Wrapper classes in java

Surya m

- What is wrapper class ?
- The wrapper class in Java helps to convert primitive Into object and object into Primitive.
- Autoboxing and unboxing feature Convert primitives Into objects and objects into Primitives automatically.
- The automatic conversion of Primitive into an object Is known as autoboxing and unboxing.

- Need of wrapper classes :
- Java is an object-oriented programming language, so We need to deal with objects many times like in Collections, Serialization, Synchronization, etc. let's See where we need to use the wrapper classes.
- Change the value in method:
- Java supports only call by value. So, if we pass a Primitive value, it will not change the original value. But, if we convert the primitive value in an object, it Will change the original value.
- Serialization :
- If we have a primitive value, we can convert it in Objects through the wrapper classes then perform Serialization.

- Synchronization:
- Java synchronization works with objects in Multithreading.
- java.util package:
- The java.util package provides the utility classes to Deal with objects.
- Collection framework:
- Java collection framework works with objects only. All classes of the collection framework ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque deal with objects.

boolean	Boolean
char	Char
byte	Byte
short	Short
int	Int
long	Long
float	Float
double	Double

- Autoboxing
- The automatic conversion of primitive data type Into its corresponding wrapper class is known as Autoboxing.

```
public class WrapperExample{

    public static void main(String args[]){
        int a=20;
        Integer i=Integer.valueOf(a); //converting int
into Integer explicitly
        Integer j=a; //autoboxing

        System.out.println(a+" "+i+" "+j);
    }
}
```

- Unboxing
- It is just the reverse process of autoboxing. Automatically converting an object of a wrapper Class to its corresponding primitive type.

```
public class Unboxing {

    public static void main(String[] args){

    Character ch = 'a';

    // unboxing - Character object to primitive
    char a = ch;
    System.out.println(a);
   }
}
```