Q1.Give the definition (1-2 Lines) and an example where applicable:

Object	An entity having specific characteristics(variables) and behaviour(Methods)
Abstraction	Hiding unnecessary details and presenting the required ones.
Encapsulation	Binding and wrapping up of methods and data members.
Class	Representation for a set of objects having common behaviour and characteristics.
Byte code	The java byte code is a machine language instruction set. The byte code is independent of the computer system it has to run upon.
Native Executable Code	A machine code program which can run on a specific platform (hardware/ software) only.
JVM	Set of Java programs (interpreter compiler etc.) which converts source code to byte code and is also responsible for executing the program. (JRE)
Applet	Internet Applets are small programs that are embedded in web pages and are run on the viewers' machine (which are meant to run on the internet browser) Stand Alone Applications – Meant to run on an independent machine.
Inheritance	When a class acquires properties of another class (may add a few of its own).
Polymorphism	Behaving differently under different situations. Having one name but many different shapes.
Interface	Template/Design of a class. (Table of contents of a class)

```
public interface in
{
    int x=0,y=0; //vars are given
    int input(); //function are not defined
    int add(int x, int y);
    int display();
 }
class example implements in
{       …
}
```

Class "Object"	A super class of which all classes are sub(derive) by default.
Source Code	An uncompiled java program
Object Code	A machine language conversion of the source code. 2 types – Applet, Stand alone applications.

Recursion	When a function calls itself
	public void abc()
	{
	abc();
	}

WORA	Write Once, Run Anywhere (+ do a few more properties)
IDE	Integrated development Environment (E.g. Blue J) – where we can type, edit, debug, run, organize etc. java programs.
Compiler	Which converts the all the source code to machine code before execution. (whole program)
Interpreter	Which converts and executes source code line by line.
API	Applications Programming Interface (Java Libraries) (inbuilt feature/packages)
JRE	Java Runtime Environment – What JVM creates to run a Java program.
Type Casting	Writing a data type in () before a value, variable or an expression to change its type temporarily. Sop((char)65) Used in explicit type conversion.
Suffixes	D, L and F, for Double, long and float respectively. Substitutes for type casting and put after a literal. (3.14f)
(Possible loss of precision) Type Compatibility	Trying to assign a bigger data type to a smaller data type. RHS data type >= LHS data type. (This is why float f=3.14; doesn't work)
Coercion	Implicit (automatic/by default) type conversion
Type Conversion Types	Narrowing(Demotion), Widening(Promotion)

	Done in 2 ways – Implicit, Explicit.
Object factory	Class
Immutable	Which cannot be changed directly. (Needs to be overwritten, E.g Strings)
Compound statement	2 or more statements in { }. Also called a block.
Type Promotion	Converting a data type to a higher data type. 2 types – 1. Implicit – automatically by the compiler 2. Explicit – By the user using type casting.
Unicode	Two byte character code for multilingual characters. ASCII is a subset of Unicodes.

`

Q2. Types of comments

Single Line //
Multi Line /* …
…
*/

Documentation Comments – Which are displayed by the IDE. The are multiline comments only given immediately before the function header.

Q3. What are tokens? Name the 5 types of tokens available in Java with an example each.

The smallest individual unit in a program is known as a token. The five types of tokens are- :

a) Keywords:- public, static, void, import etc. (Java's own terms)
b) Identifiers:- (variables) alphabets, digits, under score, dollar sign etc.
c) Literals:- (Constants) integer literals, floating literals, boolean literals, character literals, string literal, the null literal.
d) Punctuators (separators):- () {} [] ; ,
e) Operators = + - < >

Q4. What are escape sequences? Give any 5 escape sequences?

Codes for non-graphic or some special characters.
'\t', '\n', '\b', '\\', '\'', '\"', '\?'

Q5. ASCII codes?

A-Z = 65-90, a-z=97-122, '0'-'9'=48-57

Q6. Prepare a chart showing all the data types with their size and range.

See table with answer 12.

Q7. How can we declare a variable whose value cannot be changed in Java?

If we write the *final* keyword before the variable name, then the value of that variable cannot be changed later in the program. Any attempt of doing so will produce a compilation error.

final int a=5;

~~Q8.~~ On what do the shift and bitwise operators work in Java? Show the truth table for Exclusive OR.

– on binary equivalents,

Note that a>>b = a / b^2 and a<<b = a*b^2

A	B	&	|	^ xor
F	F	F	F	F
F	T	F	T	T
T	F	F	T	T
T	T	T	T	F

Q9. Give the value (with reason) that the variable on the left hand side will hold in the following statements

```
float a=(float)(21/2);
int b=5 * 3/4 + 4/3 + 6;
int c=100 + 10>10?10:20;
float d= '1'+ '2';
int e= 20%7 / 2*3;
```

a) float a= (float) (21/2); Output-: First integer division takes place= (float)(10). Then the answer is converted to floating type=10.0.

b) int b= 5*3/4+4/3+6; Output-:First multiplication takes place=15/4+4/3+6. Then division=3+1+6.At last addition= 10.

c) int c= 100+10>10?10:20; Output-:First addition takes place=110>10?10:20.Then ternary operator works and returns the answer= 10.

d) float d= '1'+'2'; Output-:The ASCII codes of these characters are added(49+50) and converted into floating type=99.0.

e) int e= 20%7/2*3; Output-:First mod (6/2*3), then division(3*3) and at last multiplication= 9.

Q10. Difference – (i) & and && (ii) = and == (iii) >> different from >>>?

& and && -

& / |	&& / ||
(bitwise) checks all the conditions	(short circuit) checks till the ans is confirmed
Can be applied to numbers Sop(5 & 6)	No Sop(5 && 6) – error

=	==
Assignment	Relational (equality)
Cannot be used as an expression Sop(a = 5); - Error	Can be Sop(a == 5) ;

Difference between >> and >>> Both are same for positive numbers But in case of negative numbers >>> brings the Sign bit in the number zone and produces a very Large number.

```
System.out.println(10<<2);
System.out.println(10>>2);
System.out.println(10>>>2);
System.out.println(-10>>2);
System.out.println(-10>>>2);
```

```
40
2
2
-3
1073741821
```

Q11. Define (i)Precedence and Associativity

(ii) Mixed/Fixed(pure expr.) mode?

(i) Precedence : Order of evaluation
(ii) Associativity : Order of evaluation when precedence is the same.
(iii) Mixed Mode or Impure Expressions – When operands are of different data types.
(iv) Fixed mode or pure expressions – When operands are of the same data type.

Q12. Prepare a table showing operator Precedence and Associativity.

Type	Size	Range
Char	2	0 to 65536 (Unicodes)
Byte	1	-128 to 127
Short	2	-32768 to 32767
Int	4	-2^31 to 2^31 -1
Long	8	-2^63 to 2^63 -1
Float	4	-3.4E+38 to +3.4E+38 (Prec-7)
Double	8	-1.7E+308 to +1.7E+308 (Prec-16)
Boolean	1	(only 1 bit is used) true/false

* Order as above in implicit conversion in mixed mode.
* Associativity L->R
(Precedence in next column)

- ++x first ++ then use,
- x++ first use then ++.

[]()	
++ -- ! ~ inOf	U
New	
* / %	A
+ -	
<< >> >>>	
<><=>===!=	R
& | ^ (Bitw.)	
&&	L
||	
?:	C
+= etc.	A

Q13. What are the three types of operators based on operands.

Unary, Binary, Ternary

Q14. Show the output of the following program if X & Y are integers.

```
{   int x=5;
    System.out.println(x++*2);
    //                      5
    System.out.println(++x*2);
    //                        7
    int y=x++ + ++x + x++ + ++x;
    //      7          9    9         11
    System.out.println(y);
    x+= x++ + ++x;
    //  11    13   (WILL BE ADDED TO 11) = X becomes 35
    System.out.println(x);
    System.out.println(x-- + " " + --x + " " + --x + " " + x--);
    //                        35         33        32       32
    System.out.println("Sum = "+5+10);
    // Sum = 510
    System.out.println('A'+5);
    // 70
}//
```

```
10
14
36
35
35      33       32       32
Sum = 510
70
```

Q15. State the output with reason

```
int i=1;
for( ; i<=5; i+=2);
        System.out.println(i);
```

Output- 7

Q16.Show one example each of Inifinity and NaN.

```
System.out.println(5.0/0.0);
System.out.println(Math.sqrt(-4));
```

Q17. What is the difference between the rint() and the round() function in Java?

Math.rint() returns a double type value whereas Math.round() returns an integer type value.

```
System.out.println(Math.rint(5.5));
System.out.println(Math.round(5.5));
```

```
6.0
```

6

Note that rint() works with even numbers with .5 as floor()

Q18. Explain the use with syntax

If-else, switch, ?:, for, while, do-while

if(condition)
  Statement
else
  Statement
	switch(var)
	{ case val:
	  break;
	  case val:
	  …
	  default:
	}
		(cond)?T:F	For(inti;
			cond; update)
			{ body;
			}
				Init
				While(cond)
				{ body;
				  Update;
				}
					Init
					do
					{ body;
					  Update;
					}
					While(cond);

Q19. Display R-Red, B-Blue, G-Green or undefined

```
      char c='R';
//If
      if(c=='R') System.out.println("Red");
      else if(c=='B') System.out.println("Blue");
      else if(c=='G') System.out.println("Green");
      else System.out.println("Undefined");
//Switch
      switch(c)
      {   case 'R':   System.out.println("Red");
                      break;
          case 'B':   System.out.println("Blue");
                      break;
          case 'G':  System.out.println("Green");
                      break;
          default:   System.out.println("Undefined");
      }
// ?:

System.out.println(c=='R'?"Red": c=='B'?"Blue": c=='G'?"Green" : "Undefined");
```

Q20. Three iteration constructs

int f=1;
for(int i=1; i<=5; i++)
{ f=f*i;
}
System.out.println(f);
	int f=1;
	int i=1;
	while(i<=5)
	{ f=f*i;
	  i++;
	}
	System.out.println(f);
		int f=1;
		int i=1;
		do
		{ f=f*i;
		  i++;
		}while(i<=5);
		System.out.println(f);	

Q21. Expressions in Java –

Math.sin(Math.abs(c))+(1.0/2.0)*Math.sqrt(Math.pow(a,2)+Math.pow(b,1.0/5.0));

Math.floor(a)+Math.max(b,c)/Math.log(c);

Q22. Name the four programming constructs?

Sequence – All statements are executed sequentially.

Selection – A selected block/statement is executed. If, switch, ?:(opeartor)

Iteration – A block/statement is executed repeatedly. For, while, do-while

Jump – Which take the program control to some other point in the program. (break, continue)

Q23. Compare if and ?:.

	If	?:
1	Statement	Operator used in expressions
2	Else is optional	False part compulsory
3	Body can have more than 1 statement	1 expression
4	Nesting possible	Nesting complicated

Q24. Compare if and switch.

	If	Switch
1	Can test many variables	Only one variable
2	Any data type	Int, char
3	Can use <, > == !=…	Only checks equality
4	Slower than switch	Faster switch operation
5	Else	Default
6	No fallthrough	Fallthrough (if break is absent)

Q25. What is fall-through and how can it be prevented?

Execution of all the cases after the true case.(in the absence of break)

By using *break*.

Q26. Name the elements that control a loop.

Initialization – condition – body – counter.

Q27. What are entry and exit controlled loops

Entry controlled – in which condition is tested before executing the body. E.g. for, While

Exit controlled – in which condition is tested after executing the body, this is why they run at least once. E.g. do-while.

Q28. Compare the three types of loops with an example.

For	While	Do-while

Should be used if number of times are known	Should be used if number of times are not known	Should be used if the loop should execute at least one.

for(int i=1; i<=5; i++)
{ system.out.println(i);
}
	N=256;
	while(n>0)
	{ System.out.println(n%10);
	n=n/10;
	}
		ch=Integer.parseInt(br.readLine());
		do{
		switch(ch)
		{ case 1: …
		case 2:…
		}
		While(c!=5);

Differences – Entry/Exit controlled, executed at least once

Q29. Differentiate between ordinary and labeled *break* and *continue.*

Just break and continue consider the loop in which they appear.
Labeled break and continue can be made to consider the outer loop.

```
for(int i=0; i<3; i++)
{   for(int j=0; j<3; j++)
    {   if(i==1) continue; // to j++
        if(j==1) break; // to point A
    }
    System.out.println("Point A");
    }
    System.out.println("Point B");
}
```

```
outer:for(int i=0; i<3; i++) // outer is the label
{   for(int j=0; j<3; j++)
    {   if(i==1) continue outer; // to i++
        if(j==1) break outer; // to point B
    }//j
    System.out.println("Point A");
    }//i
    System.out.println("Point B");
}
```

Q30. Differentiate between a counter and a flag.

Counter – Used for counting (E.g count++, i++)
Flag – Usually used to test of a condition ever got true (E.g. f=1 in sequential search)
Control variable – which controls a loop/switch etc.

Q31. Define and call any example function and show the following parts –

Prototype, signature, access specifier, modifier, return type, formal argument, declaration, Definition, actual arguments, name, extended name, call

Some terms related to functions are -:

e.g. -: public static void function (int a, int b)

```
{       int c=a + b;
        System.out.print(c);
 }.
```

a) Prototype-: The first line of a function is called its Prototype. For e.g. -: public static void function (int a, int b)

b) Signature-: Signature of a function is the formal arguments of the function.eg ------ (int a, int b, int c)

c) Access Specifier -: Access Specifiers are used to restrict access to the function or method. It is of following 3 types-private, protected & public.

d) Modifier -: Modifiers like static, transient, volatile tell us whether the function can be called without objects or not.
e) Return type -: They tells us the type of value returned by the function like void function returns no value, but integer function returns integer value, double as double and so on.
f) Formal argument-: Formal arguments or Formal parameters are the arguments given during function definition like public static void function (int a)- Formal argument.
g) Declaration-: same as Prototype.
h) Body-: Body of a function are the statements given inside parenthesis. It is actually the processing part of the function e.g.-
i) Statements-: same as body.
j) Header-: same as prototype.
(k) Definition: Header+Body
(l) Extended Name: Function name + Signature

Q32. Write example programs showing the difference between call by value and reference?

```
public class ValueRef1
{   private static void function(int a)//formal arg
    {   System.out.println("Function = " +a);
        a+=10;
        System.out.println("Function = " +a);
    }

    public static void main()
    {   int a=5;
        System.out.println("Main = " +a);
        function(a);//actual arg
        System.out.println("Main = " +a);
    }
}

public class ValueRef2
{   private static void function(int a[])
    {   System.out.println("Function = " +a[0]);
        a[0]++;
        System.out.println("Function = " +a[0]);
    }

    public static void main()
    {   int a[]={5};
        System.out.println("Main = " +a[0]);
        function(a);
        System.out.println("Main = " +a[0]);
    }
}
```

```
public class ValueRef3
{   private static void function(String a)
    {   System.out.println("Function = " +a);
        a=a.concat("DE");
        System.out.println("Function = " +a);
    }

    public static void main()
    {   String a=new String("ABC");//String is a class
        //String a="ABC";
        System.out.println("Main = " +a);
        function(a);
        System.out.println("Main = " +a);
    }
}

public class ValueRef4
{   private static void function(StringBuffer a)
    {   System.out.println("Function = " +a);
        a.append("DE");//like the concat fn'
        System.out.println("Function = " +a);
    }

    public static void main()
    {   StringBuffer a=new StringBuffer("ABC");
        System.out.println("Main = " +a);
        function(a);
        System.out.println("Main = " +a);
    }
}

class change
{   int a,b;

    void swap(int x, int y)
    {   int t=x;
        x=y;
        y=t;
    }

    void swap(change obj)
    {
        int t=obj.a;
        obj.a=obj.b;
        obj.b=t;
```

```
    }

    public static void main()
    {
        int x=5, y=10;
        change obj=new change();

        System.out.println(x+" "+y);
        obj.swap(x,y);
        System.out.println(x+" "+y);

        obj.a=5; obj.b=10;
        System.out.println(obj.a+" "+obj.b);
        obj.swap(obj);
        System.out.println(obj.a+" "+obj.b);
    }
}
```

Q33. What is the difference between call by value and reference and what decides it?

Changes made in formal arguments are reflected in the calling function in call by reference and not in call by value. This is because in call by value a different copy of the variable is maintained in the computer's memory but the memory is shared in call by reference.
The argument type decides the type of call. Primitive types are called by value whereas reference types (Arrays, classes, objects and interfaces) are called by reference.
This is also called pass by value or reference.

Q34. What is the difference between pure and impure functions?

Impure functions change the state of an object i.e. the values of instance variables. They are also called mutators or setters. Pure functions do not change any instance variable (may change a local variable). They are also called accessors or getters.

Q35. What is function overloading? What things should be kept in mind while overloading a function?

Defining more than 1 function with the same name.
Different Signature.
Void add(int a, int b), Void add(int a, int b, int c);

Q36. Function prototypes

(a) int change(char c)
(b) double calc(String name, int age)
(c) System.out.println(name.substring(name.lastIndexOf(' ')+1).length());

Q37. What is a constructor? Give its properties. How do we declare/ define it? Can they be overloaded?

A function with the same name as that of a class.

Properties

1. same name
2. no return type
3. used to initialize
4. called automatically
5. should be public
6. Provided by the compiler if we don't define.

```
Class-name()
{       initialization of instance variables)
}
```

Yes, it can be overloaded.

Q38. Default values

Values given to variables by java. Given only to instance variables.

Q39. Name and give an example of the types of constructors.

```
Class A(
{       int x,y;
        A() //Default constructor
        { x=0, y=0;
        }
        A(int a, int b) //Parameterized constructor
        { x=a; y=b;
        }
}
```

Q40. Show how is a parameterized constructor is passed values. (*show an object creation statement*)

Passed values when an object is created.

```
A obj=new A(10, 20);
```

Q41. When will you need to use *this* keyword inside a constructor? Show by example.

Use 1: When the formal argument or any local variable has the same name as that of the instance variable.

```
Class A(
{       int x,y;
        A(int x, int y) //Parameterized constructor
        { this.x=x; this.y=y;
        }
}
```

Use 2: If we have to call a constructor from another.

```
Class A(
{       int x,y;
        A()
        { this(0,0);
```

```
    }
    A(int x, int y) //Parameterized constructor
    { this.x=x; this.y=y;
    }
}
```

Q41(b). Differences

See constructor properties.

Q42. Define…

QUAD obj=new QUAD(5.0, 10.0);

Q43. What are temporary instances? Show an example too in your answer.

Object created using new but not stored.

new A(1,2);

Q44. Destructor –

Runs when an object is removed from the computers memory.

Q45. What are (i) Composite data types (ii) User defined data types.

(i) Composed of other data types

(ii) Created by the user.

Q46. How is the size of a composite data type determined?

By adding the sizes of its constituent members.

E.g. class

```
class A
{       int x,y;
        boolean b;
}
```

Size of class A : 4 + 4 + 1 = 9 Bytes

Q47. Give five differences between composite and primitive data types.

	Composite	Primitive
1	Contain other types	Independent
2	Variable size	Fixed Size
3	new always required	Only for an array
4	Available where defined	Available everywhere
5	Can have functions too	No

Q48. Mark (i) Declaration (ii) Instatiation & (iii) Initialization in the following statement

A obj = new A(5,7);

A obj – decl.

new – inst.
A(5,7);- init.

Q49. What is meant by Null Reference?

An object declared but not instantiated.
A obj; - Obj has null reference

Q50. How can we access methods and variables of a class outside the class?

By creating objects and writing their name after "obj.".

Q51. What are access specifiers? Draw a table showing all the access specifiers and their accessibility in the class, package, subclasses and other packages.

Access Specifier	Same Package (In another class of the same program using objects)	Subclasses (derived) (In another class of the same program using inheritance)	Other Package (using import)	Subclasses (derived) in other package
Private	N	N	N	N
Public	Y	Y	Y	Y
Protected	Y	Y	N	Y
Package Friendly Default	Y	Y	N	N

Q52. What are static variables/methods? What is the other name given to them?

Which are declared using the modifier static before them.
(i) Can be accessed using the class name (object not required)
and (ii) have just one copy in the memory. (change in 1 object will change all)
Class variables, Class Method.

Q53. Define a Stream? Differentiate between a byte oriented and a character oriented stream.

Stream – Flow of bytes (Source or destination of bytes – also called Input streams or output streams)
Byte oriented – 101010100 (Scanner)
Character oriented – ABCDE… (Keyboard)

Q54. Name the three types of errors.

Syntax – Grammatical errors for(i=1, i<10 : i++]
Logical – if(n%2==5)
Runtime – Which occur due to wrong data. c=a/b; (ok but will give an error in b is zero)

s.substring(x,y); - error if x or y are negative or y<x.
Also called Exception.

Q55. What is exception-handling? What is its advantage? What are the statements used for it? Show the keywords used for it in an example.

Exception means a run time error. When an exception occurs the program is terminated in between only. If we perform exception handling (using try & catch) then the program control goes to the catch block and continues running even if an exception occurs.
The advantage is that the program doesn't terminate. This is also called "Fault tolerance".

```
public static void example()
{
int a[]={11,12,33,54,55};
try{
   for(int i=0; i<=5; i++)
   {   System.out.println(a[i]);
   }
}
catch(Exception e)
{}
System.out.println("End of program");
}//eg1
```

(b) What is the importance of *throw* and *finally* in exception handling?
Throw – Send the program control to the catch block (without an error)
Finally – A special catch block which is always executed.

Q56. Name some predefined exceptions in Java? What can they be used for?
ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException, NumberFormatExcpetion, ArithmeticExcpetion...
These can be used to overload the catch block. We can write a separate catch block for each exception.

Q57. What are wrapper classes? What is its advantage? Comment on their names. Give some e.g. functions?
Class equivalent of a primitive data type.(which wraps a primitive data type)
More functions
Same as that of the primitive data type but starts with a capital letter. (but Integer, Character)
parseInt(), parseFloat(), toString()

Q58. How are Strings different from StringBuffers? Name a few StringBuffer functions too.

	String	String Buffer
1	Call by value	Call by reference
2	Different set of functions	Different set of functions
3	Does not change directly	Changes directly

	s=”ABC”; s.concat(“DE”); System.out.println(s); S=s.concat(“DE”); System.out.println(s);	s=”ABC”; s.append(“DE”); System.out.println(s);

Few string buffer functions – append(), reverse(), setLength(), setCharAt(), insert(), delete()

(ii) When is a StringTokenizer useful?

String Tokenizer is useful for word by word processing

```
import java.util.*;
public class example
{
public static void main()
   {
    StringTokenizer s=new StringTokenizer("This is an example line");
  //Token means a word
  System.out.println(s.countTokens());
   while(s.hasMoreTokens())
            {    System.out.println(s.nextToken());
            }
   }//main
}//class
```

Q59. Differentiate between the .equals() and .compareTo() functions.

	.equals()	.compareTo()
1	only checks for equality	Also checks for <, >
2	Returns a Boolean value	Returns an integer s1>s2 – Positive value s1<s2 – Negative value s1==s2 - Zero

Q60. What do you understand by chaining of calls? Show an example.

Calling function calls in succession using the dot operator.

System.out.println(s.concat("te").concat("rs").concat("."));

Will display “Computers.” if *s* is “Compu” originally.

Q61. What is a package? Name some predefined packages in Java.

A collection of classes. – graphics, awt, applet, net, swing, lang, io.

Q62. Which package is imported by default? Name some classes inside it.

lang.

String, Integer, Character, Math...

Q63. Show an example of creating and another of using a user defined package.

Creating:

package myPack; (at the top of any program)
Using
import mypack.*; (at the top of any program)

(b) what does a "*" mean in an import statement?
Import all the classes of the package.

Q64. Differentiate between Scanner and BufferedReader class input

Scanner	BR
Util	IO
throws not required	required
InputStreamReader not required	required
next()	read()
no parsing required	required

Q65. Use of PrintWriter class?
It is an alternative for printing. It is in *io* package and hast 2 functions print() and println().
E.g.
PrintWriter pw=new PrintWriter(System.out, true); //(true – print immediately)
pw.println("ABC");

Q66. Differentiate between scope and visibility.
Scope and Visibility: Scope means if a variable is accessible or not whereas visibility means if the variable is directly(*without using this etc.*) accessible or not.

Q67. Show an example of inheritance. (It is defined above). Name its 5 types.

```
Class A
{       int x, y;
}
Class B extends A
{       int z;
}
```

Now B will own x,y and z because of the extends keyword.
5 types- single, multilevel, multiple, hierarchical, hybrid.

Q68. Base – Super – Which gives
Derived – Sub – Which receives.

Q69. Abstract class?
A class used only as a base class and never instantiated.

Q70. What is an Array?
A collection of values/variables of the same type.

Q71. What are out of bound subscripts? (ii) Initializer list?

Indexes below 0 or >= array's length.

{ ... }

Q72. Differentiate the two searching and sorting methods.

Sorting	Bubble	Selection
1	Greatest value goes to the end	Smallest comes to the beginning.
2	Matches two consecutive elements	Matches an element with all
3	Swapping inside the inner loop	Swapping outside the inner loop
Searching	Linear	Binary
1	Checks all elements sequentially.	Starts matching with the middle element
2	Array need not be sorted.	Array should be sorted.
3	Matching range reduces by one every time only	Matching range halves every time

values of the following array after 2 passes of (i) Bubble sort (ii) Selection Sort. - 9 7 6 4 3 1

(i) 6 4 3 1 7 9

(ii) 1 3 6 4 7 9

Q73. Ans:- long a[]={5, 12, 100, -99999, 0L };

5x8 bytes=40 Bytes.

Q74. Shape obj[]=new Shape[10];

Q75. What is the *lvalue* of an array.

The memory address where the array is stored in the RAM. (Value inside a variable is called the rvalue)

Q76. Give 2 characteristics of the Java programming language.

WORA, Object oriented, Platform independent, Light weight, Heavily Facilitated.

END

---

11 Output Questions

```
public static void Q1()
{       int n=9;
        if(n<10)System.out.println(n++);
        if(n>10 || n<20)
        System.out.println(n++);
        else
        n+=10;
        System.out.println(n);
}//Q1
```

```
9
10
11
```

```
public static void Q2()
{       int i=0, a,b,c;
        a=b=c=0;
        for(i=1; i<=5; i++)
        { switch(i)
        { case 1: a++;
        case 2:
        default:
        case 3: b++;
        case 6: break;
        }
        System.out.println(a+" "+b+" "+c);
        }
}//Q2
```

```
1 1 0
1 2 0
1 3 0
1 4 0
1 5 0
```

```
public static void Q3()
{       int i=0, s=1;
        for(i=1; i<=3; i++);
        s=s+i++;
        System.out.println(s+" "+i);
}//Q3
```

```
5 5
```

```
public static void Q4()
{       int n=1, f=1;
        while(n<=4)
        f*=n++;
        System.out.println(f);
}//Q4
```
24

```
public static void Q5()
{       int a=5, b=3;
        do{ a-=++b;
        System.out.println(a +" "+ b);
        }while (a>0);
}//Q5
```
1 4
-4 5

```
public static void Q6()
{       int a=5+2*2/7%3;
        double b=10/100*25;
        double c=25*10/100;
        double d=25*.1;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        System.out.print(d);
}//Q6
```
5
0.0
2.0
2.5

```
public static void Q7()
{   System.out.println(Math.pow(27,1.0/3.0));
    System.out.println(Math.sqrt(4));
    System.out.println(Math.ceil(20.001));
    System.out.println(Math.floor(128.0));
    System.out.println(Math.rint(25.49));
    System.out.println(Math.round(15.456));
    System.out.println(Math.abs(12.75));
    System.out.println(Math.ceil(-3.3));
    System.out.println(Math.floor(-7.5));
    System.out.println(Math.rint(-3.14));
    System.out.println(Math.round(-15.456));
    System.out.println(Math.abs(-12.75));
```

```
   System.out.println(Math.max(1,2));
   System.out.println(Math.min(1.0,2.0));
}
```

```
3.0
2.0
21.0
128.0
25.0
15
12.75
-3.0
-8.0
-3.0
-15
12.75
2
1.0
```

```
public static void Q8()
{   int a=5, b=10, c=0;
   c=a++ * ++b;
   System.out.println(a+" "+b+" "+c);
   c=a++ * 2 + 2 * a++ ;
   System.out.println(a+" "+b+" "+c);
   c=b++ + ++b + b++;
   System.out.println(a+" "+b+" "+c);
   a+=a++ + ++a;
   b+=++b + b++;
   System.out.println(a+" "+b+" "+c);
   a=100; b=200; c=0;
   c=a+b>200?1:2;
   System.out.println(c);
   c=a>b?a:b/a<b?a:b;
   System.out.println(c);
   System.out.println(a>b?"ABC":"XYZ");
}
```

```
6 11 55
8 11 26
8 14 37
26 44 37
1
100
XYZ
```

```
public static void Q9()
   {   int a=5, b=7;
       System.out.println("Sum = "+a+b);
       System.out.println("Pro = "+a*b);
       System.out.println(a+b+" = Sum");
       System.out.println(5 + a>b?a:b);
       char c='A';
       System.out.println(c+a);
       c+=5;
       System.out.println(c);
       System.out.println(c--);
       System.out.println(++c);
       System.out.println('A'-'a');
       System.out.println(90+b);
   }
```

```
Sum = 57
Pro = 35
12 = Sum
5
70
F
F
F
-32
97
```

```
public static void Q10()
   {   String s="CANARA BANK";
       System.out.println(s.indexOf(s.charAt(8)));
       System.out.println(s.compareTo("bank"));
       System.out.println(s.equals(("canara").concat("bank")));
       System.out.println(s.substring(3,6));
       System.out.println(s.toLowerCase());
       s.replace("CANARA", "PUNJAB");
       System.out.println(s.startsWith("PUNJAB"));
       System.out.println(s);
       char c='\';
       System.out.println(Character.isLetterOrDigit(c));
       c='\n';
       System.out.println(Character.isWhitespace(c));
       c='A';
       System.out.println(Character.toUpperCase(c));
   }
```

```
1
-31
false
ARA
canara bank
false
CANARA BANK
false
true
A

public static void Q11()
   {   int a[]={1,2,3,4,5};

       for(int i=0; i<a.length-1; i++)
          a[i]=a[i+1];

       for(int i=0; i<a.length; i++)
          System.out.print(a[i]+" ");

       System.out.println();

       for(int i=a.length-1; i>0; i--)
          a[i-1]=a[i];

       for(int i=0; i<a.length; i++)
          System.out.print(a[i]+" ");

   }

2 3 4 5 5
5 5 5 5 5

}//class
```

Conversion Questions

if to switch

```
{     if(n==1 || n==2)
        System.out.println("Good");
   else
        if(n==3)
           System.out.println("Fair");
        else
           System.out.println("Poor");
```

if to the ternary operator

```
{ if(n>m)
   {   s=n/m;
       m++;
   }
   else
   {   s=m/n;
       n++;
```

```
}

switch(n)
{  case 1:
  case 2: System.out.println("Good");
            break;
   case 3: System.out.println("Fair");
            break;
   default: System.out.println("Poor");
}
```

```
   }
}

s=n>m? n/m++ : m/n++;
```

for to do-while

```
{
   for(int i=1; i<=5; i++)
   {   System.out.println(i*i);
   }
}


int i=1;
do
   {   System.out.println(i*i);
       i++;
   }
while(i<=5);
```

if to ternary

```
{
   if(a>b)
        if(a>c)
             g=a;
        else
             g=c;
   else
        if(b>c)
             g=b;
        else
             g=c;
}

g= a>b? (a>c?a:b) : (b>c?b:c);
```

while to for

```
{  count=0;
   n=256;
   while(n>0)
   {   d=n%10;
       count++;
       n/=10;
   }
}

count=0;
for(n=256; n>0; n/=10)
   {   d=n%10;
       count++;
   }
```

switch to ternary

```
{    switch(n%2)
   {   case 0: System.out.println("Even");
               break;
       default:System.out.println("Odd");
   }
}


System.out.println(n%2==0?"Even" : "Odd");
```