Q1.

```
class TestString{
    public static void main(String args[]){
        String s1="abc";
        String s2=new String("abc");
        String s3=new String("abc");
        String s4="abc";
        System.out.println("s1==s2 :"+(s1==s2));
        System.out.println("s1==s3 :"+(s1==s3));
        System.out.println("s1==s4 :"+(s1==s4));
        System.out.println("s2==s3 :"+(s2==s3));
    }
}
```

Q2.

```
class TestString{
    public static void main(String args[]){
        String s1="abc";
        s1.concat("def");
        System.out.println("s1 :"+s1);
    }
}
```

Q3.

```
    class TestString{
        public static void main(String args[]){
            String s1="abc";
            String s2=s1;
            s1=s1.concat("def");
            System.out.println("s1 :"+s1);
            System.out.println("s2 :"+s2);
        }
    }
```

Q4.

```
    class TestString{
        public static void main(String args[]){
            String s1="abc";
            String s2=s1;
            s1=s1+"abc";
            System.out.println("s1 :"+s1);
            System.out.println("s2 :"+s2);
        }
    }
```

Q5.

```
    class TestString{
        static String m(){
            String name="abc";
            return name;
        }
        public static void main(String args[]){
            String s1="abc";
            String s2=m();
            System.out.println("s1==s2 :"+(s1==s2));
        }
    }
```

Q6.

```
class TestString{
    static String m(){
        String name=new String("abc");
        return name;
    }
    public static void main(String args[]){
        String s1="abc";
        String s2=m();
        System.out.println("s1==s2 :"+(s1==s2));
    }
}
```

Q7.

```
class TestString{
    public static void main(String args[]){
        String s1="abc";
        System.out.println(s1=="abc");
    }
}
```

Q8.

```
class TestString{
    public static void main(String args[]){
        String s1="abc";
        String s2="ab";
        String s3=s2+"c";
        System.out.println(s1==s3);
    }
}
```

Q9.

```
class TestString{
    public static void main(String args[]){
        String s1="abc";
        final String s2="ab";
        String s3=s2+"c";
        System.out.println(s1==s3);
    }
}
```

Q10. String Constructors

```
class TestString{
    public static void main(String args[]){
        String s1=new String(); //No parameter
        String s2=new String("abc");//String parameter
        char ch[]={'a','b','c'};
        String s3=new String(ch);//char array
        byte br[]={97,98,99};
        String s4=new String(br);//byte array
        System.out.println(s1+" "+s2+" "+s3+" "+s4);
    }
}
```

Q11. Special Constructor methods

char	charAt(int index)	Returns the character at the specified index

```
class TestCharAt{
    public static void main(String args[]){
        String s1=new String("abcXdef");
        char ch1=s1.charAt(0);
        char ch2=s1.charAt(3);
        System.out.println(ch1+" "+ch2);

        char ch3=s1.charAt(-1); //Run time error StringIndexOutOfBoundsException
        char ch4=s1.charAt(7);  //Run time error StringIndexOutOfBoundsException
    }
}
```

Q12.

String	**concat**(String str)	Concatenates the specified string to the end of this string.

```
class Testconcat{
    public static void main(String args[]){
        String s1=new String("abc");
        s1.concat("xyz");
        System.out.println("Now s1 :"+s1);
        s1=s1.concat("xyz");
        System.out.println("Now s1 :"+s1);
    }
}
```

Q13.

boolean	**equals**(Object anObject)	Compares this string to the specified object.

```
class Test_equals{
    public static void main(String args[]){
        String s1=new String("abc");
        String s2=new String("abc");
        String s3=new String("def");
        System.out.println("s1 & s2 :"+s1.equals(s2));
        System.out.println("s1 & s3 :"+s1.equals(s3));
        System.out.println("s1==s2 :"+(s1==s2));
        System.out.println("s1==s3 :"+(s1==s3));
    }
}
```

Q14.

boolean **equalsIgnoreCase**(String anotherString)	Compares this String to another String, ignoring case considerations.

```
class Test_equalsIgnoreCase{
    public static void main(String args[]){
        String s1=new String("abc");
        String s2=new String("ABc");
        String s3=new String("abc");
        System.out.println("s1 & s2 :"+s1.equalsIgnoreCase(s2));
        System.out.println("s1 & s3 :"+s1.equalsIgnoreCase(s3));
    }
}
```

Q15.

`int length()`	Returns the length of this string

```
class TestLength{
    public static void main(String args[]){
    String s1=new String("abcd abc");
    System.out.println("No of characters :"+s1.length());
    }
}
```

Q16.

`String substring(int beginIndex)`	Returns a new string that is a substring of this string.

```
class Test{
    public static void main(String args[]){
        String s1="Sun Certified Java Programmer";
        String s2=s1.substring(14);
        System.out.println(s2);
    }
}
```

Q17.

`String substring(int beginIndex,int endIndex)`	Returns a new string that is a substring of this string.

```
class Test{
    public static void main(String args[]){
        String s1="Sun Certified Java Programmer";
        String s2=s1.substring(0,3);
        System.out.println("0-3 :"+s2);
        String s3=s1.substring(4,14);
        System.out.println("4-14 :"+s3);
        // String s4=s1.substring(5,3); //wrong index
    }
}
```

Q18.

`String toLowerCase()`	Converts all of the characters in this String to lower case using the rules of the default locale.
`String toUpperCase()`	Converts all of the characters in this `String` to upper case using the rules of the default locale.

```
class Test{
    public static void main(String args[]){
        String s1="EnGliSh";
        System.out.println("Normal Case :"+s1);
        System.out.println("Lower Case :"+s1.toLowerCase());
        System.out.println("Upper Case :"+s1.toUpperCase());
    }
}
```

Q19.

```
class Test{
    public static void main(String args[]){
        String s1="ijts";
        String s2=s1.toUpperCase();
        String s3=s1.toLowerCase();
        System.out.println("s1==s2 :"+(s1==s2));
        System.out.println("s1==s3 :"+(s1==s3));
    }
}
```

Q20.

`String` **`trim`**`()`	Returns a copy of the string, with leading and trailing whitespace omitted.

```
class Test{
    public static void main(String args[]){
        String s1=" EnGliSh ";
        System.out.println("No of character :"+s1.length());
        s1.trim();
        System.out.println("No of character :"+s1.length());

        String s2=s1.trim();
        System.out.println("No of character :"+s2.length());
    }
}
```

StringBuffer

Q21. StringBuffer

```
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer sb1=new StringBuffer("SCJP @ ");
        StringBuffer sb2=sb1;
        sb1.append("IJTS"); //appends to the end of the String
        System.out.println(sb1);
        System.out.println(sb2);
    }
}
```

Q22. StringBuffer Special Methods

`StringBuffer` **`append`**`(boolean b)`	Appends the string representation of the boolean argument to the string buffer.
`StringBuffer` **`append`**`(char c)`	Appends the string representation of the char argument to this string buffer.
`StringBuffer` **`append`**`(double d)`	Appends the string representation of the double argument to this string buffer.
`StringBuffer` **`append`**`(float f)`	Appends the string representation of the float argument to this string buffer.
`StringBuffer` **`append`**`(int i)`	Appends the string representation of the int argument to this string buffer.
`StringBuffer` **`append`**`(long l)`	Appends the string representation of the long argument to this string buffer.
`StringBuffer` **`append`**`(Object obj)`	Appends the string representation of the Object argument to this string buffer.
`StringBuffer` **`append`**`(String str)`	Appends the string to this string buffer.
`StringBuffer` **`append`**`(StringBuffer sb)`	Appends the specified StringBuffer to this StringBuffer.

```
class A{
    public String toString(){return "Class A";}
}
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer sb1=new StringBuffer();
        sb1.append(10);
        System.out.println(sb1);
        sb1.append(10.99);
        System.out.println(sb1);
        sb1.append("VV");
        System.out.println(sb1);
        sb1.append(14.5f);
        System.out.println(sb1);
        sb1.append(new A());
        System.out.println(sb1);
    }
}
```

Q23. insert()

`StringBuffer insert(int offset, boolean b)`	Inserts the string representation of the boolean argument into this string buffer.
`StringBuffer insert(int offset, char c)`	Inserts the string representation of the char argument into this string buffer.
`StringBuffer insert(int offset, double d)`	Inserts the string representation of the double argument into this string buffer.
`StringBuffer insert(int offset, float f)`	Inserts the string representation of the float argument into this string buffer.
`StringBuffer insert(int offset, int i)`	Inserts the string representation of the second int argument into this string buffer.
`StringBuffer insert(int offset, long l)`	Inserts the string representation of the long argument into this string buffer.
`StringBuffer insert(int offset, Object obj)`	Inserts the string representation of the Object argument into this string buffer.
`StringBuffer insert(int offset, String str)`	Inserts the string into this string buffer.

```
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer sb1=new StringBuffer("abcd");
        sb1.insert(2,false);
        System.out.println(sb1);
        sb1.insert(6,new A());
        System.out.println(sb1);
        }
}
```

Q24. .

`StringBuffer delete(int start, int end)`	Removes the characters in a substring of this `StringBuffer`.

```
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer sb1=new StringBuffer("abcxyzdef");
        sb1.delete(3,6);
        System.out.println("abcxyzdef deleted 3-5 :"+sb1);
    }
}
```

Q25.

`StringBuffer reverse()`	Causes this character sequence to be replaced by the reverse of the sequence.

```
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer sb1=new StringBuffer("STJI @ PJCS");
        sb1.reverse();
        System.out.println("Reverse :"+sb1);
    }
}
```

Q26.

`String toString()`	Returns a string representing the data in this sequence.

```
class TestStringBuffer{
    public static void main(String args[]){
        StringBuffer nicNo=new StringBuffer("856573702V");
        //String nic=nicNo;  //Compile Error
        String nic=nicNo.toString();
        System.out.println("NIC numer :"+nic);
    }
}
```

Wrapper Classes

Class	Constructors of the Wrapper Classes	
	Constructor(String)	Constructor (primitive)
Byte	new Byte(“10”)	new Byte(byte)
Short	new Short(“10”)	new Short(short)
Integer	new Integer(“10”)	new Integer(int)
Long	new Long(“34”)	new Long(long)
Float	new Float(“34.5f”)	new Float(float) new Float(double)
Double	new Double(“35.5”)	new Double(double)
Character	not available	new Character(char)
Boolean	new Boolean(“true”)	new Boolean(boolean)

Q27. **Byte Constructors**

```
class TestWByteConstructors{
    public static void main(String args[]){
        byte b=12;
        Byte b1=new Byte(b);
        Byte b2=new Byte("12");
        Byte b3=new Byte("012");
        // Byte b4=new Byte(12); Compile Error no Byte(int)
        // Byte b5=new Byte("0x12"); Runtime Error Only integer literal
        System.out.print(b1+" "+b2+" "+b3);
    }
}
```

Q28. **Short Constructors**

```
class TestWShortConstructors{
    public static void main(String args[]){
        short s=12;
        Short s1=new Short(s);
        Short s2=new Short("12");
        System.out.print(s1+" "+s2);
    }
}
```

Q29. **Integer Constructors**

```
class TestWIntegerConstructors{
    public static void main(String args[]){
        Integer x1=new Integer(12);
        Integer x2=new Integer(“12”);
        System.out.print(x1+" "+x2);
    }
}
```

Q30. **Long Constructors**

```
class TestLongConstructors{
    public static void main(String args[]){
        Long l1=new Long(23L);
        Long l2=new Long("23");
        Long l2=new Long("23L"); //Run time error
        System.out.print(l1+" "+l2);
    }
}
```

Q31. Float Constructors

```
class TestFloatConstructors{
    public static void main(String args[]){
        Float f1=new Float(23.5f);      //(float)
        Float f2=new Float(23.5);       //(double)
        Float f3=new Float("23.5f");    //(String)
        Float f4=new Float("23.5d");    //(String)
        Float f5=new Float("012.5");
        System.out.print(f1+" "+f2+" "+f3+" "+f4+" "+f5);
    }
}
```

Q32.

```
class TestDoubleConstructors{
    public static void main(String args[]){
        Double d1=new Double(2.5);
        Double d2=new Double("2.5");
        Double d3=new Double(2.5d);
        Double d4=new Double("2.5d");
        Double d5=new Double("02.5");
        Double d6=new Double("2.12E5");
        Double d7=new Double(2e-5);
        System.out.print(d1+" "+d2+" "+d3+" "+d4+" "+d5+" "+d6+""+d7);
    }
}
```

Q33. Character Constructor

```
class TestCharacterConstructors{
    public static void main(String args[]){
        Character c1=new Character('A');
        Character c3=new Character(((char)65));
        Character c4=new Character('\u0041');
        System.out.print(c1+" "+c3+" "+c4);
    }
}
```

Q34. Boolean Constructor

```
class TestBooleanConstructors{
    public static void main(String args[]){
        Boolean b1=new Boolean(true);
        Boolean b2=new Boolean("true");
        Boolean b3=new Boolean("True");
        Boolean b4=new Boolean("TRUE");
        Boolean b5=new Boolean("XYZ");
        // Boolean b6=new Boolean(True);
        System.out.println(b1+" "+b2+" "+b3+" "+b4+" "+b5);
    }
}
```

Special Methods in Wrapper classes

valueOf() – Static methods

Class	Byte	Short	Integer	Long	Float	Double	Character	Boolean
valueOf(byte)								
valueOf(short)								
valueOf(int)								
valueOf(long)								
valueOf(float)								
valueOf(double)								
valueOf(char)								
valueOf(boolean)								
valueOf(String)								
valueOf(Str, int)								

Q35. **Byte.valueOf()**

```
class ValueOfByte{
    public static void main(String args[]){
        byte b=12;
        Byte b1=Byte.valueOf(b);
        Byte b3=Byte.valueOf("12");
        Byte b6=Byte.valueOf("1011",2);
    }
}
```

Q36. **Short.ValueOf()**

```
class ValueOfShort{
    public static void main(String args[]){
        short s=12;
        Short s1=Short.valueOf(s);
        Short s2=Short.valueOf("12");
        Short s3=Short.valueOf("1100",2);
        System.out.println(s1+" "+s2+" "+s3);
    }
}
```

Q37. **Integer.ValueOf()**

```
class ValueOfInteger{
    public static void main(String args[]){
        int x=12;
        Integer x1=Integer.valueOf(x);
        Integer x2=Integer.valueOf("12");
        Integer x3=Integer.valueOf("1100",2);
        System.out.println(x1+" "+x2+" "+x3);
    }
}
```

Q38. **Long.ValueOf()**

```
class ValueOfLong{
    public static void main(String args[]){
        long x=12;
        Long x1=Long.valueOf(x);
        Long x2=Long.valueOf("12");
        Long x3=Long.valueOf("1100",2);
        System.out.println(x1+" "+x2+" "+x3);
    }
}
```

Q39. Float.valueOf(float) / Float.valueOf(String) / No Float.valueOf(String,int)

```
class ValueOfFloat{
    public static void main(String args[]){
        Float f1=Float.valueOf(123.123f);
        // Float f2=Float.valueOf(123.123);
        Float f3=Float.valueOf("123.123f");
        Float f4=Float.valueOf("123.123");
        Float f5=Float.valueOf("0123.123");
        System.out.println(f1+" "+f3+" "+f4+" "+f5);
    }
}
```

Q40. Double.valueOf(String) /Double.valueOf(double) / No Double.valueOf(String,int)

```
class ValueOfFloat{
    public static void main(String args[]){
        Double d1=Double.valueOf(123.123);
        Double d2=Double.valueOf(123.3E4);
        Double d3=Double.valueOf("123.3E3");
        Double d4=Double.valueOf("0123.123");
        System.out.println(d1+" "+d3+" "+d4);
    }
}
```

Q41. Character.valueOf() & Boolean.valueOf()

```
class ValueOf{
    public static void main(String args[]){
        Character c1=Character.valueOf('A');
        // Character c2=Character.valueOf(67); Compule Error
        Boolean b1=Boolean.valueOf(true);
        // Boolean b2=Boolean.valueOf(TRue); Compile Error
        Boolean b3=Boolean.valueOf("true");
        Boolean b4=Boolean.valueOf("TRUE");
        Boolean b5=Boolean.valueOf("ABCD");
        System.out.println(b1+" "+b3+" "+b4+" "+b5);
    }
}
```

Q42. type xxxValue()

Class	byteValue()	shortValue()	intValue()	longValue()	floatValue()	doubleValue()	charValue()	booleanValue()
Byte								
Short								
Integer								
Long								
Float								
Double								
Character								
Boolean								

Q43. booleanValue / charValue()

```
class xxxValue{
    public static void main(String args[]){
        Boolean b=new Boolean(true);
        boolean b1=b.booleanValue();
        Character c=new Character('Q');
        char ch=c.charValue();
        System.out.println("boolean value of c :"+b1);
        System.out.println("char value of c :"+ch);
    }
}
```

Q44. byteValue / shortValue() / intValue() / longValue() / floatValue() / doubleValue()

```
class xxxValue{
    public static void main(String args[]){
        Integer x=new Integer(233);
        byte b1=x.byteValue();
        int x1=x.intValue();
        float f1=x.floatValue();
        System.out.println("byte value of x :"+b1);
        System.out.println("float value of x :"+f1);
        System.out.println("int value of x :"+x1);
        Double d=new Double(1137.123);
        float f2=d.floatValue();
        int x2=d.intValue();
        byte b2=d.byteValue();
        System.out.println("byte value of d :"+b2);
        System.out.println("float value of d :"+f2);
        System.out.println("int value of d :"+x2);
    }
}
```

Auto Boxing & Auto Unboxing added by Java5.0

J2SE 5 adds to the Java language *autoboxing and autounboxing*. Autoboxing is the process by which a primitive type is automatically encapsulated(box) into its equivalent type wrapper whenever on object of that type is needed. There is no need to explicitly construct an object. Auto-unboxing is the process by which the value of a boxed object is automatically extracted (unboxed) from a type wrapper when its value is needed. There is no need to call method such as **intValue() or doubleValue().**

Q45. Boxing added to the JAVA 5.0

```
class Boxing1{
    public static void main(String args[]){
        Integer ob1=Integer.valueOf(123); //Here is Boxing
        Integer ob2=123; //Auto Boxing (Only Java5.0)
        int x1=ob1.intValue(); //Unboxing
        int x2=ob2; //Auto Unboxing (Only Java5.0)
        System.out.println(x1+" "+x2);
    }
}
```

Q46. Arithmetic Promotion

```
class Boxing2{
    public static void main(String args[]){
        Integer ob1=Integer.valueOf(123);
        int x1=100;
        int add=x1+ob1+200; //ob1 -auto unboxing
        System.out.println("x1+ob1 :"+add);
    }
}
```

Q47. Method Call & Returning

```
class Boxing2{
    static double m(Double d){
        return d+2.5;
    }
    public static void main(String args[]){
        Double d1=m(12.5);
        System.out.println("d1 :"+d1);
    }
}
```

Q48.

```
class Boxing3{
    static void m(Object ob){
        System.out.println("m(Object)");
    }
    public static void main(String args[]){
        m(new Integer(100));
        m(100);
        m(12.5);
        m(12.5f);
        m(true);
        m('A');
    }
}
```

Q49.

```
class Boxing3{
    public static void main(String args[]){
        Integer[] iob={2,3,4,5,4,3,4,6,9,8,7};  //Auto Boxing
        for(Integer x:iob){
            System.out.print(x.intValue()+" ");
        }
        System.out.println();
        for(int x:iob){    //Auto unboxing
            System.out.print(x+" ");
        }
        int[] ob={new Integer}

    }
}
```

Q50.

```
class Boxing5{
    public static void main(String args[]){
        int[] iob={new Integer(2),new Integer(34),new Integer(23)};  //Auto Unboxing
        for(Integer x:iob){  //Auto Boxing
            System.out.print(x.intValue()+" ");
        }
        System.out.println();
        for(int x:iob){
            System.out.print(x+" ");
        }
    }
}
```

Q51. Bad use of Autoboxing

```
class Boxing4{
    public static void main(String args[]){
        byte b=10;
        short s1=b;
        // Short ob=b; Here is Compile Error
        int x=100;
        double d=x;
        // Double ob2=x; Compile Error
        Double ob3=d;
        Object ob=123;
    }
}
```

Q52.

```
class Boxing5{
    public static void main(String args[]){
        Double[] dob1={23,45,67,89,90}; //Compile Error
        double[] dob2={new Integer(23),new Integer(45), new Integer(67)};
    }
}
```

Q53.

```
class Boxing5{
    public static void main(String args[]){
        Integer iob1=new Integer(90);
        Integer iob2=new Integer(90);
        System.out.println(iob1==iob2);     //Line 1
        System.out.println(iob1.equals(iob2));
        System.out.println(90==90);
        System.out.println(iob1==90);
        System.out.println(90==iob2);
        System.out.println(iob1.equals(90)); // 90 -> new Integer(90)

        Integer iob3=100;
        Integer iob4=100;

        System.out.println(iob3==iob4); //Check with line 1
        System.out.println(iob3+" "+iob4); //invoke toString()
    }
}
```

Wrapper.parseXXX()

Class	Byte	Short	Integer	Long	Float	Double	Character	Boolean
parseByte(String)								
pasreByte(Str, int)								
pasreShort(String)								
parseShort(Str,int)								
parseInt(String)								
parseInt(Str, int)								
parseLong(String)								
parseLong(Str,int)								
parseFloat(String)								
parseDouble(String)								
parseBoolean(Str)								

Q54. parseByte()

```
class ParseByte{
    public static void main(String args[]){
        String s1="123";
        String s2="128";
        String s3="0123";
        byte b1=Byte.parseByte(s1);
        // byte b2=Byte.parseByte(s2); //Runtime Error
        byte b3=Byte.parseByte(s3);
        System.out.println(b1+" "+b3);
        byte b4=Byte.parseByte("1100100",2);
        byte b5=Byte.parseByte("16",7);
        byte b6=Byte.parseByte("1B",16);
        System.out.println(b4+" "+b5+" "+b6);
    }
}
```

Q55. parseShort()

```
class ParseShort{
    public static void main(String args[]){
        String s1="1234";
        String s2="325377";
        String s3="01243";
        short ss1=Short.parseShort(s1);
        // short ss2=Short.parseShort(s2); Runtime Error
        short ss3=Short.parseShort(s3);
        System.out.println(ss1+" "+ss3);
        short ss4=Short.parseShort("101100",2);
        short ss5=Short.parseShort("12312",8);
        short ss6=Short.parseShort("abc",16);
        System.out.println(ss4+" "+ss5+" "+ss6);
    }
}
```

Convert to Binary, Octal, Hex

Only Integer and Long classes

Integer

static String	**toBinaryString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 2.
static String	**toHexString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 16.
static String	**toOctalString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 8

Long

static **String**	**toBinaryString(long i)** Returns a string representation of the long argument as an unsigned integer in base 2.
static **String**	**toHexString(long i)** Returns a string representation of the long argument as an unsigned integer in base 16.
static **String**	**toOctalString(long i)** Returns a string representation of the long argument as an unsigned integer in base 8.

Q56.

```
class Demo{
    public static void main(String args[]){
        int x=100;
        String binX=Integer.toBinaryString(x);
        System.out.println("Binary x: "+binX);
        x=-1;
        binX=Integer.toBinaryString(x);
        System.out.println("Binary x: "+binX);
    }
}
```

toString

	Byte	Short	Integer	Long	Float	Double	Character	Boolean
toString()								
toString(primitive)								
toString(primitive, int radix)								

Q57.

```
class Demo{
    public static void main(String args[]){
        Integer ob=new Integer(100);
        String s=ob.toString();

        int x=100;
        String sx=Integer.toString(x);

        String binX=Integer.toString(x,2);
        System.out.println("Binary x: "+binX);

        binX=Integer.toString(-1,2);
        System.out.println("Binary x: "+binX);

    }
}
```