BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Wrapper classes in Java.

Wrapper classes in Java:-

The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.

Since **J2SE 5.0**, **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 vice-versa unboxing.

Use of Wrapper classes in Java:-

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, 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: 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: 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, etc.) deal with objects only.

The eight classes of the java.lang package are known as wrapper classes in Java. The list of eight wrapper classes are given below:

Primitive Type	**Wrapper class**
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Java Boolean class:-

The Boolean class wraps a value of the primitive type boolean in an object. Its object contains only a single field whose type is boolean.

Example:-

```
public class JavaBooleanExample
{
    public static void main(String[] args)
   {
        Boolean b1= true;
        boolean b2=false;
        //assigning boolean value of b1 to b3
        Boolean b3= b1.booleanValue();
        String str1 = "Value of boolean object "+b1+" is "+b3+".";
        System.out.println(str1);
        //compare b1 and b2
           int val1 = Boolean.compare(b1,b2);
        if(val1>0)
        {
           System.out.println("b1 is true.");
         }
```

```
          else
         {
          System.out.println("b2 is true");
        }
        // logicalAnd() with return the same result as AND operator
        Boolean val2 = Boolean.logicalAnd(b1,b2);
        System.out.println("Logical And will return "+val2);
    }
}
```

Output:

```
Value of boolean object true is true.
b1 is true.
Logical And will return false
```

Java Character class

The Character class generally wraps the value of all the primitive type char into an object. Any object of the type Character may contain a single field whose type is char.

All the fields, methods, and constructors of the class Character are specified by the Unicode Data file which is particularly a part of Unicode Character Database and is maintained by the Unicode Consortium.

Example:-

```
import java.util.Scanner;
public class JavaCharacterExample
{
   public static void main(String[] args)
   {
    // Ask the user for the first input.
       System.out.print("Enter the first input:");
    // Use the Scanner class to get the user input.
       Scanner scanner = new Scanner(System.in);
    // Gets the user input.
       char[] value1 = scanner.nextLine().toCharArray();
       int result1 = 0;
    // Count the characters for a specific character.
       for (char ch1 : value1)
      {
       result1 = Character.charCount(ch1);
       }
    // Print the result.
        System.out.print("The value comes to: "+result1+"\n");
```

```
 System.out.print("Enter the second input:");
 char[] value2 = scanner.nextLine().toCharArray();
  for (char ch2 : value2)
  {
  int result2 = Character.hashCode(ch2);
 System.out.print("The hash code for the character
'"+ch2+"' is given as:"+result2+"\n");
 }
System.out.print("Enter the third input:");
 char[] value3 = scanner.nextLine().toCharArray();
 for (char ch3 : value3)
{
boolean result3 = Character.isDigit(ch3);
if(result3)
  {
System.out.println("The character '" + ch3 + "' is a digit. ");
   }
else
{
System.out.println("The character '" + ch3 + "' is not a digit.");
}
```

```
      System.out.print("Enter the fourth input:");
       char[] value4 = scanner.nextLine().toCharArray();
       for (char ch4 : value4)
      {
      boolean result4 = Character.isISOControl(ch4);
      System.out.println("The fourth character '"+ch4+"' is an
ISO Control:"+result4);
        }
      }
  }
}
```

Output:

```
Enter the first input:89
The value comes to: 1
Enter the second input:J
The hash code for the character 'J' is given as:74
Enter the third input:5
The character '5' is a digit.
Enter the fourth input:h
The fourth character 'h' is an ISO Control:false
```

Java Byte class:-

The Byte class wraps a primitive byte type value in an object. Its object contains only a single field whose type is byte.

Example:-

```
public class JavaByteExample
{
    static int i=1;
    public static void main(String[] args)
     {
        Byte byte1 = 5 ;
        Byte byte2 = 67 ;
        //compares the two specified byte values
        int val = Byte.compare(byte1,byte2);
        if (val>0)
        {
            System.out.println(i++ + ". "+byte1 + " is greater than
" + byte2);
        }
        else
        {
            System.out.println(i++ + ". "+byte2 + " is greater than
" + byte1);
        }
        // hash code of byte value byte1
        int f1 = byte1.hashCode();
        System.out.println(i++ + ". "+"Hash code value of
"+byte1+ " is : "+f1);
       //returns the value of this Byte as a Float
        Float f2 = byte2.floatValue();
        System.out.println(i++ + ". "+"Float value of "+byte2+ "
is : "+f2);
        //returns the value of this Byte as a Float
        Integer f3 = byte2.intValue();
        System.out.println(i++ + ". "+"Integer value of "+byte2+ "
is : "+f3);
        //decodes a String into a Byte
        String str="123";
        Byte f4 = Byte.decode(str);
        System.out.println(i++ + ". "+"Decoded value of "+str+ "
is : "+f4);
    }
}
```

Output:-

```
1. 67 is greater than 5
2. Hash code value of 5 is : 5
3. Float value of 67 is : 67.0
4. Integer value of 67 is : 67
5. Decoded value of 123 is : 123
```

Java Short Class:-

The short class wraps a primitive short type value in an object. Its object contains only a single field whose type is short.

Example:-

```
public class JavaShortExample
{
    static int i=1;
    public static void main(String[] args)
     {
        Short short1 = 500 ;
        Short short2 = 12 ;
        Short short3=12;
       // It compares two Short objects numerically
```

```
        int val = short1.compareTo(short2);
        if (val>0)
        {
            System.out.println(i++ + ". "+short1 + " is greater than
" + short2);
        }
        else
        {
            System.out.println(i++ + ". "+short2 + " is greater than
" + short1);
        }
        //It is used check whether both short values are equal
or not.
        Boolean b1 = short3.equals(short2);
        if (b1)
        {
            System.out.println(i++ + ". "+short2 + " and " + short3
+" are equal.");
        }
        else
        {
            System.out.println(i++ + ". "+short1 + " and " + short2
+" are not equal.");
        }
        //returns the value of this Short as a long
        Long f3 = short2.longValue();
        System.out.println(i++ + ". "+"Long value of "+short2+ "
is : "+f3);
        //Returns a string representation of the Short?s object
        String f4 = short2.toString();
        System.out.println(i++ + ". "+"String value of "+short2+ "
is : "+f4);
       //It returns a double value for this Short object
        Double f5 = short1.doubleValue();
        System.out.println(i++ + ". "+"Double value of "+short1+
" is : "+f5);
    }
}
```

Output:-

```
1. 500 is greater than 12
2. 12 and 12 are equal.
3. Long value of 12 is : 12
4. String value of 12 is : 12
```

5. Double value of 500 is : 500.0

Java Integer Class:-

The Java Integer class comes under the Java.lang.Number package. This class wraps a value of the primitive type int in an object. An object of Integer class contains a single field of type int value.

Example:-

```
public class IntegerMinExample
{
   public static void main(String[] args)
    {
       // Get two integer numbers
         int a = 5485;
         int b = 3242;
         // print the smaller number between x and y
         System.out.println("Math.min(" + a + "," + b + ")=" +
Math.min(a, b));
   }
}
```

Output:

```
Math.min(5485,3242)=3242
```

Java Long class:-

The Long class generally wraps the primitive type long into an object. An object of Long class contains a field with the type Long.

Example:-

```
public class JavaLongExample

{

public static void main(String[] args)

{

    Long x1=67l;

    Long x2=98l;

    Long x3=6l;

    // Compares two long values.

    int b1 = Long.compare(x1, x2);

    if(b1==0)

    {

   System.out.println("Both '"+x1+"' and '"+ x2+"' are same");

    }

    else if(b1>0)

    {

        System.out.println(x1+" is greater than "+x2);
    }
    else
    {
        System.out.println(x1+" is smaller than "+x2);
    }
  }
}
```

Output:

```
67 is smaller than 98
```

Java Float class:-

The Float class wraps a primitive float type value in an object. Its object contains only a single field whose type is float.

Example:-

```
public class JavaFloatExample
{
    public static void main(String[] args)
     {
        Float f1 = new Float(34) ;
        Float f2 = new Float(f1/0);
```

```
      //isInfinite will return true for infinity values
       System.out.println(f2+" value for isInfinite() method is : " +Float.isInfinite(f2));
       System.out.println(f1+" value for Finite() method : "+Float.isFinite(f1));
       //converting float into int
       int f3=f1.intValue()*f2.intValue();
       System.out.println("intValue() method will return : "+f1+"*"+f2+" = "+f3 );
       // hash code of float value f1
       int f4 = f1.hashCode();
       System.out.println("Hash code value of "+f1+ " is : "+f4);
   }
}
```

Output:-

```
Infinity value for isInfinite() method is : true
34.0 value for Finite() method : true
intValue() method will return : 34.0*Infinity = -34
Hash code value of 34.0 is : 1107820544
```

Java Double class:-

The Double class generally wraps the primitive type double into an object. An object of Double class contains a field with the type Double.

Example:-

```
public class JavaDoubleExample

{

  public static void main(String[] args)

  {

      double d1 = 59d;

      double d2 = 90;

      System.out.println("The first value is : "+d1);

      System.out.println("The second value is : "+d2);

     int obj = Double.compare(d1, d2);

     if(obj==0)

     {

        System.out.println("Both the values are same");

     }

     else if(obj > 0)

     {

        System.out.println("The value '"+d1+"' is greater.");
```

```
    }
    else
    {
       System.out.println("The value '"+d2+"' is the greatest.");
    }
    // returns the maximum of the two numbwers.
     System.out.println("The maximum of the two values '"+
d1+"' and '"+d2+"' is given as :"+Double.max(d1, d2));
  }
}
```

Output:-

```
The first value is : 59.0
The second value is : 90.0
The value '90.0' is the greatest.
The maximum of the two values '59.0' and '90.0' is given as
 :90.0
```

Implementation of the wrapper class in Java:-

Autoboxing in Wrapper Class:-

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Wrapper class Example: Primitive to Wrapper

```
//Java program to convert primitive into objects

//Autoboxing example of int to Integer

public class WrapperExample

{

public static void main(String args[])

{

//Converting int into Integer

int a=20;

//converting int into Integer explicitly

Integer i=Integer.valueOf(a);

//autoboxing, now compiler will write Integer.valueOf(a)
internally

Integer j=a;

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

}

}
```

Output:

```
20 20 20
```

Unboxing in Wrapper Class:-

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

Wrapper class Example: Wrapper to Primitive

```
//Java program to convert object into primitives

//Unboxing example of Integer to int

public class WrapperExample

{

public static void main(String args[])

{

//Converting Integer to int

Integer a=new Integer(3);

//converting Integer to int explicitly

int i=a.intValue();

//unboxing, now compiler will write a.intValue() internally

int j=a;

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

}}
```

Output: - 3 3 3