Wrapper classes:

primitive type	Wrapper classes
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

1. All wrapper classes available in the java.lang package
2. All wrapper classes are final classes
3. You cant develop a subclass to any wrapper class.
4. All wrapper classes are subclass to Comparable interface.
5. Comparable objects are allowed to the sorting in the collection framework.
6. All wrapper classes are implementing Serializable interface
7. Serializable objects only elizible for Serialization
8. toString() method of Object class got overrided in All wrapper classes.
9. toString() method is executing while printing a reference variable or
   in the concatination operation
10. hashCode() and equals() methods of Object class got overrided in All wrapper classes.
11. hashCode() and equals() methods are used to identify duplicates based on the content
    in the collection api.
12. wrapper classes used for:
    1. boxing and unboxing operations
    2. primitive to String and String to primitive
13. all numeric classes are subclass to Number class.
14. one wrapper object cant be converted into another wrapper type.
15. there is no support to convert String to char type. no method to convert String to char
    in the Character wrapper class.
16. wrapper objects are immutable in some extent
17. valueOf() method is overloaded in all the wrapper classes except in the Character.

```
    valueOf(primitive  type)
    valueOf(String type)

    but in the Character
    only one method ==>   valueOf(primitive  type)

18. if alphabets or spl chars are available while converting a
string into numeric,
    we get a NumberFormatException


19. Anything can be converted into boolean. if the content is true
(immeterial of the case), then it returns true
    if content is other than true, then it returns false

20. incase of Float and Double, one spl character allowed in the
String. that is dot.

21. letter l or L not allowed in the String while converting into
long type.

22. byte plus byte results int type

23. double plus double results double type

24. letter f or F allowed in the String while converting into float
type

25. before JDK 1.5  ==> programmer only should do an explicit boxing
and un boxing
    from JDK 1.5  ==> compiler only doing an automatic boxing and un
boxing


26. preference for the auto operations by the compiler
    widening
    boxing
    upcasting
    var-arg

27. 1. var-arg ==> variable number of arguments.
    2. var-arg should be the last argument to the method
    3. maximum one var-arg allowed to any method.
    4. var-arg is an array by default.
    5. overloading a method with an array as an argument to one
method and var-arg as an argument to another method
       not possible.
```

```
public class M1 {
	public static void main(String[] args) {
//		Integer obj = Integer.valueOf(100); //		wrapping  or
boxing
//		int i = obj.intValue();				//	un-
wrapping or unboxing
//		System.out.println("done:" + i);
//		M1 ref = new M1();
//		System.out.println(ref);
//		System.out.println(obj);

//		Integer i1 = Integer.valueOf(100);
//		Integer i2 = Integer.valueOf(100);
//		System.out.println(i1 == i2);
//
//		Integer i3 = Integer.valueOf(10000);
//		Integer i4 = Integer.valueOf(10000);
//		System.out.println(i3 == i4);

//		Integer  obj1 = Integer.valueOf(10);
//		Integer  obj2 = Integer.valueOf("10");
//		System.out.println(obj1);
//		System.out.println(obj2);
//
//		Double d1 = Double.valueOf(1.4);
//		Double d2 = Double.valueOf("1.4");
//		System.out.println(d1);
//		System.out.println(d2);

		//Character.valueOf("")

//		Integer obj1 = Integer.valueOf("10");
//		System.out.println("------------");
//		Integer obj2 = Integer.valueOf("10Y");

//		Boolean b1 = Boolean.valueOf("true");
//		Boolean b2 = Boolean.valueOf("hello");
//
//		System.out.println(b1);
//		System.out.println(b2);
//
```

```
//		Boolean b1 = Boolean.valueOf("TRUE");
//		System.out.println(b1);

//		int i = Integer.parseInt("10");
//		double j = Double.parseDouble("1.5");
//		System.out.println(i + ", " + j);

//		long lon1 = Long.valueOf("123L");
//		long lon2 = Long.parseLong("123l");
//		
//		System.out.println(lon1);
//		System.out.println(lon2);

		//long lon1 = 123245367477885;
		long lon2 = 123245367477885L;
		long lon3 = 123245367477885l;
		//long lon4 = (long) 123245367477885;

//		float f1 = 1.5;
//		float f2 = 1.5f;
//		float f3 = 1.5F;
//		float f4 = (float) 1.5;
//		
//		byte b1 = 10, b2 = 30;
//		int b3 = b1 + b2;
//		System.out.println(b3);

//		double d1 = 1.5, d2 = 5.6;
//		
//		double i = d1 + d2;
//		System.out.println(i);

//		Float f1 = Float.valueOf("1.5F");
//		System.out.println(f1);
//		
//		float f2 = Float.parseFloat("1.4f");
//		System.out.println(f2);

//		int i = Integer.valueOf(1000);
//		Integer obj = i;
//		i = obj;
//		Integer obj1 = 80;

//		Integer obj1 = Integer.valueOf(100);
//		Integer obj2 = Integer.valueOf(200);
//		
//		obj1 = obj2;
//		
//		int i = obj1.intValue();
```

```
//
//          obj1 = Integer.valueOf(i);
//
//          i = obj2;
//
//          obj2 = i;

//          Boolean b1 = true;
//
//          if(b1)
//          {
//                System.out.println("done");
//          }

            byte var = 10;
            //test(var);
      }
//    static void test(byte b1)
//    {
//          System.out.println("byte");
//    }
//    static void test(int b1)
//    {
//          System.out.println("int");
//    }
//    static void test(Byte b1)
//    {
//          System.out.println("Byte");
//    }
      static void test(Integer obj)
      {
            System.out.println("Integer");
      }
//    static void test(Number obj)
//    {
//          System.out.println("Number");
//    }
//    static void test(Object obj)
//    {
//          System.out.println("Object");
//    }
//    static void test(byte ... bs)
//    {
//          System.out.println("byte ...");
//    }

}
```