Wrapper Class:

Try Out :

Creating Objects

```
Integer i1 = new Integer(5);
Integer i2 = new Integer(“5”);

Float f1 = new Float(5.6f);
Float f2 = new Float(“5.6f”);

Double d1 = new Double(6.7);
Double d2 = new Double(“6.7”);

Character c = new Character(‘a’);

Boolean b1 = new Boolean(true);
Boolean b2 = new Boolean(“true”);
Boolean b3 = new Boolean(“TrUe”)
```

Tryout: autoboxing & AutoUnboxing

1.

```
class Tester {
        public static void main(String[] args) {

                char ch1 = 'a';
                Character z1 = ch1;
                System.out.println(z1); //concept of Boxing

                Integer ch2=10;
                int z2=ch2;
                System.out.println(z2); //concept of UnBoxing


                System.out.println("****************************************");
                // Type casting cannot be used to convert any wrapper type to another
                // Long phoneNoLong = (Long) new Integer(44281234);      // Will give
compilation error

                // We can make use of methods like intValue(), byteValue(), floatValue(), etc.
                Integer phoneNo = 44281234;
                Long phoneNoLong = phoneNo.longValue();
                System.out.println("Integer to Long: " + phoneNoLong);

            System.out.println("****************************************");
```

```
		// Wrappers can be used to convert Integer to numeric String data type by using
toString() method
		Integer my = 100;
		String myStr = my.toString();
		System.out.println(myStr.length());

		System.out.println("****************************************");
		// Wrappers can be used to convert numeric strings to numeric data types
		double numDouble = Double.parseDouble("123.45");
			System.out.println("String to double: " + numDouble);

			System.out.println("****************************************");
			// Wrappers can be used to convert an Integer to Binary String by using
toBinaryString() method
			System.out.println("Integer 5 as binary string: " + Integer.toBinaryString(5));
	}
}
```

2.

//This program demonstrates usage **of various methods in Character Wrapper class**

```
 class Tester {
	public static void main(String[] args) {
		//Creating Character instances using valueOf()
		Character alphaObj = Character.valueOf('A');
		Character digitObj = Character.valueOf('5');

		//method to check if a given character is Alphabet
		System.out.println("Z is Alphabet:"+Character.isLetter('Z'));

		//method to check if a given character is Digit
		System.out.println("5 is Digit:"+Character.isDigit(digitObj));

		//method to check if a given character is LowerCase
		System.out.println("b is LowerCase:"+Character.isUpperCase('b'));

		//method to convert to LowerCase
		System.out.println(Character.toLowerCase('Z'));
		System.out.println(Character.toLowerCase(66));

		//method to convert Character to String
		String xobj = Character.toString('x');
		System.out.println(xobj);

		//method to convert Character to char primitive data type
		char alpha = alphaObj.charValue();
		System.out.println(alpha);
```

```
            //Comparing chars using compare which returns int
            //It returns 0 if char1 == char2;
            //a value less than 0 if char1 < char2;
            //and a value greater than 0 if char1 > char1
            System.out.println(Character.compare('A', 'b'));

            //Comparing Character objects using compareTo which returns int
            //It returns 0 if obj1 == obj2;
            //a value less than 0 if obj1 < obj2;
            //and a value greater than 0 if obj1 > obj2
            Character anotherCharacter = Character.valueOf('b');
            System.out.println(alphaObj.compareTo(anotherCharacter));
        }

}
```

Wrapper class methods

```
public class WrapperClassMethodsExample {
    public static void main(String[] args) {
        // valueOf() - Converts a string to the corresponding wrapper object
        Integer intValue = Integer.valueOf("42");
        Double doubleValue = Double.valueOf("42.42");
        Boolean booleanValue = Boolean.valueOf("true");
        System.out.println("valueOf: Integer=" + intValue + ", Double=" + doubleValue +
", Boolean=" + booleanValue);

        // xxxValue() - Retrieves the primitive value from the wrapper object
        int primitiveInt = intValue.intValue();
        double primitiveDouble = doubleValue.doubleValue();
        System.out.println("xxxValue: int=" + primitiveInt + ", double=" +
primitiveDouble);

        // parseXxx() - Parses a string to a primitive value
        int parsedInt = Integer.parseInt("100");
        double parsedDouble = Double.parseDouble("99.99");
        boolean parsedBoolean = Boolean.parseBoolean("false");
        System.out.println("parseXxx: int=" + parsedInt + ", double=" + parsedDouble +
", boolean=" + parsedBoolean);

        // isLetter(), isDigit(), isUpperCase() - Character class methods
        char sampleChar = 'A';
        System.out.println("isLetter: " + Character.isLetter(sampleChar));
```

```
        System.out.println("isDigit: " + Character.isDigit('5'));
        System.out.println("isUpperCase: " + Character.isUpperCase(sampleChar));

        // toLowerCase() - Converts character to lowercase
        char lowerChar = Character.toLowerCase(sampleChar);
        System.out.println("toLowerCase: " + lowerChar);

        // toString() - Converts a wrapper object or primitive to a string
        String intAsString = Integer.toString(primitiveInt);
        String doubleAsString = Double.toString(primitiveDouble);
        System.out.println("toString: intAsString=" + intAsString + ", doubleAsString=" +
doubleAsString);
    }
}
```

Questions: Wrapper

Q3 of 4 (don't use)

```
Integer x = 100;
Integer y = 100;
System.out.println(x == y);
x = 128;
y = 128;
System.out.println(x == y);
```

true false –
true true
false false
false true
Explanation :

The second print statement gives out false due to the cache managed by the JVM. The JVM returns an already created object for any values between -128 and 127, where as any Integer value outside this range is converted to a new Integer object.

Q4 of 5

Predict the output of the following code snippet:

```
public class Tester {

    public static void main(String[] args) {
                char test[] = {'a', 'A' , '5'};
                System.out.println(Character.isDigit(test[2]));
                        System.out.println(Character.isUpperCase(test[1]));
            }
}
```

true true –
false true
true false
Explanation :

Character.isDigit() checks whether the value is digit and Character.isUpperCase() checksfor uppercase value.
test[2] is digit and test[1] is in uppercase, therefore both are true.

Q5 of 5

In the below code snippet, auto-boxing occurs in which statement?

```
Integer[] intArray = new Integer[12];  //line 1
intArray[0] = 20;                      //line 2
int value = intArray[0];               //line 3
```

line 1
line 2 –
line 3
No auto-boxing used
Explanation :

The int value 20 is wrapped as integer object reference.

TryOut: String

```
class Tester {
        public static void main(String args[]) {

                // length()
```

```
String str = "Welcome";
System.out.println(str.length());

// concat()
String s = "Hello";
s.concat(" World");
System.out.println(s);
// s is still "Hello"
// String objects are immutable which means they cannot be changed
// Here, when we concat the two strings a new string object gets created

String s1 = s.concat("World");
System.out.println(s1);

// + operator can also be used for string concatenation
String fname = "Jack";
String lname = "Black";
System.out.println(fname + " " + lname);

// equals()
System.out.println(s.equals("Hello"));

// equals compares only the values of the strings
String s2 = new String("Hello");
System.out.println(s.equals(s2));

// == compares the object reference and will return false in the below
// case
System.out.println(s == s2);

// equalsIgnoreCase()
System.out.println(s.equalsIgnoreCase("hello"));

// toLowerCase() and toUpperCase()
System.out.println(str.toLowerCase());
System.out.println(str.toUpperCase());

// substring()
String subs = "Learning is fun";
System.out.println(subs.substring(4, 8));
System.out.println(subs.substring(4));

// charAt()
System.out.println(subs.charAt(10));

// contains()
System.out.println(subs.contains("is"));
```

```
                // replace()
                System.out.println(subs.replace('i', 'k'));
        }

}
```

TryOut: StringBuffer

```
 public class StringBufferDemo{

   public static void main(String[] args){

     String firstName="Sachin";

     String lastName="Tendulkar";

     String fullName=firstName+lastName;

     //'+'operator concatenates the string but creates a new object in the heap memory as sting is immutable

     System.out.println(fullName);

     StringBuffer sb=new StringBuffer(firstName);

     String fName=sb.append(lastName).toString();//toString method converts StringBuffer to String

     //StringBuffer is mutable, it appends to a single object

     System.out.println(fName);

    sb.insert(1,"java");  //Output Sjavachin

   sb.delete(1,3);      // Shin

   sb.reverse();   //nihcaS

  sb.replace(1,3,"Java")   //SJavahin

   }

 }
```

TryOut: StringBuilder

```
//creation of StringBuilder Object with capacity 50.

StringBuilder name = new StringBuilder(50);

name.append("Mississippi");

int length = name.length();  // will give the length of address
```

```
name.insert(length, " River");
System.out.println(name);   //Output :- Mississippi River
name.reverse();    // Output :- reviR ippississiM
name.delete(5, 10); // Output :- reviRssissiM
System.out.println(name.charAt(3));  // Output :- i
```

StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

Assignment: String

Method Description

Remove all the white spaces from the string passed to the method and return the modified string.

Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Sample Input	Expected Output
"Hello How are you "	HelloHowareyou
"J ava pro gramming"	Javaprogramming