1. What is Java..?

- Java is a programing language
- Java is based on OOPS concept
- OOPS mean Object oriented programing structute
- Java is introduced by SunMicro System

---

2. What are the datatype..?

- String - Declare as String
- Integer - Declare as int
- Boolean - Declare as boolean
- Charactor - Declare as char
- Float - Declare as float

---

3. How to declare the variable..?

```
DataType   variableName  = value ;
```

```
String name = "Java";
 int age = 56;
 char initial = 'A';
```

---

4. How to print the variable..?

We can print any value by using System.out.println()

```
String name="Java";

System.out.println(name);
```

---

5. What is String..?

- String is one of the Datatype
- It is used to declare the values with collection of characters
- We are declaring the String value by using double quotes(" ")

```
String topic = “Java”;
```

---

6. What is integer..?

- Integer is one of the datatype
- It is used to declare the value with only numbers
- We can declare the integer directly

```
int age = 30;
```

---

7. What is boolean..?

- Boolean is one of the datatype
- It is used to declare the value true/false
- We can declare the boolean value directly

```
Boolean flagTrue = true;
Boolean flagFalse = false;
```

---

8. What is Character..?

- Character is one of the datatype
- It is used to declare the value with single character
- We can declare the character value with single quote('D')

```
char initial = 'A';
```

---

9. How to print any statement..?

```
System.out.println("This is my name")
```

---

10. How to create main method..?

- Main method is a default method in the Java
- It is used to execute the class

```
public static void main (String [] args )
  {
      // Body of the method
  }
```

---

11. How to create a class..?

We can create a class by using class keyword in lowercase.

```
public class ClassName
  {
      // Body of the class
  }
```

---

12. How to create an object for the class..?

```
ClassName objectName = new ClassName();
```

Example: Created object for Seenium class

```
Selenium sel = new Selenium();
```

13. What are the control statement…?

- Control statement is used to control the flow of execution of the program
- There are different types of control statement
    - If statement
    - If else statement
    - else if statement
    - For loop
    - While loop
    - Switch

---

13. Explain If condition:

- It is one type of control statement
- It is used to check the condition if true or false
- If the condition is true mean, the body of the if block will execute.

```
if(condition)
        {
                // Body of if block
        }
```

---

13. Explain If condition:

- It is one type of control statement
- It is used to check the condition if true or false
- If block gets executed once the condition is true

```
if(condition)
        {
                // Body of if block
        }
```

---

14. Explain If else condition:

- It is one type of control statement
- It is used to check the condition if true or false
- If block gets executed once the condition is true
- Else block gets executed once the condition is false.

```
if(condition)
        {
                // Body of if block
        }
 else
        {
                // Body of else block
        }
```

15. What is else if statement..?

else if statement is used to specify a new condition if the first condition is false.

```
if(condition)
        {
            // Body of if block
        }
else if(condition)
        {
            // Body of if else  block
        }
 else
        {
            // Body of else block
        }
```

---

16. What is for loop statement..?

- For loop is used to perform the perticular action for repeated time.
- The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

Syntax:

```
for(initialization; condition; increment/decrement)
          {
            //statement or code to be executed
           }
```

Example:

```
for(int I=0; I<10; I++)
      {
          System.out.println("Java")
      }
```

1. **Initialization**: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.
2. **Condition**: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.
3. **Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.
4. **Statement**: The statement of the loop is executed each time until the second condition is false.

17. What is while loop statement..?

- The Java *while loop* is used to iterate a part of the program repeatedly until the specified Boolean condition is true.
- As soon as the Boolean condition becomes false, the loop automatically stops.
- The while loop is considered as a repeating if statement. If the number of iteration is not fixed, it is recommended to use the while loop.

Syntax:

```
while (condition)
      {
            //code to be executed
             Increment / decrement statement  ;
      }
```

Example:

```
int a=10;

while(a==10)
      {
            System.out.println("java");
             a++;
      }
```

---

18. What is break keyword..?

- When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
- The Java break statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.
- We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

Syntax: break;

Example:

```
for (int i=0; i<10; i++)
 {
          If (i==10)
                {
                   break;
                }
 }
```

19. What is the static method..?

- Static method is created with the help of "**static**" keyword.

```
public static void tester()
 {
        // body of the method
 }
```

---

20. How to access the static method from other class or same class..?

- We can access the static method by using it's class name and variable name.

Syntax:

```
ClassName.methodName();
```

Example:

```
class  Job
    {
    public static void main(String[]args)
        {
            Job.tester();
        }
    public static void tester()
        {
                // body of the method;
        }
    }
```

---

21. What is the static variable..?

- Static variable is created with the help of "**static**" keyword.

```
static String name = "tester";
```

---

22. How to access the static method from other class or same class..?

- We can access the static method by using it's class name and variable name.

Syntax:

```
ClassName.methodName();
```

Example:

```
class  Job
    {
      static String name = "tester";

    public static void main(String[]args)
        {
            Job.name;
        }
   }
```

23. What is the non-static method..?

- Non-Static method is created with the help of "**static**" keyword.

```
public void tester()
 {
        // body of the method
 }
```

---

24. How to access the non-static method from other class or same class..?

- We can access the Non-static method by creating the object for the class.

Syntax:

```
ClassName objectName = new ClassName();
objectName.tester();
```

Example:

```
class  Job
    {
    public static void main(String[]args)
        {
            Job job = new Job();
            Job.tester();
        }
    public void tester()
        {
                // body of the method;
        }
    }
```

---

25. What is the non-static variable..?

- Non-Static variable is created with the help of "**static**" keyword.

```
String name = "tester";
```

---

26. How to access the non-static method from other class or same class..?

- We can access the non-static method by creation of object for the class.

Syntax:

```
ClassName objectName = new ClassName();
objectName.tester();
```

Example:

```
class  Job
    {
      static String name = "tester";
```

```
    public static void main(String[]args)
        {
            Job job = new Job();
            Job.name;
        }
}
```

---

27. What is an identifiers..?

- identifiers in Java are symbolic names used for identification.
- They can be a class name, variable name, method name, object name.

Rules:

1. We should not special characters except underscore(_)
2. Spaces should not use in identifiers
3. Number should not use at starting letter

Class Name:

1. First letter should start with capital letter
2. Every word should start with capital letter

Example: StudentsNameList

Method Name:

It should write based on the camel case

Example: studentNameList()

Object Name:

It should write based on the camel case

Example: studentNameList

Variable Name:

It should write based on the camel case

Example: studentNameList

28. What is Error…?

- An Error is a syntax mistake in the Java
- An error is occurs when compiling the programe.

---

29. What is an Exception..?

- An Exception is an un-expected event which is occurs at Runtime.
- The programme gets terminated once the exception occurs
- Exception is one of the Class in the Java.

---

30. What are the types of Exception…?

An Exception is classified into 2 types.

- Checkecd Exception
- Un-checked Exception

1. **Checked Exception:**
   - The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked **exceptions**.
   - Checked exceptions are checked at compile-time.
   - **Example:** IOException, SQLException

2. **Un-Checked Exception:**
   - The classes that inherit the RuntimeException are known as unchecked exceptions
   - Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
   - **Excample:** ArithmeticException, NullPointerException

---

31. How can we handle an Exception…?

We can handle the Exception in 2 ways.

a). By using **throws** keyword

b). By using **try, catch, finally** block

a) By using throws keyword:

- We can use throws keyword in the method to handle an exception

```
public static void method() thorows IOException
{
        Thread.sleep(2000)
}
```

b) By using try, catch, finally block:

- We can use try, catch, finally block to handle an Exception mannually.

1. <u>try:</u>

- Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.
- If an exception occurs at the particular statement in the try block, the rest of the block code will not execute. So, it is recommended not to keep the code in try block that will not throw an exception.
- Java try block must be followed by either catch or finally block.

2. <u>catch:</u>

- Catch block get's executed once the exception is occcured in the try block
- Control gets moved from try block to catch block once the exception occurred
- We can add multiple catch block after the try block
- We should pass the Exception name as the argument in the catch block
- It is an alternative block if the exception is occurred.

3. <u>finally:</u>

- **Java finally block** is a block used to execute important code such as closing the connection, etc.
- Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.
- finally block in Java can be used to put "**cleanup**" code such as closing a file, closing connection, etc.
- The important statements to be printed can be placed in the finally block.

Syntax:

```
try
  {
      // body of try block
  }
catch(Exception e)
    {
            // body of catch block
    }
finally
  {
     // body of finally block
  }
```

32. What is an Inheritance…?

- Inheritance is nothing but once class is acquired the properties of another class
- Properties mean the variable and method of the another class
- Inheritance is done by using **extends** keyword
- We can classified the classes in inheritance into 2 types
    1. Super class (parent class)
    2. Sub class (child class)
- We can not inherite the final class
- We can not inherite the Non-static and private members

Types of Inheritance:

The inheritance is classified into differennt types

1. Single Inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Multiple inheritance
5. Hybrid inheritance

1. Single Inheritance:

- In single inheritance, subclasses inherit the features of one superclass.
- In below example, class Daughter is inherited the only one class named Mother.

```
class Mother
{
    public static void saree()
        {
            System.out.println("Saree")
        }
}


class Daughter extends Mother
{
    public static void main(String [] args)
        {
           Saree();
        }
}
```

2. **Multi-level Inheritance:**

- In multi-level inheritance, one class (child) inherites the properties of another class (parent) and the same class inherited by other class

```
class Parent
 {
     public static void saree()
          {
              System.out.println(“Saree”)
          }
 }


class Child extends Parent
 {
     public static void main(String [] args)
          {
             Saree();
          }
 }


class GrandChild extends Child
 {
     public static void main(String [] args)
          {
             Saree();
          }
 }
```

3. Hierarchical inheritance

- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass

```
class A
{
  public void print_A()
        {
            System.out.println("Class A"); \
          }
   }

class B extends A
{
  public void print_B()
        {
            System.out.println("Class B");
        }
}

class C extends A
{
  public void print_C()
        {
             System.out.println("Class C");
        }
}

class D extends A
{
   public void print_D
        {
            System.out.println("Class D");
        }
}
```

4. Mutliple Inheritance:

- In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.
- Please note that Java does not support multiple inheritances with classes.
- In java, we can achieve multiple inheritances only through Interfaces.

```
interface one
        {
           public void print_geek();
        }

interface two
         {
           public void print_for();
         }

interface three extends one, two
          {
           public void print_geek();
           }

class child implements three
{
    @Override
        public void print_geek()
        {
          System.out.println("Geeks");
        }

     public void print_for()
          {
                System.out.println("for");
          }
    }
```

5. Hybrid Inheritance:

- Hybrid Inheritance(Through Interfaces): It is a mix of two or more of the above types of inheritance.
- Since java doesn't support multiple inheritances with classes, hybrid inheritance is also not possible with classes.
- In java, we can achieve hybrid inheritance only through Interfaces.

Important terminology used in Inheritance:

Default superclass: Except Object class, which has no superclass, every class has one and only one direct superclass (single inheritance). In the absence of any other explicit superclass, every class is implicitly a subclass of the Object class.

Superclass can only be one: A superclass can have any number of subclasses. But a subclass can have only one superclass. This is because Java does not support multiple inheritances with classes. Although with interfaces, multiple inheritances are supported by java.

Inheriting Constructors: A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

Private member inheritance: A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods(like getters and setters) for accessing its private fields, these can also be used by the subclass.

-

33. What is Java Comment..?

- Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.
- Java comment is classified into 2 types
  1. Single line comment
  2. Multi line comment

1. **Single line Comment:**
   - Single-line comments start with two forward slashes (//).
   - Any text between // and the end of the line is ignored by Java (will not be executed).

Example:

```
// This is a comment
System.out.println("Hello World");
```

2. **Multi line Comment:**
   - Multi-line comments start with /* and ends with */.
   - Any text between /* and */ will be ignored by Java.

Example:

```
/* The code below will print the words Hello World
   to the screen, and it is amazing */
   System.out.println("Hello World");
```

---

34. How to covert data from integer into String value…?

We can convert from integer value to String value by adding empty string into integer value

```
int age = 30;
String newAge = age+"";
System.out.println(newAge);
```

---

35. How to convert data from String into Integer…?

We can convert from String value to integer value by parseInt() method from Integer class

```
String age = "22";
int newAge = Integer.parseInt(age);
System.out.println(newAge);
```

36. How to get the value from Run time…?

We can get the value from Run time using Scanner class.

Code Steps:

1. Create an object for Scanner class by passing the **System.in** as an argument
2. Call the method from object based on the input value

nextInt(): To get the Integer value

next(): To get the String value

nextBoolean(): To get the Boolean values

```
Scanner input = new Scanner(System.in);
int age = input.nextInt();
String name = input.next();
boolean value = input.nextBoolean();
System.out.println(age);
System.out.println(age);
System.out.println(age);
```

---

37. What is the String class…?

- String is one of the class in the Java
- Strings are used for storing text.
- A String variable contains a collection of characters surrounded by double quotes:

Example:

```
String greeting = "Hello";
```

---

38. What are the methods available in String class..?

- Length()
- toUpperCase()
- toLowerCase()
- contains()
- equals()
- charAt()
- indexOf()
- subString()
- isEmpty()
- trim()
- startsWith()
- endsWith()
- replace()
- concat()
- Split()

39. What is length() method in String class..?

- The length() is used to find the number of characters in the String value
- It returns the length value in integer

```
String name = “Java”;
int totalCharacter = name.length();
```

---

40. What is toUpperCase() method in String class..?

- The toUpperCase() method is used to convert the String value into fully upper case.
- It returns the value in String

```
String name = “Java”;
String convertedUpperCaseValue = name.toUpperCase();
```

---

41. What is toLowerCase() method in String class..?

- The toLowerCase() method is used to convert the String value into fully lower case.
- It returns the value in String

```
String name = “Java”;
String convertedLowerCaseValue = name.toLowerCase();
```

---

42. What is contains() method in String class..?

- The contains() method checks whether a string contains a sequence of characters.
- It returns the boolean value either true or false.

```
String name = “Java”;
boolean value= name.contains(“ava”);
```

---

43. What is equals() method in String class..?

- The equals() method compares two strings, and returns true if the strings are equal
- It returns the boolean value either true or false.

```
String name = “Java”;
boolean value= name.equals(“Java”);
```

44. What is indexing in Java..?

- Indexing means to given number to sequence of value
- In Java, The Index value is starting from zero(0)

---

45. What is charAt() method in String class..?

- The charAt() method returns the character at the specified index in a string.
- The index of the first character is 0, the second character is 1, and so on.
- It returns the Character value.

```
String name = "Java";
char value= name.charAt(0);
```

---

46. What is indexOf() method in String class..?

- The indexOf() method returns the position of the first occurrence of specified character(s) in a string.
- It returns the index value in Integer

```
String name = "Java";
int value= name.indexOf('v')
```

---

47. What is subString() method in String class..?

- The subString() method is used to get the some part of the String value.
- It returns the part of String value in String type.
- We need to mention start index and end index of the String value.

```
String name = "Automation Engineer";
String outputValue = name.subString(0,3);
```

---

48. What is isEmpty() method in String class..?

- The isEmpty() method is used to verify whether the string is empty or not
- It returns the boolean value either true or false.

```
String name = "";
boolean value= name.isEmpty();
```

49. What is trim() method in String class..?

- The trim() method is used to remove the spaces in first and last of the String value.
- It returns the trimmed string value.

```
String name = "    java     ";
boolean value= name.trim();
```

---

50. What is startsWith() method in String class..?

- The startsWith() method checks whether a string starts with the specified character(s).
- It returns the value in boolean type.

```
String name = "Java";
boolean value= name.startsWith("Ja");
```

---

51. What is endsWith() method in String class..?

- The endsWith() method checks whether a string ends with the specified character(s).
- It returns the value in boolean type.

```
String name = "Java";
boolean value= name.endsWith("va");
```

---

52. What is replace() method in String class..?

- The replace() method searches a string for a specified character, and replace it.
- It returns a new string where the specified character(s) are replaced.

```
String name = "Java";
String value= name.replace( 'a' , 'e' );
```

---

53. What is concat() method in String class..?

- The concat() method appends (concatenate) a string to the end of another string.
- It returns a new string value.

```
String name = "Java";
String language = "Language";
String value= name.concat(language);
```

54. What is split() method in String class..?

- The split() method is used to split the string value into individual character as array
- It returns a Array value

```
String name = "J a v a";
String [] array = name.split(' ')
```

---

55. What is an Array…?

- An Array is used to store the multiple values in a single variable,
- We can only store the same type of value in the variable
- To declare an array, define the variable type with **square brackets**:

Syntax:

```
DataType [] variableName = { value1, value2 };
```

Example:

```
String [] name = { "Java" , "Python" , "Oracle" }
int [] age = { 10, 20, 30 };
char [] firstLetter = { 'J' , 'P' , 'O' };
```

---

56. How to access the value from an Array..?

- We can access the value from an Array by using it's Index value

Example:

```
String [] name = { "Java" , "Python" };
String java = name[0];
```

---

57. How to find the value of an Array..?

- We can find the value of an Array by using lengh property.
- It will return the value in Integer.

Example:

```
String [] name = { "Java" , "Python" };
int count = name.length;
```

58. How to change the value in an Array..?

- We can change the value by using it's index value
- It will override the existing value

Example:

```
String [] name = { "Java" , "Python" };
name[0] = "Python";
```

---

59. How to loop through an Array..?

- We can loop through an array by using for loop.

Example:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};f

for (int i = 0; i < cars.length; i++)

        {

         System.out.println(cars[i]);

        }
```

---

60. What is method..?

- A **method** is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- Methods are used to perform certain actions, and they are also known as **functions**.
- Why use methods? To reuse code: define the code once, and use it many times.

---

61. How to create a method..?

- A method must be declared within a class
- It is defined with the name of the method, followed by parentheses **()**.

Example:

```
public class Java
    {
        static void myMethod()
            {
                // code to be executed
            }
    }
```

myMethod() : Name of the Method

static : It is define that it is static method

Void : It is a return type

---

62. How to execute a method..?

- We can execute the method by calling the method name followed by two parentheses () and a semicolon(;)
- A method can also be called multiple times:

Example:

```
myMethod();
```

---

63. What are Parameter and Arguments..?

- Information can be passed to methods as parameter. Parameters act as variables inside the method.
- Parameters are specified after the method name, inside the parentheses.
- We can add as many parameters as we want
- The multiple parameters are separated by comma(,)
- When a parameter is passed to the method, it is called an argument.
- Note that when you are working with multiple parameters, the method call must have the same number of arguments as there are parameters, and the arguments must be passed in the same order.

Example:

```
public static  void myMethod(String name, int age)
    {
      System.out.println(name);
      System.out.println(age);
    }
```

---

64. What is Return in Java..?

- Return mean something returned by a method
- If the method is not returning any value, we should use **void** as return type
- If we want to return any value from the method, we should use primitive data type in the method declaration.
- Example: String, int, char, boolean.
- Use return keyword to mention which variable is returned from the method.

Example:

```
public static String name()
    {
        String name = "Java";
        return name ;
    }
```

```
public static int age()
    {
        Int age = 100 ;
        return age ;
    }
```

---

65. What is Method overloading in Java…?

- Method overloading is nothing but creating the multiple method in the same name
- We can create a same name method by differentiate the argument name and it's order

Example:

```
public static void name()
 {
   // code
 }

public static void name(String name)
 {
   // code
 }

public static void name(String name1, String name2)
 {
   // code
 }
```

---

66. How can varry the argument in the Method over-loading ..?

- We can varry the argument by using 3 ways
    1. Number of Argument (String name1, String name2)
    2. Sequence of Argument (String name2, String name1)
    3. Different datype (String name, int age)

---

67. What is Method declaration…?

- Declaring the method alone without body is called as Method declaration.

Example:

```
public static void name()
 {

 }
```

68. What is Method definition…?

- Declaring the method with exection content in the body is called as Method definition.

Example:

```
public static void name()
 {
     System.out.println("This is name");
 }
```

---

69. What is Method over-riding in Java…?

- Method over-riding is nothing but inherite the other class and change it's method definition
- We can achieve the method over-riding by using concept of Inheritance.

Example:

```
public class Language
  {
     public static void language()
       {

       }
  }


 public class Java extends Language
  {
     public static void language()
       {
          System.out.println("Java");
       }
  }
```
