Unit 07

WRAPPER CLASSES

UNIT STRUCTURE

7.0 Learning Objectives :

After learning this unit, you will be able to understand the use of :

- Byte Class
- Short Class
- Integer Class
- Long Class
- Float Class
- Double Class
- Boolean Class
- Character Class
- String Class

7.1 Introduction :

Java framework provides the Java.lang package. The java.lang contains several classes and interfaces. Java also provide the basic data types like bytes, short, int, long, float, double, char and Boolean. These all data types are used using variables. Sometimes programmer need to use the same data type as an objects. So Java provides object wrappers mechanism, this mechanism transformed the basic type in to object and become immutable. The object of the concern type can help to pass them by reference to methods instead of passing by value. The wrapper mechanism is as follow :

Integer int_obj = new Integer (35);

Here int_obj object encapsulates (wrapper) the integer value 35. So here the Integer class is known as a wrapper type.

7.2 Number Class :

Java provide abstract class "Number" under the java.lang package. The Number class contain six concrete subclasses to wrap the basic types. All the six classes provides several methods which can be used for I/O operation and conversion from one form the other form.

The six subclasses are as follow. :

- Byte
- Short
- Integer
- Long
- Double
- Float

We will discuss the methods of each class latter in this unit. Here bellow table 7.1 show the common methods that are available in all six classes.

Table 7.1 : Common Methods in all subclasses of Number Class

Method	Description
byte byteValue()	Return the byte value of the invoking object
short shortValue()	Return the short value of the invoking object
int intValue()	Return the integer value of the invoking object
long longValue()	Return the long value of the invoking object
float floatValue()	Return the float value of the invoking object
double doubleValue()	Return the double value of the invoking object
String toString()	Return the string value of the invoking object
int hashCode()	Return the hashcode value of the invoking object

7.3 Byte Class :

The Byte class is a wrapper class for the byte data type. The constructors for Byte class are as follow :

❖ **Constructors :**

Byte (byte num);

Byte(string str);

Here num is byte type and str is string representation of a byte.

The bellow table 7.2 gives a brief description of methods available under Byte Class :

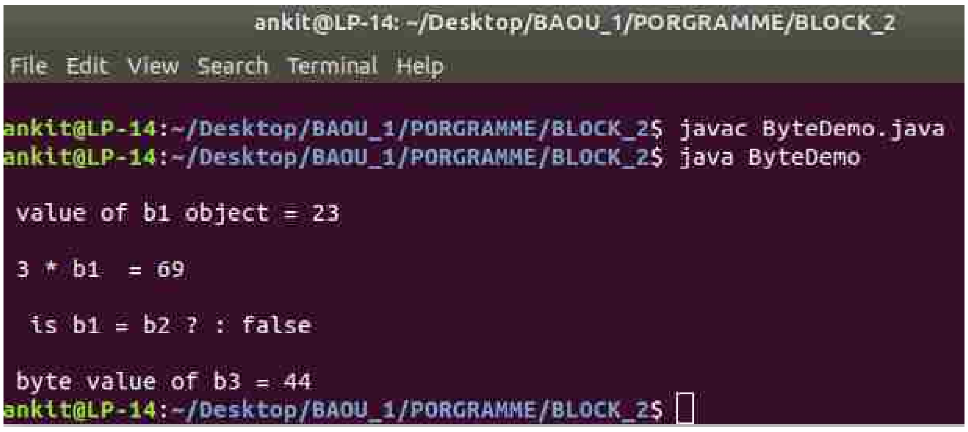
Table 7.2 : Methods defined in Byte Class

Method	Description
byte byteValue( )	This method return the value of the invoking object as a byte.
int compareTo(Byte byte)	This method compare the numerical value of the invoking object with that of byte. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value. It will return a positive value if the invoking object has a greater value.
int compareTo(Object obj)	To operate same as compareto(Byte)method, if obj is of class Byte Otherwise, this method will throw a ClassCastException.
double doubleValue( )	This method return the value of the invoking object as a double.
boolean equals(Object Obj)	This method return true if the invoking Byte object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	This method return the value of the invoking object as a float.
int intValue( )	This method return the value of the invoking object as an int.
long longValue( )	This method return the value of the invoking object as a long.
static byte parseByte(String s) throws NumberFormat Exception	This method return the byte equivalent of the number contained in the string specified by s. This method will used radix 10.
short shortValue( )	This method return the value of the invoking object as a short.
static String toString(byte number)	This method return a string that contains the decimal equivalent of number.
static Byte valueOf(String s) throws NumberFormat Exception	This method return a Byte object that contains the value specified by the string in s.
static Byte valueOf(String s, int radix) throws NumberFormat Exception	This method return a Byte object that contains the value specified by the string in s using the specified radix.

The following program show the use of some methods defined in Byte class :

```
class ByteDemo
{
 public static void main(String args[])
 {
  Byte b1 = new Byte((byte)23);
  Byte b2 = new Byte ("80");

  System.out.println("\n value of b1 object = " + b1);
  System.out.println("\n 3 * b1  = " + 3 * b1.byteValue());
  System.out.println("\n  is b1 = b2 ? : " + b1.equals(b2));
  Byte b3 = Byte.parseByte("44");
  System.out.println("\n byte value of b3 = " + b3);
 }
}
```


ankit@LP-14: ~/Desktop/BAOU_1/PORGRAMME/BLOCK_2
File Edit View Search Terminal Help

```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac ByteDemo.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java ByteDemo

 value of b1 object = 23

 3 * b1  = 69

  is b1 = b2 ? : false

 byte value of b3 = 44
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ 
```

Figure 7.1 Output of Program

❑ **Check Your Progress – 1 :**

1. Explain Byte wrapper Class.

....

....

7.4 Short Class :

The Short class is a wrapper class for the short data type. The constructors for short class are as follow :

❖ **Constructors :**

Short(short num);

Short(string str);

Here num is short type and str is string representation of a short.

The bellow table 7.3 gives a brief description of methods available under Short Class :

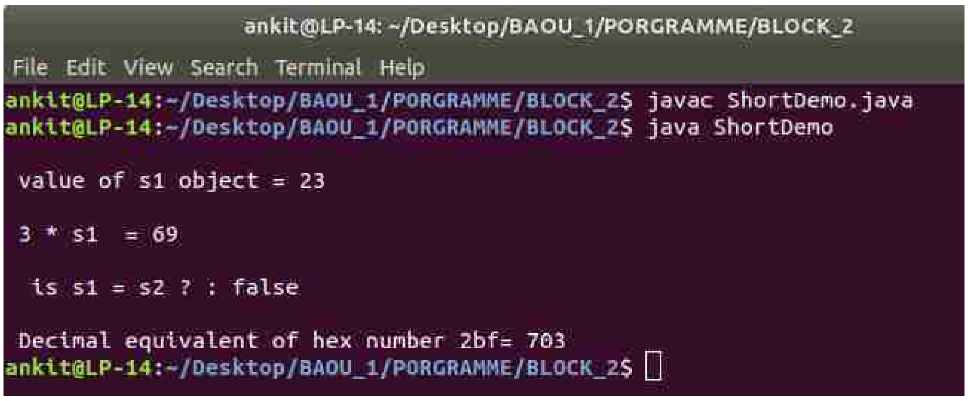
Table 7.3 : Methods defined in Short Class

Method	Description
byte byteValue( )	This method return the value of the invoking object as a byte.
int compareTo(Short obj)	This method compare the numerical value of the invoking object with that of byte. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value.
int compareTo(Object obj)	This method operate same as compareTo (Byte)method, if obj is of class Short Otherwise, this method will throw a ClassCast Exception.
double doubleValue( )	This method return the value of the invoking object as a double.
boolean equals(Object Obj)	This method return true if the invoking Integer object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	This method return the value of the invoking object as a float.
int intValue( )	This method return the value of the invoking object as an int.
long longValue( )	This method return the value of the invoking object as a long.
static byte parseByte(String str) throws NumberFormat Exception	This method return the short equivalent of the number contained in the string specified by str using radix 10.
short shortValue( )	This method return the value of the invoking object as a short.
static toString(short number)	This method return a string that contains the decimal equivalent of the invoking obj.
static String toString(short number)	This method return a string that contains the decimal equivalent of number.
static Short valueOf(String s) throws NumberFormat Exception	This method return a Short object that contains the value specified by the string in s. This method will use radix 10.
static Short valueOf(String s, int radix) throws NumberFormatException	This method return a Short object that contains the value specified by the string in s. This method will use the radix specified by the user.

The following program show the use of some methods defined in Short class :

```
class ShortDemo
{
 public static void main(String args[])
 {
  Short s1 = new Short((short)23);
  Short s2 = new Short("80");

  System.out.println("\n value of s1 object = " + s1);
  System.out.println("\n 3 * s1  = " + 3 * s1.shortValue());
  System.out.println("\n  is s1 = s2 ? : " + s1.equals(s2));
  short s3 = Short.valueOf("2bf",16);
 System.out.println("\n Decimal equivalent of hex number 2bf= " + s3);
 }
}
```


ankit@LP-14: ~/Desktop/BAOU_1/PORGRAMME/BLOCK_2
File Edit View Search Terminal Help

```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac ShortDemo.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java ShortDemo

 value of s1 object = 23

 3 * s1  = 69

  is s1 = s2 ? : false

 Decimal equivalent of hex number 2bf= 703
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$
```

Figure 7.2 Output of Program

❑ **Check Your Progress – 2 :**

1. Explain Short wrapper Class.

....

....

7.5 Integer Class :

The Integer class is a wrapper class for the int data type. The constructors for Integer class are as follow :

❖ **Constructors :**

Integer(int num);

Integer(string str);

Here num is int type and str is string representation of a int.

The bellow table 7.4 gives a brief description of methods available under Integer Class :

Table 7.4 : Methods defined in Interger Class

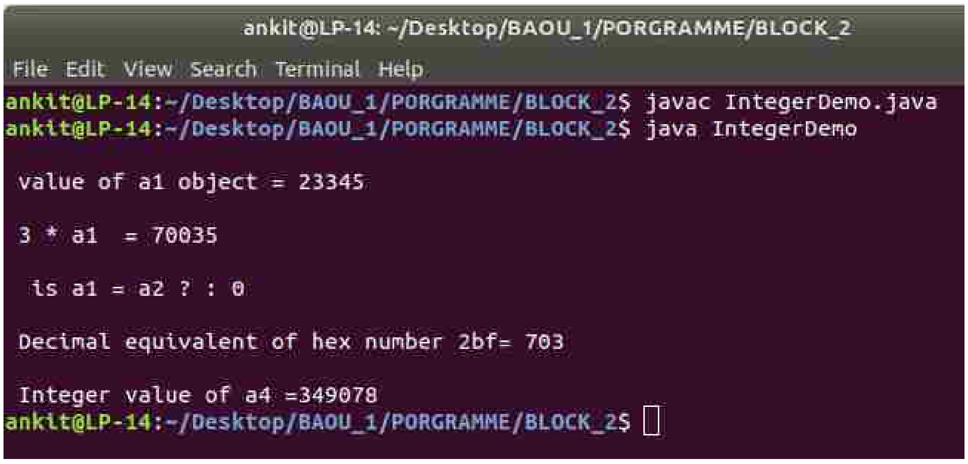
Method	Description
byte byteValue( )	This Method return the value of the invoking object as a byte.
int compareTo(Integer number)	Compare the numerical value of the invoking object with that of number. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value. It will return a positive value if the invoking object has a greater value.
double doubleValue( )	This Method return the value of the invoking object as a double.
boolean equals(Object Obj)	This Method return true if the invoking Integer object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	To return the value of the invoking object as a float.
int intValue( )	This Method return the value of the invoking object as an int.
long longValue( )	This Method return the value of the invoking object as a long.
static int parseByte(String s) throws NumberFormat Exception	This Method return the byte equivalent of the number contained in the string specified by s. This method will used radix 10.
short shortValue( )	This Method return the value of the invoking object as a short.
static toString(short number)	This Method return a string that contains the decimal equivalent of the invoking object.
static String toString(short number)	This Method return a string that contains the decimal equivalent of number.
static Short valueOf(String s) throws NumberFormat Exception	This Method return a Short object that contains the value specified by the string in s.

The following program show the use of some methods defined in Integer class :

```
class IntegerDemo
{
 public static void main(String args[])
 {
  Integer a1 = new Integer((23345));
```

```
    Integer a2 = new Integer("23345");

    System.out.println("\n value of a1 object = " + a1);
    System.out.println("\n 3 * a1  = " + 3 * a1.shortValue());
    System.out.println("\n  is a1 = a2 ? : " + a1.compareTo(a2));
    Integer a3 = Integer.valueOf("2bf",16);
   System.out.println("\n Decimal equivalent of hex number 2bf= " + a3);
    int a4 = Integer.parseInt("349078");
    System.out.println("\n Integer value of a4 =" + a4);
   }
  }
```


ankit@LP-14: ~/Desktop/BAOU_1/PORGRAMME/BLOCK_2
File Edit View Search Terminal Help

```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac IntegerDemo.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java IntegerDemo

 value of a1 object = 23345

 3 * a1  = 70035

  is a1 = a2 ? : 0

 Decimal equivalent of hex number 2bf= 703

 Integer value of a4 =349078
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ 
```

Figure 7.3 Output of Program

❑ **Check Your Progress – 3 :**

1. Explain Integer wrapper Class.

....

....

7.6 Long Class :

The Long class is a wrapper class for the long data type. The constructors for Long class are as follow :

❖ **Constructors :**

Long (long num);

Long(string str);

Here num is long type and str is string representation of a long.

The bellow table 7.5 gives a brief description of methods available under Long Class :

Table 7.5 : Methods defined in Long Class

Method	Description
byte byteValue( )	This method return the value of the invoking object as a byte.
int compareTo(Long number)	Compare the numerical value of the invoking object with that of number. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value. It will return a positive value if the invoking object has a greater value.
double doubleValue( )	This method return the value of the invoking object as a double.
boolean equals(Object Obj)	This method return true if the invoking long object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	This method return the value of the invoking object as a float.
int intValue( )	This method return the value of the invoking object as an int.
long longValue( )	This method return the value of the invoking object as a long.
static long parseByte(String s) throws NumberFormat Exception	This method return the byte equivalent of the number contained in the string specified by s. This method will used radix 10.
short shortValue( )	This method return the value of the invoking object as a short.
static toString(long number)	This method return a string that contains the decimal equivalent of the invoking object.
static String toString(long number)	This method return a string that contains the decimal equivalent of number.
static Long valueOf(String s) throws NumberFormat Exception	This method return a Short object that contains the value specified by the string in s.

7.7 Float Class :

The Float class is a wrapper class for the float data type. The constructors for Float class are as follow :

❖ **Constructors :**

Float (float num);

Float(double num);

Float(String str);

Here num is float and double type respectively and str is string representation of a float.

The bellow table 7.6 gives a brief description of methods available under Float Class :

Table 7.6 : Methods defined in Float Class

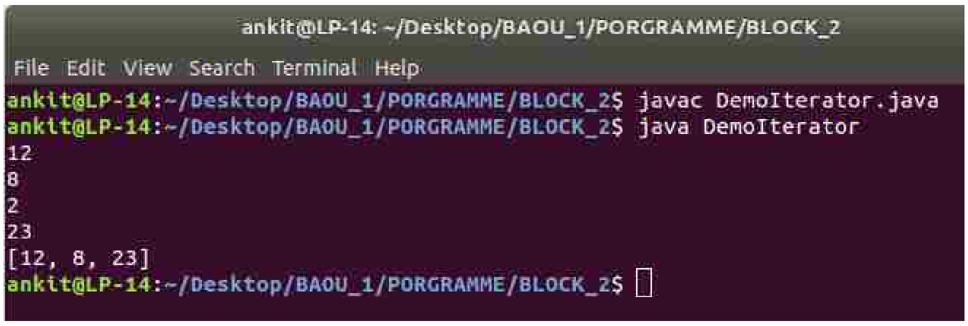
Method	Description
byte byteValue( )	This Method return the value of the invoking object as a byte.
int compareTo(Float number)	Compare the numerical value of the invoking object with that of number. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value. It will return a positive value if the invoking object has a greater value.
double doubleValue( )	This Method return the value of the invoking object as a double.
boolean equals(Object Obj)	This Method return true if the invoking float object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	This Method return the value of the invoking object as a float.
int intValue( )	This Method return the value of the invoking object as an int.
boolean isInfinite ()	This Method return true if the invoking object has an infinite value.
static Boolean isInfinite(float number)	This Method return true if the number specifies an infinite value. Otherwise, it will return false.
Boolean isNan ()	This Method return true if the invoking object contains a value that is not a number. Otherwise, it will return false.
static Boolean isNaN(float number)	This Method return true if number specifies an infinite value. otherwise, it returns false.
long longValue( )	This Method return the value of the invoking object as a long.
static long parseByte(String s) throws NumberFormat Exception	This Method return the float equivalent of the number contained in the string specified by s. This method will used radix 10.
short shortValue( )	This Method return the value of the invoking object as a short.
static toString( )	This Method return a string that contains the decimal equivalent of the invoking object.

static String toString(float number)	This Method return a string that contains the decimal equivalent of number.
static Long valueOf(String s) throws NumberFormat Exception	This Method return a Short object that contains the value specified by the string in s.

The following program show the use of some methods defined in Float class :

```
class FloatDemo
{
 public static void main(String args[])
 {
  Float a1 = new Float(23.345);
  Float a2 = new Float(745.22f);
  Float a3 = new Float("789.35");

  System.out.println("\n value of a1 object = " + a1);
  System.out.println("\n value of a1 object = " + a2);
   System.out.println("\n byte value of Float object a1  = " + a1.byteValue());
   System.out.println("\n Float value of Float object a3  = " + a3.floatValue());
  System.out.println("\n  is a1 = a2 ? : " + a1.compareTo(a2));
  float a4 = Float.parseFloat("688.564");
  System.out.println("\n Float value of a4 =" + a4 );
 }
}
```



```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac DemoIterator.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java DemoIterator
12
8
2
23
[12, 8, 23]
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$
```

Figure 7.4 Output of Program

❑ **Check Your Progress – 4 :**

1. Explain Float wrapper Class.

....

....

7.8 Double Class :

The Double class is a wrapper class for the double data type. The constructors for Double class are as follow :

❖ **Constructors :**

Double(double num);

Double(String str);

Here num is double type and str is string representation of a double.

The bellow table 7.7 gives a brief description of methods available under Double Class :

Table 7.7 : Methods defined in Double Class

Method	Description
byte byteValue( )	To return the value of the invoking object as a byte.
int compareTo(Double number)	To compare the numerical value of the invoking object with that of number. It will return 0 if the values are equal. It will return a negative value if the invoking object has a lower value. It will return a positive value if the invoking object has a greater value.
double doubleValue( )	To return the value of the invoking object as a double.
boolean equals(Object Obj)	To return true if the invoking double object is equivalent to Obj. Otherwise, it return false.
float floatValue( )	To return the value of the invoking object as a float.
int intValue( )	To return the value of the invoking object as an int.
boolean isInfinite ()	To return true if the invoking object has an infinite value.
static Boolean isInfinite(double number)	To return true if the number specifies an infinite value. Otherwise, it will return false.
Boolean isNan ()	To return true if the invoking object contains a value that is not a number. Otherwise, it will return false.
static Boolean isNaN(double number)	To return true if number specifies an infinite value. otherwise, it returns false.
long longValue( )	To return the value of the invoking object as a long.
static long parseByte(String s) throws NumberFormat Exception	To return the float equivalent of the number contained in the string specified by s. This method will used radix 10.

short shortValue( )	To return the value of the invoking object as a short.
static toString( )	To return a string that contains the decimal equivalent of the invoking object.
static String toString(double number)	To return a string that contains the decimal equivalent of number.
static Long valueOf(String s) throws NumberFormat Exception	To return a Short object that contains the value specified by the string in s.

7.9 Boolean Class :

The Boolean class is a wrapper class for the boolean data type. The constructors for Boolean class are as follow :

❖ **Constructors :**

Boolean (Boolean num);

Boolean (String str);

Here num is boolean type and str is string representation of a boolean.

The bellow table 7.8 gives a brief description of methods available under Boolean Class :

Table 7.8 : Methods defined in Boolean Class

Method	Description
Boolean BooleanValue ()	This method return boolean equivalent.
Boolean equals(Object obj)	This method return true if the invoking object is equivalent to obj. Otherwise this method will return false.
Static Boolean getBoolean (String propertyName)	This method return true if the system property specified by propertyName is true. Otherwise this method will return false.
int hashCode( )	This method return the hash code for the invoking object.
String toString ( )	This method return the string equivalent of the invoking object.
Static Boolean valueOf(Strign str)	This method return true if str contains the string "true". Otherwise this method will return false.

7.10 Character Class :

The Character class is a wrapper class for the char data type. The constructors for Character class are as follow :

❖ **Constructors :**

Character (char c);

Here c is char type.

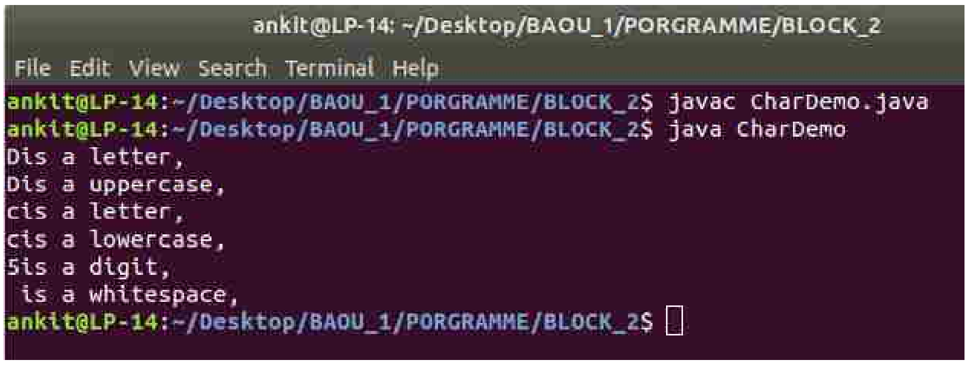
The bellow table 7.9 gives a brief description of methods available under Character Class :

Table 7.9 : Methods defined in Character Class

Method	Description
static Boolean isDefined(char c)	To return true if c is defined by Unicode. Otherwise, this method will return false.
static Boolean isDigit(char c)	To return true if c is a digit. Otherwise, this method will return false.
static Boolean isIdentifierIgnorable(char c) static Boolean isISoControl(char c)	To return true if c should be ignored in an identifier. Otherwise, this method will return false. To return true if c is an ISO control character. Otherwise, this method will return false.
static Boolean isJavaIdentifier Part(char c)	To return true if c is allowed as part of a Java identifier. Otherwise, this method will return false.
Static Boolean isJavaIdentifier Start(char c)	To return true if c is allowed as part of first character of a Java Identifier. Otherwise, this method will return false.
Static Boolean isLetter (char c)	To return true if c is a letter. Otherwise, this method will return false.
Static Boolean isLowerOrDigit (char c)	To return true if c is a letter of a digit. Otherwise, this method will return false.
Static Boolean isLowerCase (char c)	To return true if c is a lowercase letter. Otherwise, this method will return false.
Static Boolean isSpaceChar (char c)	To return true if c is Unicode space character. Otherwise, this method will return false.
Static Boolean isTitleCase(char c)	To return true if c is a Unicode titlecase character. Otherwise, this method will return false.
Static Boolean isUnicode IdentifierPart(char c)	To return true if c is allowed as part of a Unicode identifier. Otherwise, this method will return false.
Static Boolean isUnicode IdentifierStart(char c)	To return true if c is allowed as the first character of a Unicode identifier. Otherwise, this method will return false.
Static Boolean isUpperCase (char c)	To return true if c is an uppercase letter. Otherwise, this method will return false.
Static Boolean isWhitespace (char c)	To return true if c is whitespace. Otherwise, this method will return false.
Static Boolean toLowerCase (char c)	To return lowercase equivalent of c.
Static Boolean toTitleCase (char c)	To return titlecase equivalent of c.
Static char to UpperCase(char c)	To return uppercase equivalent of c.

The following program show the use of some methods defined in Character class :

```
class CharDemo
{
 public static void main (String args[])
 {
  char c[] = {'D', 'c', '@', '5', ' '};
  for(int i=0; i<c.length; i++)
  {
   if(Character.isDigit(c[i]))
    System.out.println(c[i]+ "is a digit,");
   if(Character.isLetter(c[i]))
    System.out.println(c[i]+ "is a letter,");
   if(Character.isWhitespace(c[i]))
    System.out.println(c[i]+ "is a whitespace,");
   if(Character.isUpperCase(c[i]))
    System.out.println(c[i]+ "is a uppercase,");
   if(Character.isLowerCase(c[i]))
    System.out.println(c[i]+ "is a lowercase,");
  }
 }
}
```


ankit@LP-14: ~/Desktop/BAOU_1/PORGRAMME/BLOCK_2
File Edit View Search Terminal Help

```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac CharDemo.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java CharDemo
Dis a letter,
Dis a uppercase,
cis a letter,
cis a lowercase,
5is a digit,
 is a whitespace,
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ 
```

Figure 7.5 Output of Program

❑ **Check Your Progress – 5 :**

1. Explain Character wrapper Class.

....

....

7.11 String Class :

In Java, String is defined as a sequence of characters. Java provides two classes for handling String, String and StringBuffer. String class deals with strings that are not altered after creation. StringBuffer class deals with strings that need alteration after they are created.

The constructors for String class are as follow :

❖ **Constructors :**

```
String stringName;
stringName = new String("ABC");

String StringName = new String("ABC");

char[] str = {'j','a','v','a'};
String strone = new String(str);

String str = "java";
```

The bellow table 7.10 gives a brief description of methods available under String Class :

Table 7.10 : Methods defined in String Class

Method	Description
char charAt(int index)	This method returns the character at the index position of the invoking string object.
void getChars(int start, int end, char target[], int target start)	It copies characters from object string starting at 'start' up to 'end' character in to 'target', starting at 'target start'
byte[] getBytes()	This method returns an array of characters as bytes from the String object.
boolean equals(Object str)	This method returns true if str contains the same string as that in the invoking object.
boolean equalsIgnoreCase (String str)	This method returns true if str contains the same string as in the invoking object by ignoring the case
boolean regionMatches (int start, String s2, int s2startindex, int numchars)	This method compare a region of the invoking object based on parameters pass.
boolean endsWith(String str)	This method returns true if the invoking String object end withstr.
boolean startWith(String str)	This method returns true if the invoking String object starts with str
int compareTo(String str)	This method compare the invoking String object with str.
int indexOf(int ch)	This method returns the index of first occurrence of the character ch in the invoking string object
int lastindexOF(int ch)	This method returns the index of last occurrence of the character ch in the invoking string object
int indexOf(String str)	This method returns the index of first occurrence of the string str in the invoking string object

int lastindexOF(String str)	This method returns the index of last occurrence of the string str in the invoking string object
String substring(int startIndex)	This method return a substring starting at startIndex till the end of the invoking String object
String substring(int startIndex, int endIndex)	This method return a substring starting at startIndex up to the endindex of the invoking String object
String concat(String str)	This method returns a new String after appending the str to the invoking String object
String replace(char existing Char, char newChar)	This method returns the new string created by replacing existingChar with newChar
String trim()	This method returns a new String after removing the leading and trailing white spaces of the invoking String object
String toLowerCase()	This method converts the uppercase characters of invoking String object to lowercase
String toUpperCase()	This method converts the lowercase characters of invoking String object to uppercase
int length()	This method return the number of characters in the invoking String object

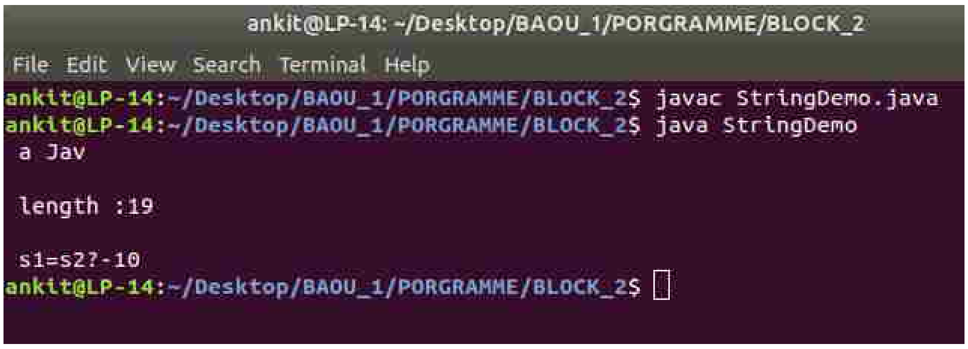
The following program show the use of some methods defined in String class :

```
class StringDemo
{
 public static void main (String args[ ])
 {
  String s1 = "This is a Java Text";
  String s2 = "This is a Text";
  char  data[] = new char [10];

  s1.getChars(7,13, data, 0);
  System.out.println(data);

  int count = s1.length();
  System.out.println("\n length :" + count);

  System.out.println("\n s1=s2?" + s1.compareTo(s2));
 }
}
```


ankit@LP-14: ~/Desktop/BAOU_1/PORGRAMME/BLOCK_2
File Edit View Search Terminal Help

```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac StringDemo.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java StringDemo
 a Jav

 length :19

 s1=s2?-10
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$
```

Figure 7.6 Output of Program

❑ **Check Your Progress – 6 :**

1. Explain String Class.

   ....

   ....

2. Java provide ________ abstract class for the wrapper the data type.

   (A) Integer (B) Number (C) Data Type (D) String

3. ________ return the byte value of the invoking object as a byte in Byte class.

   (A) byteValue() (B) doubleValue()

   (C) byte() (D) ToByte()

4. ________ method compare the numerical value of the invoking object with that of byte under Short class.

   (A) compareTo(Short obj) (B) compareTo(int obj)

   (C) compareTo(Float obj) (D) compareTo(Double obj)

5. ________ method return the value of the invoking object as a long under Integer class.

   (A) Long () (B) Longvalue()

   (C) IntegerValue() (D) longValue()

6. ________ method return the true if the invoking object contains a value that is not a number under Double class.

   (A) isNan() (B) isnotnumber()

   (C) isnumber() (D) isdigit()

7. The isInfinite() method return true if the invoking object has an infinite value under Float class.

   (A) True (B) False

8. Boolean isDigit(char c) method return true if char c is a digit under Character class.

   (A) True (B) False

9. ________ method return lowercase equivalent of character under Character class.

   (A) toLowerCase(char c) (B) LowerCase(char c)

   (C) toLowerCase() (D) toSmallCase(char c)

10. Java provides two classes for handling String, String and StringBuffer.

    (A) True (B) False

11. String class deals with strings that can altered after creation

    (A) True (B) False

7.12 Converting Numbers to and from Strings :

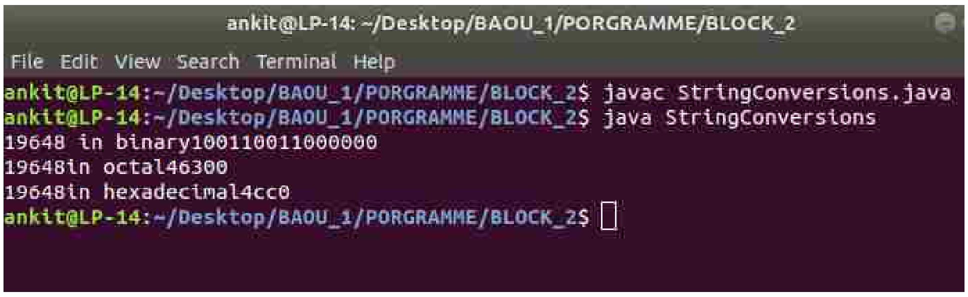
Java provides an easy way to convert numbers into string. The Byte, Short, Integer and Long classes provide the parseByte( ), parseShort( ), parseInt( ) and parseLong( ) methods, respectively. These methods return the byte, short, int or long equivalent of the numeric string with which they are called.

The given below program demonstrates the use of parseInt( ). It finds the sum of a list of integers entered by the user. It reads the integers using readLine ( ) and uses parseInt( ) to convert these strings into their int equivalents.

The following program show the use convert an integer into binary, hexadecimal and octal :

```
class StringConversions
{
 public static void main (String args[])
 {
  int num=19648;
 System.out.println(num + " in binary" + Integer.toBinaryString(num));
  System.out.println(num + "in octal" + Integer.toOctalString (num));
 System.out.println(num + "in hexadecimal" + Integer.toHexString (num));
 }


}
```



```
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ javac StringConversions.java
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$ java StringConversions
19648 in binary100110011000000
19648in octal46300
19648in hexadecimal4cc0
ankit@LP-14:~/Desktop/BAOU_1/PORGRAMME/BLOCK_2$
```

Figure 7.7 Output of Program

7.13 Let Us Sum Up :

In this Unit we have learned about a superclass which is defined by the abstract class Number that implements the classes that wrap the numeric type's byte, short, int, long, float and double. Number possesses abstract methods that return the value of the object in each of the different number formats. That is, doubleValue ( ) returns the value as a double, floatValue ( ) returns the value as a float and so on. Double and Float are wrappers for floating–point values of type double and float respectively. The constructors of float are Float (double num) ,Float (float num) and Float (String str) throws

NumberFormatException. The Float objects can be constructed with values of type float or double. They can also be constructed from the string representation of a floating–point number. Whereas, the constructors for Double are shown as Double (double num), Double (String str) throws Number Format Exception. Double objects can be constructed with a double value or a string containing a floating–point value.

Java provides an easy way to convert numbers into string. The Byte, Short, Integer and Long classes provide the parseByte( ), parseShort( ), parseInt ( ) and parseLong( ) methods, respectively. These methods return the byte, short, int or long equivalent of the numeric string with which they are called.

7.14 Suggested Answer for Check Your Progress :

- **Check Your Progress 1 :**

  See Section 7.3

- **Check Your Progress 2 :**

  See Section 7.4

- **Check Your Progress 3 :**

  See Section 7.5

- **Check Your Progress 7 :**

  See Section 7.7

- **Check Your Progress 5 :**

  See Section 7.10

- **Check Your Progress 6 :**

1 : See Section 7.11	**2 :** B	**3 :** A	**4 :** A	**5 :** D
6 : A	**7 :** A	**8 :** A	**9 :** A	**10 :** A
11 : B				

7.15 Glossary :

1. **Number Class** – Java provide Number class with six subclasses. Using the subclasses we can create the object of basic data type.
2. **Byte Class** – The Byte class is a wrapper class for the byte data type.
3. **Short Class** – The Short class is a wrapper class for the short data type.
4. **Integer Class** – The Integer class is a wrapper class for the int data type.
5. **Long Class** – The Long class is a wrapper class for the long data type.
6. **Float Class** – The Float class is a wrapper class for the float data type.
7. **Double Class** – The Double class is a wrapper class for the double data type.
8. **Boolean Class** – The Boolean class is a wrapper class for the boolean data type.
9. **Character Class** – The Character class is a wrapper class for the char data type.
10. **String Class** – String is defined as a sequence of characters. Java provides two classes for handling String, String and StringBuffer.

7.16 Assignment :

1. Write a note on Number Class.
2. Write a note on Wrapper Class.

7.17 Activities :

1. Write a program to show the use of String Class.
2. Write a program to show the use of Double Class.
3. Write a program to show the use of Long Class.
4. Write a program to show the use of Boolean Class.

7.18 Case Study :

1. Prepare the Chart of five method from each wrapper class.

7.19 Further Reading :

1. Core Java 2, 2 volumes, Cay S. Horstmann, Gary Cornell, The Sun Microsystems Press, 1999, Indian reprint 2000.
2. Java 2, the Complete Reference, Patrick Naughton and Herbert Schildt, Tata McGraw Hill, 1999.
3. Programming with Java, Ed. 2, E. Balagurusamy, Tata McGraw Hill, 1998, reprint, 2000.
4. The Java Tutorial, Ed. 2, 2 volumes, Mary Campione and Kathy Walrath, Addison Wesley Longmans, 1998.
5. The Java Language Specification, Ed. 2, James Gosling, Bill Joy, Guy Steele & Gilad Bracha, (Ed. 1 1996, Ed. 2 2000), Sun Microsystems, 2000.
6. Using Java 2, Joseph L. Weber, Prentice Hall, Eastern Economy Edition, 2000