The java.lang package

27) Determine the result of applying any operator including assignment operators and instanceof to operands of any type class scope or accessibility or any combination of these.

See 14th objective

28) Write code using the following methods of the java.lang.Math class: abs ceil floor max min random round sin cos tan sqrt.

The Math class is a public final class. All methods in Math package are static so you need not instantiate it. The methods are used for performing basic numeric operations such as the square root, and trigonometric functions.

abs() :
int abs(int i); //returns the absolute value of an int value.
long abs(long l); //returns the absolute value of a long value.
float abs(float f); //returns the absolute value of a float value.
double abs(double d); //returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, it returns the argument truncating the negation.

eg,

```
int i = Math.abs(−5); //i will
be 5
```

returns "5" truncating the negation.

ceil() :
Returns the smallest (closest to negative infinity) double value that is not less than the argument and is equal to a mathematical integer.

eg,

```
double c= Math.ceil(9.01); //returns 10.0
double c= Math.ceil(−0.1); //returns −0.0
double c= Math.ceil(100); //returns 100.0
double c= Math.ceil(Double.MIN_VALUE); //returns 1.0
```

floor() :
Returns the largest (closest to positive infinity) double value that is not greater than the argument and is equal to a mathematical integer.

eg,

```
double d = Math.floor(9.01); //returns 9.0
Math.floor(−0.1); //returns −1.0
```

```
Math.floor(100); //returns 100.0
Math.floor(Double.MIN_VALUE); //returns 0.0
```

max() :

If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero.

method :

int max(int i1, int i2); //returns the greater of two int values
long max(long l1, long l2); //returns the greater of two long values
float max(float f1, float f2); //returns the greater of two float values
double max(double d1, double d2); //returns the greater of two double values

eg,

```
mx = Math.max(5,10); // mx will be 10
```

min() :

Returns the smaller of two int/long/double/float values. If either value is NaN, then the result is NaN. Unlike the the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero.

method :

int min(int i1, int i2); //returns the smaller of two int values.
long min(long l1, long l2); //returns the smaller of two long values.
float min(float f1, float f2); //returns the smaller of two float values.
double min(double d1, double d2); //returns the smaller of two double values.

eg,

```
mn = Math.max(5,10); // mn will be 5
```

random() :

returns a pseudo−random number between 0.0 and 1.0

eg,

```
double r = Math.random();
```

round() :

int round(float a); //returns the closest int to the argument.
long round(double a); //returns the closest long to the argument.

methods :

- If the argument is negative infinity or any value less than or equal to the value of Integer.MIN_VALUE/Long.MIN_VALUE, the result is equal to the value of Integer.MIN_VALUE/Long.MIN_VALUE.
- If the argument is positive infinity or any value greater than or equal to the value of

Integer.MAX_VALUE/Long.MAX_VALUE, the result is equal to the value of Integer.MAX_VALUE/Long.MAX_VALUE.

eg,

```
long r = Math.round(32.6375261584);
//returns 33
```

sin() :

Returns the trigonometric sine of an angle.

```
double s = Math.sin(1.32434545);
```

cos() :

Returns the trigonometric cosine of an angle.

```
double c =
Math.cos(1.32434545);
```

tan() :

```
double t =
Math.tan(1.32434545);
```

sqrt() :

Returns the square root of a double value. If the argument is NaN or less than zero, the result is NaN.

```
double
sq=Math.sqrt(1.32434545);
```

---

29) Describe the significance of the immutability of String objects.

- Once you instantiate a String Object with a set of characters, you can't change its size or replace its characters, in effect the String becomes a constant. Strings are immutable.

String Methods :

- length() :
  - ♦ returns the number of characters in a string

  eg,
  String a ="Hello";
  System.out.println("Length" + a.length()); //prints 5

  Remember : For finding an array length you use a.length and for length of a String a.length()

- toUpperCase() :
  - ♦ converts string to upper case

    eg,
    String a="Hello";

System.out.println("Uppercase : " + a.toUpperCase()); //prints "HELLO"

- toLowerCase() :
  - ♦ converts string to lower case

- equals() :
  - ♦ compares to Strings.
  - ♦ Returns boolean.

- equalsIgnoreCase() :
  - ♦ like equals() method. But the comparison ignores case.

- charAt() :
  - ♦ returns the character at the specified index.

    eg,
    String a = "hello";
    char c=a.charAt(4); //prints "o"

- concat() :
  - ♦ Concatenates the specified string to the end of this string.

- indexOf() :
  - ♦ returns int, the index of the first occurance of character.

- lastIndexOf() :
  - ♦ lastIndexOf(int ch)
    Returns the index within this string of the last occurrence of the specified character.
  - ♦ lastIndexOf(int ch, int fromIndex)
    Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
  - ♦ lastIndexOf(String str)
    Returns the index within this string of the rightmost occurrence of the specified substring.
  - ♦ lastIndexOf(String str, int fromIndex)
    Returns the index within this string of the last occurrence of the specified substring.

    eg,
    String a = "abcdeabcdeabc";
    int b = a.lastIndexOf('b');//returns 11

- substring() :
  - ♦ Returns a string that is a substring of this string

    eg,
    String a = "abcdefg";
    String s1= a.substring(1,3);//returns "bc"
    String s2=a.substring(4);//returns "efg"

  - ♦ toString() :

    eg,
    int i=20;

Integer(i).toString();//converts integer to String

- trim() :
  - ♦ This returns the String that results from removing white space characters from the beginning and ending of the current String

    eg,
    String s=" hello ";
    s=s.trim() //returns "hello"

- For operations on strings such as concatenation Java creates String object instead of modifying the original. So Java would be constantly creating new objects, eating up memory and time. So you can use StringBuffer class because it can be modified.
- To change the contents of a String you first convert it to a StringBuffer by passing it as a parameter to the StringBuffer constructor. Then you can operate on the StringBuffer.
- After making changes you can use StringBuffer's toString() method to convert it back to String.

eg,

```
StringBuffer sbuf = new StringBuffer("hello");
sbuf.reverse(); // returns "olleh"
sbuf.insert(3, "zzz"); //returns "ollzzzeh"
sbuf.append("333"); //returns "ollzzzeh333"
```

Remember :

- You can't compare String to a StringBuffer. If you do, it would return false
- equals() is overridden only by String and Wrapper classes and not by StringBuffer.
- If you compare two StringBuffers which has same value with equals(), it will return false.

```
String s = "abcde";
StringBuffer s1 = new StringBuffer("abcde");
if(s==s1) System.out.println("=="); //compile time error.
//Can't convert java.lang String to java.lang.StringBuffer
if(s.equals(s1)) System.out.println("equals");
else System.out.println("notequal"); //prints notequal
```