOf(primitivetype x)

Integer i1=Integer.valueOf(10);
Double  d1=Double.valueOf(10.5);
Character c=Character.valueOf('a');
Boolean b=Boolean.valueOf(true);
   Primtive/String =>valueOf() => WrapperObject


2. xxxValue()
```

```
     We can use xxxValue() to get primitive type for the given Wrapper Object.
     These methods are a part of every Number type Object.
     (Byte,Short,Integer,Long,Float,Double) all these classes have these 6 methods
which is
      Written as shown below.

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

eg#1.
 Integer i=new Integer(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

3. charValue()
       Character class contains charValue() to get Char primitive for the given
Character
       Object.
      public char charValue()
eg#1.
  Character c=new Character('c');
  char ch= c.charValue();
  System.out.println(ch);

4. booleanValue()
         Boolean class contains booleanValue() to get boolean primitive for the
given boolean
         Object.
      public boolean booleanValue()

eg#1.
   Boolean b=new Boolean("nitin");
   boolean b1=b.booleanValue();
    System.out.println(b1);//false

In total xxxValue() are 36 in number.
 => xxxValue() => convert the Wrapper Object => primitive.


parseXXXX()
===========
 We use parseXXXX() to convert String object into primitive type.

form-1
======
public static primitive parseXXX(String s)
  Every wrapper class,except Character class has parseXXX() to convert String into
primitive type.
```

```
eg: int i=Integer.parseInt("10");
    double d =Double.parseInt("10.5");
    boolean b=Boolean.parseBoolean("true");
```

form-2
======

```
public static primitive parseXXXX(String s, int radix)
                                       |=> range is from 2 to 36
```

Every Integral type Wrapper class(Byte,Short,Integer,Long) contains the following parseXXXX()
to convert Specified radix String to primitive type.

```
eg: int i=Integer.parseInt("1111",2);
    System.out.println(i);//15
```

Note: String => parseXXX() => primitive type

toString()
=========

To convert the Wrapper Object or primitive to String.

Every Wrapper class contain toString()

form1
=====

```
 public String toString()
```

1. Every wrapper class (including Character class) contains the above toString()
   method to convert wrapper object to String.
2. It is the overriding version of Object class toString() method.
3. Whenever we are trying to print wrapper object reference internally this
   toString() method only executed

```
eg:  Integer i=Integer.valueOf("10");
     System.out.println(i);//internally it calls toString() and prints the Data.
```

form2
=====

```
   public static String toString(primitivetype)
```

1. Every wrapper class contains a static toString() method to convert primitive to String.

```
String s=Integer.toString(10);
                           |=> primitive type int.
```

```
eg:
 String s=Integer.toString(10);
 String s=Boolean.toString(true);
 String s=Character.toString('a');
```

form3
=====

Integer and Long classes contains the following static toString() method to convert the primitive to specified radix String form.

```
 public static String toString(primitive p,int radix)
                                    |=> 2 to 36

eg: String s=Integer.toString(15,2)
    System.out.println(s); // 1111
```

form4
=====
Integer and Long classes contains the following toXxxString() methods.

```
public static String toBinaryString(primitive p);
public static String toOctalString(primitive p);
public static String toHexString(primitive p);
```

Example:

```
class WrapperClassDemo {
   public static void main(String[] args) {
      String s1=Integer.toBinaryString(7);
      String s2=Integer.toOctalString(10);
      String s3=Integer.toHexString(20);
      String s4=Integer.toHexString(10);

      System.out.println(s1);//111
      System.out.println(s2);//12
      System.out.println(s3);//14
      System.out.println(s4);//a
  }
}
```

            refer diagram for images