Wrapper classes in Java

A Wrapper class is a class whose object wraps or contains primitive data types, or basically in laymen terms in java the wrapper class provides the appliance to convert primitive into object and vice-versa i.e. object into primitive.

v

PRIMITIVE DATA TYPE	WRAPPER CLASS
BYTE	Byte
SHORT	Short
INT	Integer
LONG	Long
FLOAT	Float
DOUBLE	Double
BOOLEAN	Boolean
DOUBLE	Character

7060888670
Mukulsingh3344@gmail.com
MUKUL SINGH

Use of Wrapper classes

They convert crude information types into objects. Articles are required on the off chance that we wish to change the contentions passed into a technique (since crude kinds are passed by esteem).

The classes in java.util bundle handles just articles and thus covering classes help for this situation too.

Information structures in the Collection system, for example, ArrayList and Vector, store just articles (reference types) and not crude sorts.

An item is expected to help synchronization in multithreading.

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: We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.

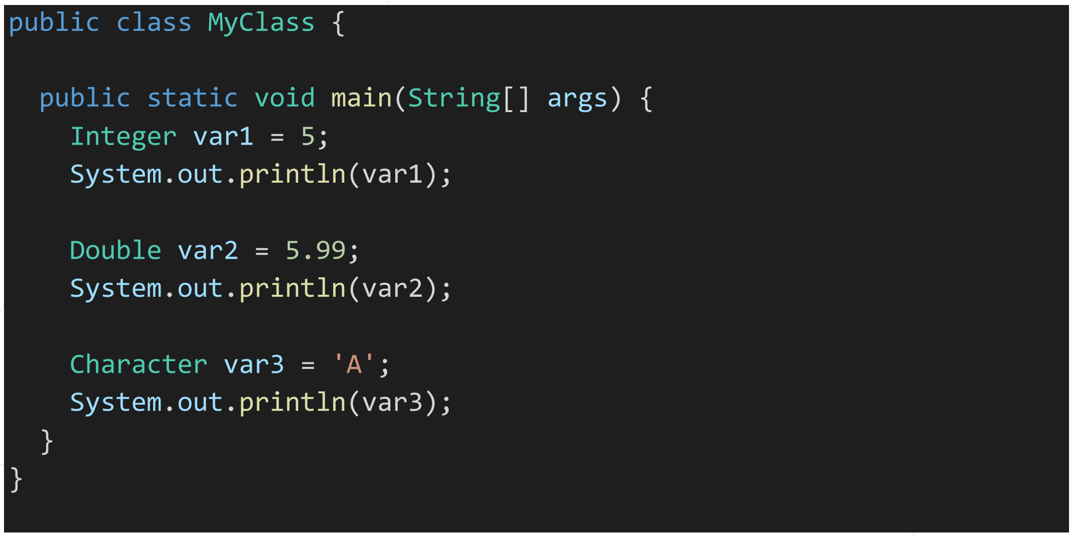
Synchronization: It works with objects in Multithreading.

java.util package: The java.util package provides the utility classes to deal with objects.

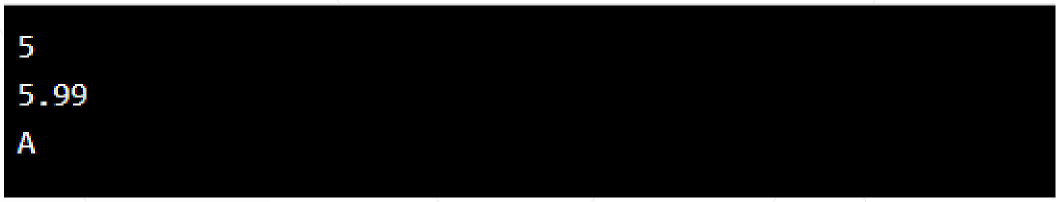
Collection Framework: Collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, Priority Queue, Array Deque, etc.) deal with objects only.

Creating Wrapper Objects

Example: -



```java
public class MyClass {

  public static void main(String[] args) {
    Integer var1 = 5;
    System.out.println(var1);

    Double var2 = 5.99;
    System.out.println(var2);

    Character var3 = 'A';
    System.out.println(var3);
  }
}
```



```
5
5.99
A
```

Since you're now working with objects, you can use certain methods to get information about the specific object.

For example, the following methods are used to get the value associated with the corresponding wrapper object: intValue(), byteValue(), shortValue(), longValue(), floatValue(), doubleValue(), charValue(), booleanValue().

```java
public class MyClass {

  public static void main(String[] args) {
    Integer var1 = 5;
    Double var2 = 5.99;
    Character var3 = 'A';
    System.out.println(var1.intValue());
    System.out.println(var2.doubleValue());
    System.out.println(var3.charValue());
  }
}
```

```
5
5.99
A
```

Another useful method is the toString() method, which is used to convert wrapper objects to strings.

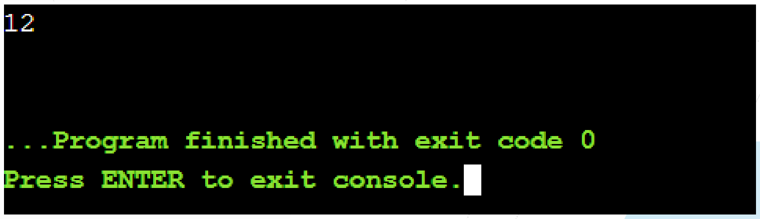
```java
public class MyClass {

  public static void main(String[] args) {
    Integer var1 = 100;
    String var2 = var1.toString();
    System.out.println(var2.length());
  }
}
```

```
3
```

Autoboxing and Unboxing

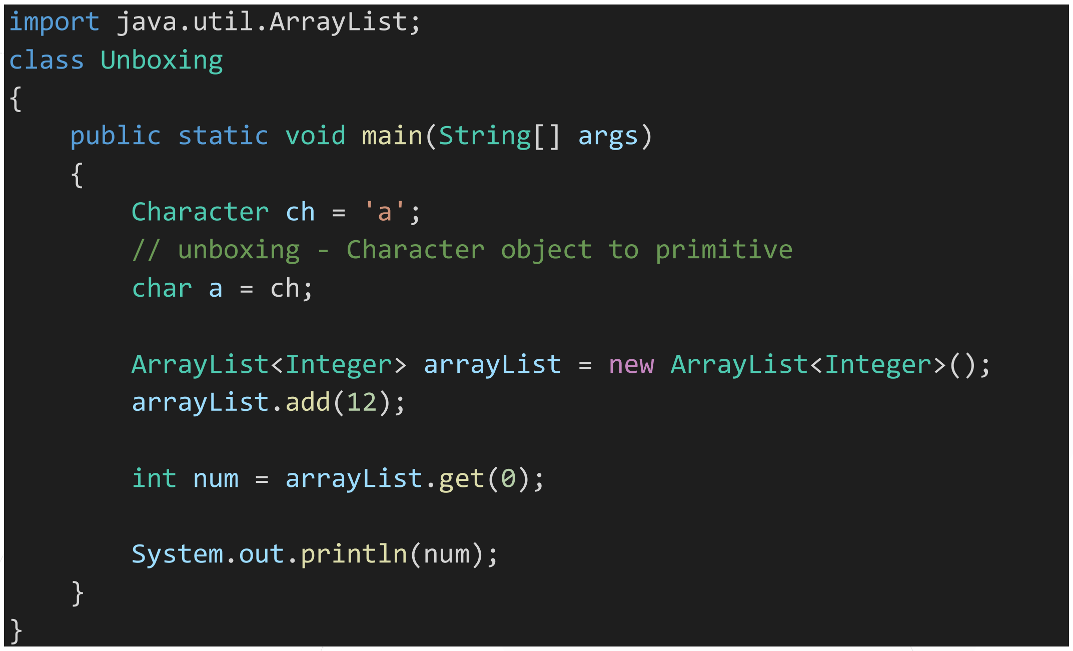
Autoboxing: Automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxing. For example – conversion of int to Integer, long to Long, double to Double etc.

```java
import java.util.ArrayList;
class Autoboxing
{
    public static void main(String[] args)
    {
        char ch = 'a';

        // Autoboxing- Character to object
        Character a = ch;

        ArrayList<Integer> arrayList = new ArrayList<Integer>();

        arrayList.add(12);

        System.out.println(arrayList.get(0));
    }
}
```



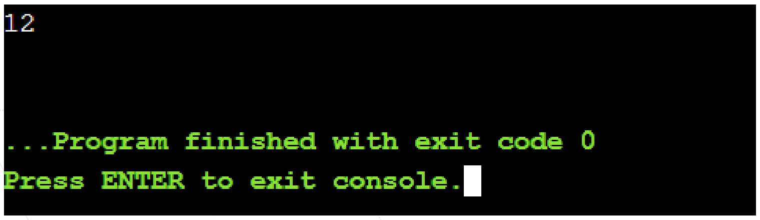
```
12


...Program finished with exit code 0
Press ENTER to exit console.
```

Unboxing: It is just the reverse process of autoboxing. Automatically converting an object of a wrapper class to its corresponding primitive type is known as unboxing. For example – conversion of Integer to int, Long to long, Double to double, etc.



```java
import java.util.ArrayList;
class Unboxing
{
    public static void main(String[] args)
    {
        Character ch = 'a';
        // unboxing - Character object to primitive
        char a = ch;

        ArrayList<Integer> arrayList = new ArrayList<Integer>();
        arrayList.add(12);

        int num = arrayList.get(0);

        System.out.println(num);
    }
}
```

v



```
12


...Program finished with exit code 0
Press ENTER to exit console.
```