VIPS
Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

Course : OBJECT ORIENTED PROGRAMMING

Paper Code: AIDS-202,AIML 202,IOT 202

Faculty : Dr. Shivanka

Assistant Professor

VIPS

The Wrapper Classes

- "Wraps" the value of a primitive data type into an object
- Most programs use a combination of primitive data types and objects. But some class methods will accept only objects
- Useful when methods require an object argument
- Also useful for converting *Strings* to an *int* or *double*
- *Example: String to int when users input their age in a text box which returns a String always*

Wrapper Classes

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

Wrapper classes define an instance variable of that primitive data type and also provide useful constants and methods for converting between the objects and the primitive data types

Wrapper class

- 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

Uses of Wrapper class

- **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.

Why Do We Need Wrapper Classes in Java?

- Whenever the primitive types are required as an object, wrapper classes can be used. Wrapper classes also include methods to unwrap the object and give back the data type.
- In java.util package, the classes handle only objects. In this case wrapper class are helpful as they convert primitive data type to objects.
- In the Collection framework, Data Structures such as ArrayList store data only as objects and not the primitive types.

Autoboxing and Unboxing

- **Autoboxing**:
  - Automatic conversion between a primitive type and a wrapper object when a primitive type is used where an object is expected

    Integer intObject = 42;
- **Unboxing**
  - Automatic conversion between a wrapper object and a primitive data type when a wrapper object is used where a primitive data type is expected

    int fortyTwo = intObject;

Autoboxing

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.

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

Autoboxing example

Wrapper class Example: Primitive to Wrapper

```
//Java program to convert primitive into objects
//Autoboxing example of int to Integer
public class WrapperExample1{
public static void main(String args[]){
//Converting int into Integer
int a=20;
Integer i=Integer.valueOf(a);//converting int into Integer explicitly
Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a)  internally

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

Output:

```
20 20 20
```

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.

Using valueOf Static methods

- By using valueOf Static method, a Wrapper object can be created.
- Syntax:
- ClassName object = ClassName.valueOf(argument);
- Example:

- Integer hundred = Integer.valueOf("100");
- //100 is stored in variable. Here, Integer.valueOf(String str) is used.

- Integer seven = Integer.valueOf("111", 2);
- //binary 111 is converted to 7. Here, Integer.valueOf(String str, int base) is used.

Converting an Integer to a String


```
//Another example where we are converting an Integer to a String, and using the length() method to calculate the length of the "string":
public class CreatingWrapperObject2 {
  public static void main(String[] args) {
     //Creating the object using the Wrapper class
    Integer intValue = 1000;
     //Converting the integer value to String and assigning it to stringObject
    String stringObject = intValue.toString(); //toString() method used for the conversion
    //Printing the length of the String using length() method
    System.out.println(stringObject.length());
  }
```

Output:

4

Unboxing

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

```
//Unboxing example of Integer to int and Character to char
public class UnboxingExample{
public static void main(String args[]){
Character ch = 's';
//Unboxing - Character object to primitive conversion
char s = ch;
Integer a=new Integer(5);
//Converting Integer to int explicitly
int first=a.intValue();
//Unboxing, now compiler will write a.intValue() internally
int second=a;
 System.out.println(a);
System.out.println(first);
System.out.println(second);
}}
```

Output:

```
5
5
5
```

Unboxing example


```
//Java program to convert object into primitives
//Unboxing example of Integer to int
public class WrapperExample2{
public static void main(String args[]){
//Converting Integer to int
Integer a=new Integer(3);
int i=a.intValue();//converting Integer to int explicitly
int j=a;//unboxing, now compiler will write a.intValue() internally

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

Output:

```
3 3 3
```

Java Program to convert all primitives into its corresponding


```
//wrapper objects and vice-versa
public class WrapperExample3{
public static void main(String args[]){
byte b=10;
short s=20;
int i=30;
long l=40;
float f=50.0F;
double d=60.0D;
char c='a';
boolean b2=true;
```

```
//Autoboxing: Converting primitives into objects
Byte byteobj=b;
Short shortobj=s;
Integer intobj=i;
Long longobj=l;
Float floatobj=f;
Double doubleobj=d;
Character charobj=c;
Boolean boolobj=b2;
```

Java Program to convert all primitives into its corresponding


```
//Printing objects
System.out.println("---Printing object values---");
System.out.println("Byte object: "+byteobj);
System.out.println("Short object: "+shortobj);
System.out.println("Integer object: "+intobj);
System.out.println("Long object: "+longobj);
System.out.println("Float object: "+floatobj);
System.out.println("Double object: "+doubleobj);
System.out.println("Character object: "+charobj);
System.out.println("Boolean object: "+boolobj);
```

```
//Unboxing: Converting Objects to Primitives
byte bytevalue=byteobj;
short shortvalue=shortobj;
int intvalue=intobj;
long longvalue=longobj;
float floatvalue=floatobj;
double doublevalue=doubleobj;
char charvalue=charobj;
boolean boolvalue=boolobj;
```

Java Program to convert all primitives into its corresponding


```
//Printing primitives
System.out.println("---Printing primitive values---");
System.out.println("byte value: "+bytevalue);
System.out.println("short value: "+shortvalue);
System.out.println("int value: "+intvalue);
System.out.println("long value: "+longvalue);
System.out.println("float value: "+floatvalue);
System.out.println("double value: "+doublevalue);
System.out.println("char value: "+charvalue);
System.out.println("boolean value: "+boolvalue);
}}
```

Output:

```
---Printing object values---
Byte object: 10
Short object: 20
Integer object: 30
Long object: 40
Float object: 50.0
Double object: 60.0
Character object: a
Boolean object: true
-
```

Java Program to convert all primitives into its corresponding

Output

```
--Printing primitive values---
byte value: 10
short value: 20
int value: 30
long value: 40
float value: 50.0
double value: 60.0
char value: a
boolean value: true
```

Wrapper class

- Methods Supported by the Wrapper Classes
- All of the numeric wrapper classes are subclasses of the abstract class Number such as Byte, Integer, Double, Short, Float, Long.

Uses of Wrapper class

S. No.	Method	Method Description
1	typeValue()	Converts the value of this Number object to the specified primitive data type returned
2	compareTo()	Compares this Number object to the argument
3	equals()	Determines whether this Number object is equal to the argument
4	valueOf()	Returns an Integer object holding the value of the specified primitive data type value
5	toString()	Returns a String object representing the value of specified Integer type argument
6	parseInt()	Returns an Integer type value of a specified String representation
7	decode()	Decodes a String into an integer
8	min()	Returns the smaller value after comparison of the two arguments
9	max()	Returns the larger value after comparison of the two arguments
10	round()	Returns the closest round off long or int value as per the method return type

Thank You