

JAVA @17

Wrapper Classes

Objective

After completing this session you will be able to understand,

- What are Wrapper Classes?
- Need of Wrapper Classes?
- What are the types of Wrapper Classes?
- How to use Wrapper API to manipulate the wrapper classes?

Wrapper Classes

Wrapper Classes are an object representation of primitive data types.



Toffee

Toffee wrapped using a wrapper

Similarly, the primitive data types (the actual chocolate) are wrapped or converted into objects using wrapper classes

Primitive data types

converted to



Wrapper Objects

Why to Wrap primitive Data?

Why do we use wrapper Classes?

- Primitive data type are not objects. Whenever the **data** is **required** as a **object**, wrapper classes can be used to convert the primitive data into an object

Example: Collection can store only objects so primitives should be converted into Objects using wrapper and stored in collections.

- Wrapper classes provide many **utility** methods for processing the primitive data types.

Example: Comparing two values, converting number to String etc.

Refer to the Java documentation for the various API's available to process the primitive data using Wrapper classes.

<https://docs.oracle.com/javase/8/docs/api/>

Primitive To Wrapper Mapping

- Each primitive data type has a corresponding Wrapper class.
- The **name** of the wrapper object is **same** as the primitive data type except that the **first letter** is **capitalized**.
- **Eight** wrapper classes exist in **java.lang** package that represent 8 primitive data types

Example: Wrapper class of

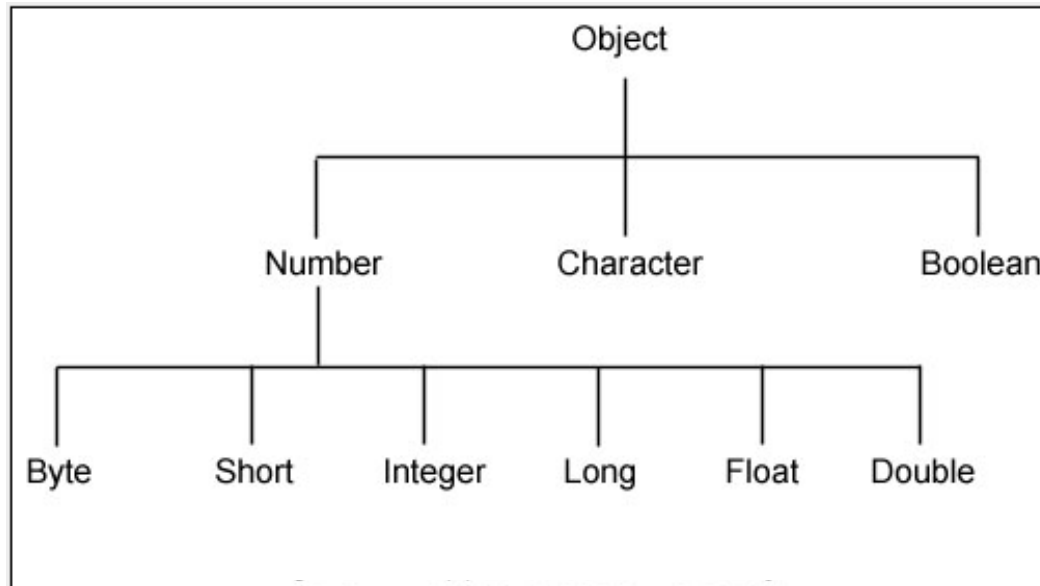
- `boolean` is `Boolean`
- `double` is `Double`

Primitive To Wrapper Mapping

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

All the wrapper classes comes under the **java.lang** package.

Wrapper Classes Hierarchy



- Super class of **Boolean** and **Character** is **Object**
- Super class of all Numeric wrapper classes is **Number**
- **Number** has six concrete subclasses that hold explicit values of each numeric type **Double**, **Float**, **Byte**, **Short**, **Integer**, and **Long**.

Converting Primitives to Wrappers

How to convert primitive to wrappers?

Wrapper constructors are used to create class objects from primitive types.

Example: `Double salary = new Double("5.0d");` 

Here the double primitive value 5.0 is converted to double wrapper object.

Similarly each primitive can be converted to wrapper object using the respective constructor.

- `Boolean boolean1 = new Boolean("false");`
- `Byte byte1 = new Byte("2");`
- `Short short1 = new Short("4");`
- `Integer int1 = new Integer("16");`
- `Long long1 = new Long("123");`
- `Float float1 = new Float("12.34f");`
- `Double double1 = new Double("12.56d");`
- `Character char1 = new Character('c');`

Converting Wrappers to Primitives

How to convert wrappers to primitive ?

Each wrapper provides methods to return the primitive value of the Object representation.

Example: `double sal = salary.doubleValue();`



Here the **Salary** Double wrapper object is converted to double primitive value.

Wrapper object can be converted to the respective primitive value as follows,

- `int i = int1.intValue();`
- `boolean b = boolean1.booleanValue();`
- `byte bt = byte1.byteValue();`
- `short s = short1.shortValue();`
- `long l = long1.longValue();`
- `float f = float1.floatValue();`
- `char c = char1.charValue();`

Convert int to String

How to convert int to String & vice versa?

You will often have the need to convert String to int and vice versa. This is done using methods in the Integer class.

Example1:

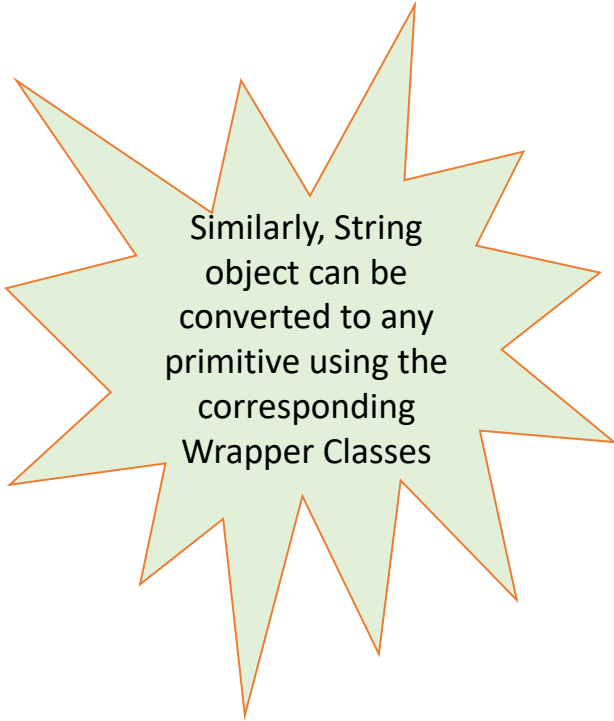
```
int penn = Integer.parseInt("45000");
```

The above statement converts the String "45000" into an int data type with value as 45000.

Example2:

```
String penn = Integer.toString(100);
```

The above statement converts the int 100 into an String data type with value as 100.



Similarly, String object can be converted to any primitive using the corresponding Wrapper Classes

Wrapper Classes

Let us learn about each of the Wrapper classes in detail



Integer Wrapper Class

The **Integer** class wraps a value of the primitive type **int** into an object.

This class also provides several methods for **converting int** to **String** and vice versa, as well as other useful methods for processing int value.

Let us learn to use the various Integer APIs:

Example 1: To get the primitive int Value of the Integer object

```
Integer intObject = new Integer(5);  
int priIntValue = intObject.intValue();
```

The Integer object with value 5 is converted into a primitive int data type.

Example 2: To get the value of the Integer as a String.

```
Integer intObject = new Integer(5);  
String stringValue = intObject.toString();
```

The Integer object with value 5 is converted into a String Object.

Are You Smart?

The Integer class has lot more APIs.

Are you smart enough to remember all API's?



You do not have to remember every method in the Integer class. Instead you can refer to the APIs available on the Web.

Navigate to <https://docs.oracle.com/javase/8/docs/api/>

and go through the various APIs available for the Integer class.

Lend a Hand – Integer Wrapper



Refer the Java documentation and choose the right API's for doing the below exercise.

1. Create a java class “IntegerDemo” inside the package com.wrapper.demos.
2. Create a main method and create two Integer Objects “val1” and “val2” with values 21122 and 43222. Implement the following logic,

Problem # 1: Convert val1 as int and val2 as long values and print the values as below.

Expected Output: “The int value of the Integer= ”<int value> & “The long value of the Integer= ”<long value>

Problem # 2: Compares both the Integer variables val1 and val2 using an API and print the bigger value.

Expected Output: “ The<value> is bigger than <value>”

Problem # 3: Retrieves and prints the Maximum and minimum value that an int can have.

Expected Output: “Maximum Int value=”<Max Value> “Minimum Int value=”<Min Value>

Problem # 4: Converts a String value of String s = “1234” to an int value and add 100 to it.

Expected Output: The string should be converted into a int incremented to 100 and store in a int variable.

Lend a Hand - Solution

Develop the solution as mentioned below and check the output.

```
package com.wrapper.demos;

public class IntegerDemo {

    public static void main(String[] args) {

        Integer val1=new Integer(21122);
        Integer val2=new Integer(43222);

        int intVal1=val1.intValue();
        System.out.println("Problem 1: "+"The int value of the Integer "+val1+" is "+intVal1 );

        long longVal1=val1.longValue();
        System.out.println("Problem 1: "+"The long value of the Integer "+val1+" is "+longVal1 );

        int compareValues=val1.compareTo(val2);
        if(compareValues>0){
            System.out.println("Problem 2: "+val1+" is bigger than "+ val2);}
        else{
            System.out.println("Problem 2: " +val2+" is bigger than "+ val1); }

        System.out.println("Problem 3: "+"The Maximum Value an int can hold is "+val1.MAX_VALUE);
        System.out.println("Problem 3: "+"The Minimum Value an int can hold is "+val1.MIN_VALUE);

        String strValue="1234";
        int num=Integer.parseInt(strValue);
        int finalValue=num+100;
        System.out.println("Problem 4: "+"The Converted String Value to int with 100" +
            " added to it is "+finalValue);

    }
}
```

Long Wrapper Class

Class Long:

The **Long** class wraps a value of the primitive type **long** in an object.

This class also provides several methods for **converting long** to **String** and vice versa, as well as other **constants** and useful methods when processing a long value.

Long is similar to a Integer, the difference is size of Long is 64 bits rather int is 32 bits.

Syntax:

```
Long salary= new Long("12678") // Converting a String to a Long Wrapper
```

(or)

```
Long salary = new Long(12678); // Converting a long primitive to a Long Wrapper.
```


Float Wrapper Class

Class Float:

The *Float* class wraps a value of the primitive type *float* in an object.

This class also provides several methods for converting *float* to *String* and vice versa, as well as other useful methods when dealing with a float.

Example:

$$\pi = 3.145$$

Lend a Hand – Float Wrapper



Refer the Java documentation and choose the right API's for doing the below exercise.

1. Create a java class "FloatDemo".
2. Create a main method and create two Float objects "val1" and "val2" with values 12.56f and 22.89f. Implement the following logic,

Problem 1: Compares two float variables and displays the maximum value.

Expected Output : " The<value> is bigger than <value>"

Problem 2: Converts a String value of "1234" to a float value and add 10.2f

Expected Output: The string should be converted into a float incremented to 100 and store in a int variable.

Problem 3: Checks if the specified float value val1 is a number or not.

Expected Output: If the number is not a number display the message "This is not a Number" else "This is a Number"

Lend a Hand - Solution

Develop the solution as mentioned and check the output.

```
package com.wrapper.demos;

public class FloatDemo {

    public static void main(String[] args) {

        Float val1=new Float(12.56f);
        Float val2=new Float(12.89f);

        int compareValues=val1.compareTo(val2);
        if(compareValues>0){
            System.out.println("Problem 1: "+val1+" is bigger than "+ val2);)
        }
        else{
            System.out.println("Problem 1: " +val2+" is bigger than "+ val1);)

        String strValue="1234";
        float num=Float.parseFloat(strValue);
        Float finalValue=num+10.2f;
        int intValue=finalValue.intValue();
        System.out.println("Problem 2: The int value of Float with " +
            "10.2f added to it="+intValue);

        boolean numCheck=val1.isNaN();
        if (numCheck) {

            System.out.println("Problem 3: "+val1+" is not a Number");)
        }
        else{
            System.out.println("Problem 3: "+val1+" is a Number");)
        }

    }
}
```

Double Wrapper class

Class Double:

The *Double* class wraps a value of the primitive type *double* in an object.

This class also provides several methods for converting double to String and vice versa, as well as other useful method to process double value.

Double is similar to a float, the difference is size of double data type is 64 bits rather float is 32 bits.

Where is it used?

This is typically used for storing currency values.



salary = 5000.45

Lend a Hand – Double



Refer the Java documentation and choose the right API's for doing the below exercise.

1. Create a java class "DoubleDemo".
2. Create a main method and create two Double objects "val1" and "val2" with values 87.89 and 212.82. The main method will have to print the following

Problem 1: Converts the val1 to int and val2 to float value.

Expected Output: "The int value of the Double Wrapper = "<double value> & "The float value of the Double Wrapper= "<long value>

Problem 2: Compares two double variables.

Expected Output: " The<value> is bigger than <value>"

Problem 3: Converts a String value of "1234.89" to an double value and add 100.89 to it.

Expected Output:"The double value of the String with 100.89 added to it ="<double value>"

Lend a Hand - Solution

```
package com.wrapper.demos;

public class DoubleDemo {

    public static void main(String[] args) {

        Double val1=new Double(87.89);
        Double val2=new Double(212.82);

        double doubleVal1=val1.doubleValue();
        System.out.println("Problem 1: The int value of the Double Wrapper= "+doubleVal1);

        float floatVal2=val2.floatValue();
        System.out.println("Problem 1: The float value of the Double Wrapper= "+floatVal2);

        int compareValues=val1.compareTo(val2);
        if(compareValues>0){
            System.out.println("Problem 2: "+val1+" is bigger than "+ val2);}
        else{
            System.out.println("Problem 2: " +val2+" is bigger than "+ val1);
        }

        String strValue="1234.89";
        double num=Double.parseDouble(strValue);
        double finalValue=num+100.89;
        System.out.println("Problem 3: The double value of String with " +
            "100.89 added to it="+finalValue);
    }
}
```

Develop the solution as mentioned
and check the output.

Byte Wrapper Class

Class Byte:

The **Byte** class wraps a value of the primitive type **byte** in an object.

This class also provides several methods for **converting byte** to **String** and vice versa, as well as other **constants** and useful methods for processing a byte.

Where is it used?

This is typically used for storing numbers ranging between **-128 and 127**.

Syntax:

```
Byte id = new Byte("12") //Converting a String to a Byte Wrapper
```

(or)

```
Byte id = new Byte(12) //Converting a primitive to a Byte Wrapper
```

Short Wrapper Classes

Class Short:

The **Short** class wraps a value of the primitive type **short** in an object.

This class also provides several methods for **converting short** to **String** and vice versa, as well as other **constants** and useful methods. useful when dealing with a short.

Where is it used?

This is typically used for storing numbers ranging between **-32768 and 32767**.

Syntax:

```
Short id = new Short("1261")    //Converting a String to a Short Wrapper
```

(or)

```
Short salary = new Short (1267); //Converting a short primitive to a Short Wrapper.
```


Character Wrapper Class

Class Character:

The **Character** class wraps a value of the primitive type **char** in an object.

This class also provides several methods for determining a character's category(lowercase letter, digit etc.) and for converting characters from uppercase to lowercase and vice versa.

Example: Characters 'a', 'b', '5', '?', 'A' can be represented a Character Object

Syntax:

Character flag = new Charachter('A') //Converting a String to a Boolean Wrapper

Boolean Wrapper class

Class Boolean:

The **Boolean** class wraps a value of the primitive type **boolean** in an object.

This class also provides several methods for **converting boolean** to **String** and vice versa, as well as other **constants** and useful methods when processing boolean data.

Syntax:

```
Boolean flag = new Boolean("true") //Converting a String to a Boolean Wrapper
```

(or)

```
Boolean flag = new Boolean(true); //Converting a boolean primitive to a Boolean Wrapper.
```

Lend a Hand – Boolean



Refer the Java documentation and choose the right API's for doing the below exercise.

1. Create a java class “**BooleanDemo**”.
2. Create a main method, inside the main method, declare two Boolean wrapper objects, val1 and val2 with values true and false respectively.
3. The main method should implement the following logic,

Problem # 1: The hashcode for val1 and val2 needs to be printed.

Expected Output:

“The hash code of val1 = ”<value> & “The hash code of val2= ”<value>

Lend a Hand - Solution

Develop the solution as mentioned
and check the output.

```
package com.wrapper.demos;

public class BooleanDemo {

    public static void main(String[] args) {

        Boolean val1=new Boolean(true);
        Boolean val2=new Boolean(false);
        System.out.println("The hash code of val1 "+" = "+val1.hashCode());
        System.out.println("The hash code of val2 "+" = "+val2.hashCode());

    }
```

Where Wrappers can be used?

Wrappers can be used in the following scenarios,

- As instance variables in class.

Example:

```
public class Employee {  
    private Double Salary = new Double(23456.87);  
    private Integer employeeId = new Integer(102345);  
}
```

- As method arguments.

Example:

```
public class Employee {  
    public void calculateTax(Double Salary, Float tax)  
    {  
        //Method implementation goes in here  
    }  
}
```

Method with Double and Float arguments.

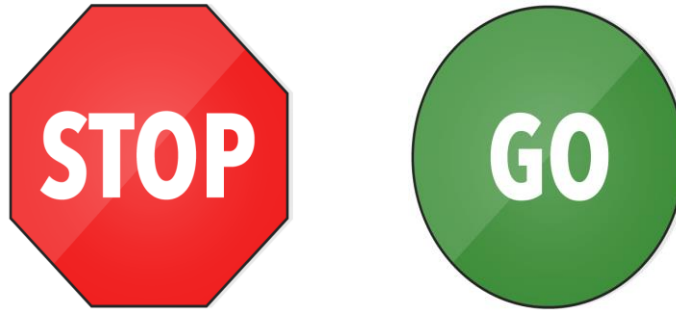
- As method return type.

Example:

```
public class Employee {  
    public Integer calculateTax(Double Salary, Float tax)  
    {  
        //Method implementation goes in here  
    }  
}
```

Method returns Integer type.

Time To Reflect



Trainees to reflect the following topics before proceeding.

- Why do we need Wrapper Objects?
- What Wrapper Object is used to store currency data?
- How many Wrapper Objects are there?
- What is the package under which the Wrapper objects are present?

Thank you

You have successfully completed
Wrapper Class