Q. What is Java?

a. Java is a general purpose language.
b. It means Java language can be used to develop different types of applications.

Q. Explain. The different versions of Java programming language.

a. Java programming language has 3 different versions:
1. JSE(Java Standard Edition)
2. JSE is also known as core Java.
3. JSE can be used to develop desktop applications.

J2EE(Java Enterprise Edition): J2EE is also known as advanced Java.
J2EE can be used to develop web applications.

JME(Java Micro Edition): JME is also known as Android edition. JME can be used for the development of an android application.

Q. What are the features of Java programming language?

A. Simple, Java is a simple programming language because we can easily create and maintain Java syntax. It is possible to learn Java without having knowledge of other programming languages.
B. Platform independent, Java is platform independent language it means we can execute Java programs by using any OS platform.
C. Portable Java is a portable programming language. It means we can switch the development environment it means from one operating system to another operating system, in other words we can develop Java applications in various platforms.
D. Object-oriented, Java is an object oriented programming language because we can represent each and everything in terms of objects.
E. Robust Java is a robust programming language because of it's memory management. In Java memory management can be done implicitly as well as explicitly.

Q. Explain the process of coding compilation and execution in Java programming language.
Every programming language will be having 3 common steps

- Coding: the process of writing Java statements inside the .Java file is known as coding. The file which contains the source code is also known as source file.
- Compilation:After coding we have to compile the .java file. Java compiler is responsible for compilation.

The Java compiler performs the following activities in order to complete the compilation process.

1. Checks the syntax and rules of language.
2. Converts Java statements into the byte code.

After successful compilation the compiler will generate . Class file.

- Execution: JVM is responsible for the execution of .class file.

To complete the execution process we have to install the JDK software.

Q. Explain the different environments present in the IT industry.

A. Following are the environments present inside the IT industry.

- Development environment.
- Testing environment.
- Production environment.

1. Developers are responsible for coding, compilation and execution processes. In the development environment we have to install JDK software.
2. Testers are responsible for the execution of software applications. In a testing environment we have to use only JRE.
3. In production environments beta testing will take place. We have to install only JRE in the production environment.

Q. What is JDK?

A. JDK stands for Java development kit. It is the collection of development related components, for example: compiler, JVM, JIT. JDK is an installable software for the Java programming language.

Q. What is JVM?

A. JVM stands for Java virtual machine. JVM is responsible for the interpretation of byte code. In other words JVM is the component which understands the meaning of code. JVM is responsible for the execution of class files.

Q. What is JIT?

A. JIT stands for Just In Time compiler. JIT is responsible for converting byte code into Machine code in order to improve the performance of Java programs.

Q. What is JRE?

A. JRE stands for Java Runtime Environment. JRE provides an execution environment for JVM in order to execute .class files. Without JRE JVM cannot execute the class file.

Q. Explain. Configuration process of JDK.

A. JDK is an installable software for the Java programming language. After installation it is mandatory to configure Java components. In order to configure Java components we have to create path variables by using advanced system settings.

Q. What is a path variable?

A. Path variable represents location of different Java components. We have to provide the location of the bin folder present inside the JDK as below for the path variable.
B. Following are the steps to set path variable:
    - Go to C:-> Program Files/ Java/ JDK/ bin. Copy this path.
    - Go to this PC properties-> advanced system settings-> environment variables.
    - Create a new variable as path and paste the copied path as value for that variable.

- **Java is a procedure oriented programming language hence we have to follow a specific process for the execution of a Java program.**

Q. Explain the structure of the Java program.

A. Class class_name
```
{
 public static void main (string[] args)
 {
  Java statements;
 }
}
```

What is the use of + operator?

A. + Operator can be used to perform addition and concatenation. We can pass different types of input for + operator. If the expression contains only numeric values, only character values or else numeric and character values then addition will take place.
B. If the expression contains string value JVM will perform concatenation.
C. Concatenation is the process of joining two different inputs (concatenation is applicable only for string input).

Q. What is a data type?

A. Data type is used for storing a specific value of type.
B. Data types can be used to refer to different types of values.
C. In Java data types are classified into 2 type: 1 primitive
            2 non-primitive

Q. Explain primitive data type available in Java.

- Java language has provided 8 primitive data types:
  1 int : 4 bytes.
  2 float: 4 bytes.
  3 double: 8 bytes.
  4 long(int): 8 bytes.
  5 short(int): 2 bytes.
  6 Boolean: 1bit
  7 Byte: 8 bits.
  8 char: 2 bytes.

Q how to store string values in Java?

A. We have to use a predefined class named String.
B. String is not a primitive data type, so it is considered as a non-primitive data type.

Q. What is a variable? Explain the usage of variables.

- Variables are used to store program data.
- Variables act as a container for different types of program information.
- To store the data in terms of variables, we have to follow 3 steps:
    1. Declaration
    2. Initialisation.
    3. Re-initialization.

- The process of creating a variable by using an appropriate data type is known as declaration. Data type variable name;

- The process of assigning the initial value to an already declared variable is known as initialization. Variable name= value;

- The process of assigning different values to the already initialised variable is known as re-initialization.
  Variable name= value;

Q. Explain the difference between keywords and identifiers.

- Keywords:
    1. All the keywords are predefined words.
    2. The words provided by Java language are known as keywords.

3. Every keyword in Java has a specific meaning.
4. All the keywords are reserved words because we cannot use keywords as a variable name, class name or method name.
5. It is mandatory to declare all the keywords in lowercase format. For example: class, public, static, int, etc are keywords.

- Identifiers:
  1. Identifiers are the words created by developers.
  2. Variable name, class name and method name will be considered as identifiers.
  3. To create an identifier we have to follow some rules: 1. Every identifier must and should contain alpha or alpha-numeric.
     2. Identifier name should start with alphabet or else _.
     For example: ABC123 is valid.
     123abc is invalid.

Q. What is the use of method in Java programming?

A. Method is a collection of executable Java statements. Java methods can be used to develop business logic. The main advantage of using a method is reusability, it means "**Write once, execute multiple times**".

Q. Explain the types of methods advisories in Java language.

A. In Java methods are mainly classified into 2 types:
   1. Internal method.
   2. External method.

- Internal method:
  1. Methods provided by Java language are known as internal methods.
  2. JVM is responsible for the execution of internal methods.
  3. It is not possible to modify the signature of an internal method.

Method structure:

Public(Access modifier (optional for external methods)) **static**(non-access modifier) **void** (return type) **main** (method name) **(string [] args)**(parameters).

- External methods:
  1. Methods created by developers are known as external methods.
  2. Developer is responsible for the execution of external methods.
  3. It is mandatory to call the external method inside the main method.
  4. Developers can create any number of external methods.

Q. Explain the return type of method.

1. Every method in Java must have a return type.
2. If the return type is void it means the method cannot return any value.
3. We can transfer the data from one method to another method but using a return statement.
4. If the method returns the value then we have to change the return type of a specific method.
5. Return type should be similar to the data type of value which is going to be returned by method.
6. At a time method can return only one value, because we cannot use multiple return statements.
7. Multiple return statements are not allowed because a method should have only one return type.

Q. What is the use of Scanner class?

1. Scanner class is an internal class which can be used to accept input from the end user.
2. We can accept different types of input by using Scanner class.
3. To Accept the input from user we have to follow 3 steps
    - We have to import the scanner class from the java.util package.
      Import java.util.Scanner;
    - After importing the package, we have to create a reference of scanner
      Scanner s = new Scanner(System.in);
    - After creating a reference of scanner class we have to call internal methods of scanner class to different types of input.
4. Types of Scanner methods:
    - next() : For String and character input.
    - nextInt(): for Integer input.
    - nextfloat/double(): for Decimal.
    - nextBoolean(): for Boolean input;
    - nextByte(): for Byte input;

Q. What is the use of control statements?

A. Control statements can be used to apply conditions or restrictions on Java statements. Developers can control the flow of execution of a program by using different control statements. In Java control statements are classified into 2 categories.
    1. Decision making statement.
    2. Looping statement.

Q. What is meant by a decision making statement?

A. Decision making, it means programmers can decide execution of Java statements based on specific conditions. In Java "if statement" is used for decision making operations.

Q. Explain the working of if and else statements?

A. If statements always accept conditions from the developer, if the condition is true then respective Java statements will be executed. If statement always returns Boolean type of value i.e true or false. Following are the basic scenarios of if else statement:

1.
```
if(condition)
{
  Java statements;
}
```

2.
```
if(condition)
{
  Java statements;
}
else
{
 Java statements;
}
```

3.
```
if(condition)
{
  Java statements;
}
else if(condition)
{
 Java statements;
}
.
.
.
else
{
  Java statements;
}
```

Q. Explain concept of members.

A. 1. Anything declared inside the class body will be considered as members of a class.
2. Members are mainly classified into 2 types: data member, function member.
3. Data members are nothing but variables declared inside the class.
4. Methods of the class will be considered as function members.
5. Members of the class body can be static or non static.

MEMBERS

DATA MEMBERS:

1. STATIC DM.
2. Non-STATIC DM

FUNCTION MEMBERS:

1. STATIC METHOD.
2. NON STATIC METHOD.

Q. Is it possible to create multiple classes inside the single .java file?

1. Yes it is possible to create multiple classes inside the single .Java project.
2. In such a case, only one class should contain the main method.
3. The class which contains the main method is known as main class or executable class.
4. The class which does not contain the main method will be considered as business logic class or functional class.
5. Filename should be the same as a class name which contains the main method.

Q. Explain the static members of the class.

A. Members declared along with static keywords are known as static members of the class.
B. Static means a common accessible resource (everyone can access the same).
C. Static members will be having a single copy into JVM memory, to access static members we have to provide a class name as reference.

Accessibility of members:

- Static context.
  1. Static members (without any reference inside the same class)
  2. Static members (with reference class name, outside the same class)
  3. Non-static members (with reference to an object, same & different class).

- Non static context:
    1. Non static members (without any reference inside the same class).
    2. Non static members, with reference outside the class.
    3. Static members (same class= no reference required)
       diff class: class name reference.

Q. Explain non static members of a class.

A. Members declared without using static keywords are known as non static members.
B. Non static members will be having multiple copies inside the JVM memory.
C. To access non static members outside the class or else inside the static context we have to create an object.
D. We can create an object by using a new operator.

Q. Explain concept of reference variable.

A. Reference variable is a class type variable, which holds the address of an object.
B. Reference variables can be used to access non-static members of a specific class.
C. Default value of every reference variable is always null.
D. If we print the reference variable, the address of an object will get printed.
E. We can create multiple reference variables for a single object.
F. It is possible to copy the address of one reference variable into another reference variable.

Q. What is an object?

A. An entity having its own state and behaviour is known as an object.
B. State represents characteristics of an object, whereas behaviour represents functionality.
C. In terms of Java, an object is a copy of non-static data members and function members.
D. Non static data members represent the state of an object, whereas non static function members represent behaviour of an object.
E. Class is a container for object state and behaviour, hence we have to create an object of a class.
F. In Java an object is also known as an instance of class.

Q. Explain use of Java operators.

A. Operators are mainly used to perform operations on data.
B. Java operators play an important role in order to control program execution.
C. In Java, operators are classified into 6 categories:
    1. Arithmetic
    2. Assignment
    3. Logical
    4. Relational

5. Unary
6. Bit-wise.

- Arithmetic operators: Are responsible to perform arithmetic operations in Java programs. Following are the arithmetic operators available:

Operator Name	Usage & input	Example
+	Addition and concatenation	C=a+b
-	Subtraction	C=a-b
*	Multiplication	C=a*b
/	Division	C=a/b
%	Modulus/ find remainder	C=a%b

- Assignment operators:

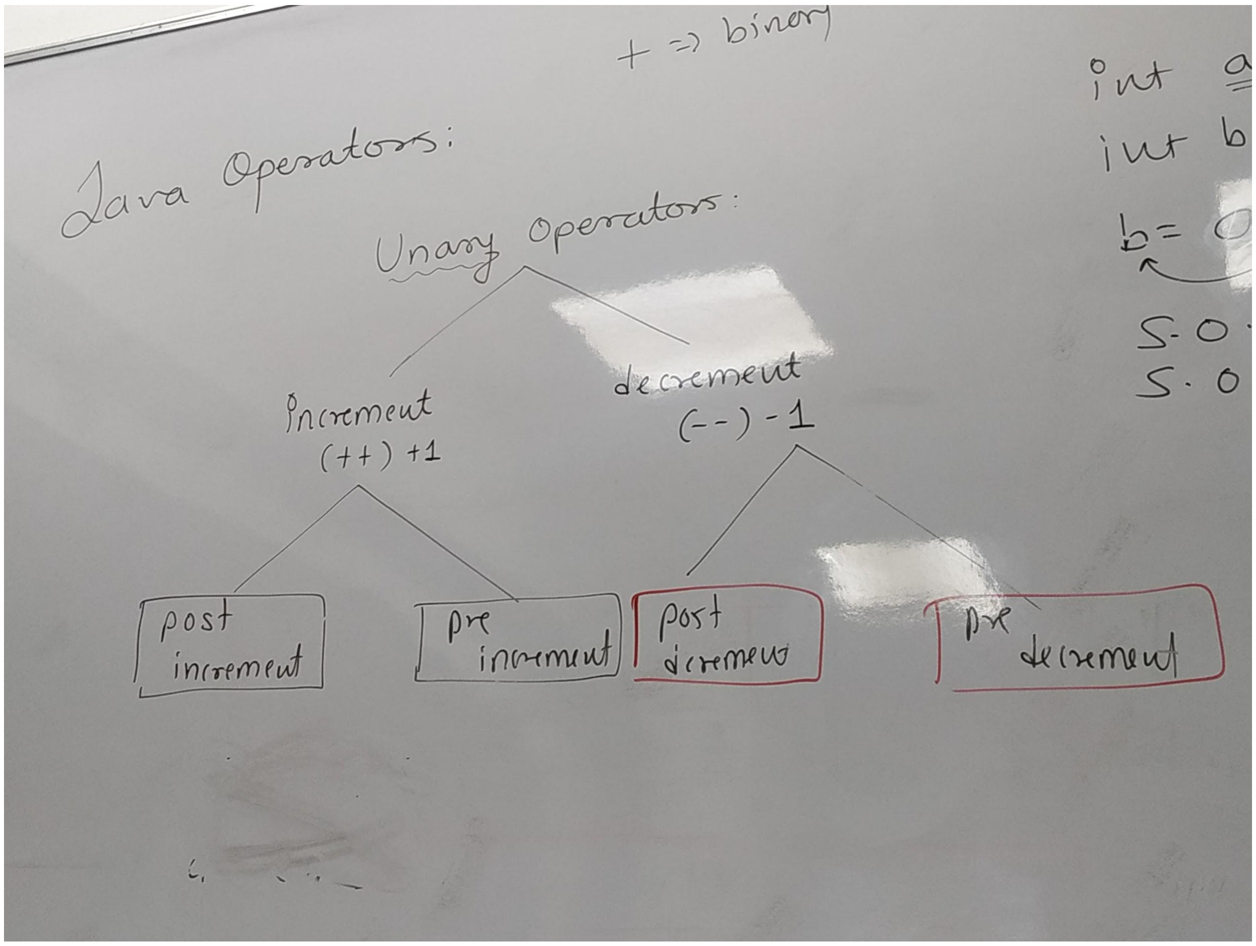
Operator Name	Usage and input	Example
=	Assign a value	A=10
+=	Addition assignment	a=a+b or a+=b
-=	Subtraction and assignment	a=a-b or a-=b
*=	Multiplication and assignment	a=a*b
/=	Division and assignment	a=a/b
%=	Modulo and assignment	a=a%b

- Relational operator: always returns Boolean values.

Operator Name	Usage & implementation	Example
==	Comparison	a==b
!=	Negative comparison	a!=b

<	Less than	a<b
>	Greater than	a>b
<=	Less than or equal to	a<=b
>=	Greater than or equal to	a>=b

- Unary operators: increment or decrement by 1. Unary operators are used to perform the operation on a single variable. Under operator classified in 2 categories
  1. Increment
  2. Decrement.



- Post increment: it means first use the current value inside the expression then increment the original value.
- Pre-increment: it means first increment the original value then use the latest value inside the expression.
- Post decrement: it means first use current value and decrease the original value after assignment.

- Pre-decrement: it means first decrease the original value and then use inside the expression.

Q. Logical operator

1. Logical operator acts as a conjunction between multiple conditions.
2. We can use these operators to perform logical operations on program data.
3. Following are the operators available in this category.

&&	Logical and
||	Logical or
!	Logical not

- Logical and:

1st	2nd	Output
T	T	T
T	F	F
F	F	F

- Logical or:

1st	2nd	Output
T	T	T
T	F	T
F	F	F

- Logical not: this operator always gives the negation of given value. If you are using no operator then output will always be Boolean value. We can use this operator along with other logical operators.

Q. Bitwise operator:

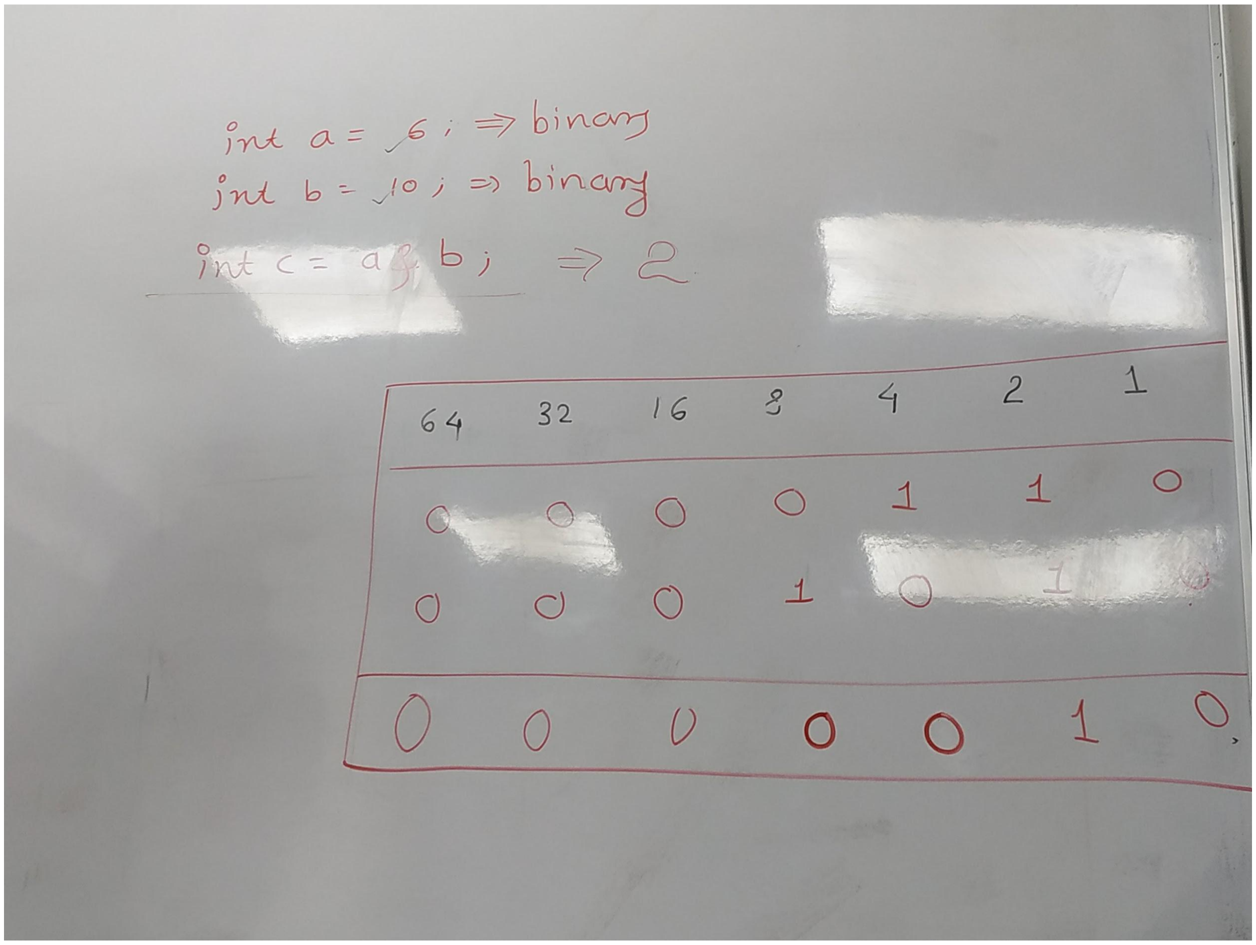
- They are used to perform operations on individual bits.
- They are applicable only for integer data types. For example: int short long byte.
- It always converts a given decimal value in the binary format before performing operations.
- It is classified in following types:

    1. Bitwise and: &
    2. Bitwise or: |
    3. Left shift: <<
    4. Right shift: >>

1. Bitwise and:

IP 1	IP 2	Op
T(1)	T(1)	1
T(1)	F(0)	0
F(0)	F/T (1/0)	0
F(0)	F(0)	0

- Example:



int a = 6; ⇒ binary
int b = 10; ⇒ binary
int c = a & b; ⇒ 2

64	32	16	8	4	2	1
0	0	0	0	1	1	0
0	0	0	1	0	1	0
0	0	0	0	0	1	0

2. Bitwise Or:

- Example:

3. Bitwise left shift: left shift can be used to shift the left by the number of times specified by the users.
    - In case of left shift operator decimal value will be converted into the binary format before performing shift operation.
    - We can use one formula to calculate the value of left shift expression.
    - Shifting the digit by one means multiplying by 2.
    - For example: int a = 10, int result= a<<3.
      1st shift : 10*2: 20
      2nd shift: 20*2: 40
      3rd shift: 40*2: 80

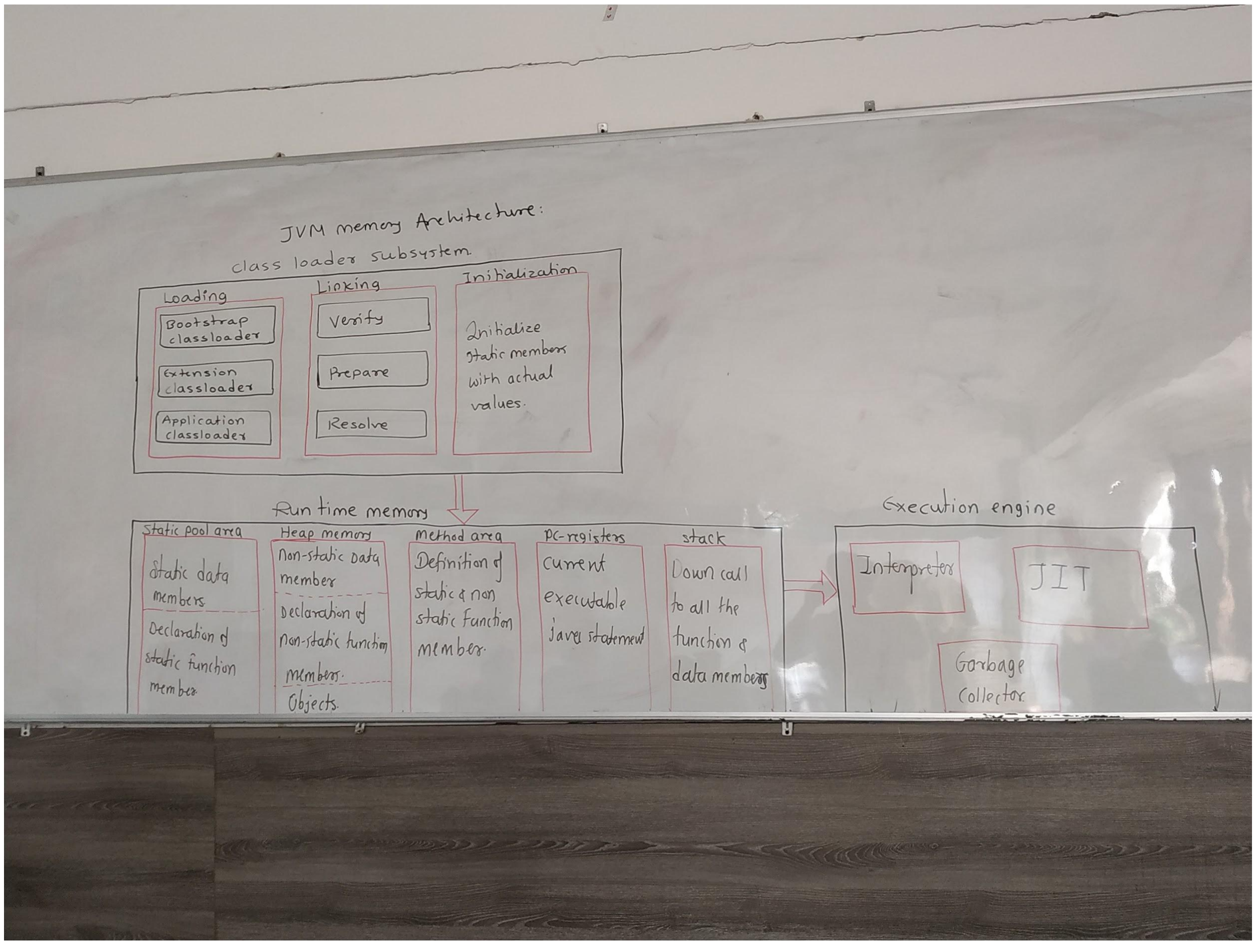
4. Bitwise Right shift operator to shift the given value to the right side by the number of times specified by the user.
   In the case of the right shift operator as well, the given decimal value will be converted into binary format before performing any operation.
   In order to Perform right shift operation we cannot exceed the limit of total number of positions. It means for right soft operation the end point is first position.
   If we are exceeding the limit then output will be 0.

Q. What is a ternary operator?

A. It can be used to apply conditions and expressions in a Java statement.
   It is an alternative to if-else statement.
   This operator accepts or operates on 3 parameters, hence it is known as ternary operator.
   Syntax: var_name=(exp1...n)? Exec 1:exe2;

Q. What is JVM memory architecture?

A.



1. JVM memory architecture is a collection of different components.
2. JVM memory architecture classified into 3 sections:
    a. Class loader subsystem
    b. Runtime memory
    c. Execution engine.

A. Class loader subsystem is responsible to perform following activities:
A. Loading
B. Linking
C. Initialisation.
    1. Loading: IN this phase Java classes will be loaded inside the JVM memory. In Java 3 types of class loader are present Bootstrap class loader. Extension class loader. Application class loader.

A. Bootstrap class loader is responsible to load inbuilt class inside the JVM memory for example scanner, string,etc
B. Extension class loader: this class h is responsible for the loading of external classes or 3rd party classes. For example database classes, file classes, server classes, etc.
C. Application class loader this class loader is responsible to load the classes created by developer (use defined classes)

2. Linking: in this phase JVM will perform three tasks in order to link the byte code:

1. Verify: in this phase JVM will verify the byte code generated by the compiler. If there is a problem in byte code then JVM will throw a run-time error.
2. Prepare: after verification default values will be assigned to all the class variables.
3. Resolve: in this phase JVM will resolve internal issues related with byte code.

3. Initialisation: after linking all the static variables will be initialised with actual values.

B. Runtime memory: it is classified into five sections:

1. Static pool area: this area acts as a container for all the static data members and declaration of static function members.
2. Heap memory: In this section all the non static data members and declaration of non-static function members. Objects of the class will be present inside the heap memory.
3. Method area: this area acts as a container for all the static and non-static destinations of the method.
4. PC registers: This section acts as a container for current executing Java statements.
5. Stack: is responsible to make a call for all the data members and function members.

C. In Java JVM itself acts as an interpreter, interpreter is responsible to understand the meaning of code. JIT just in time the compiler is responsible to convert byte code into Machine code, in order to improve performance of Java applications. Garbage collector: it deallocates the memory allocated for all the members of class after successful execution. It can be done implicitly as well as explicitly.

Q. Explain the process of class loading.

A. The process of loading class into the JVM memory is known as class loading. Class loading can be done implicitly and explicitly.
   If a Java file contains only the main class then JVM will automatically load the respective class inside the memory.
   In case a Java file is having more than one class then JVM is responsible for loading only the main class.
   Programmer is responsible for loading the business logic class inside the JVM memory.
   If a programmer is loading the business logic class then it is known as an explicit class loading.
   We can load the business logic class by accessing any one member of the respective class.

Q. What is a block?

A. Block is a collection of executable Java statements.
   Blocks can be used to perform the special operations at the time of software development.

Blocks are mainly classified into 2 types

1. Static
2. Non-static block

Q. Explain the use of static blocks.

A. The block which is declared along with a static keyboard is known as a static block.
 Static blocks can be used to develop a business logic which is common for all the users, and will be executed only once in the application cycle.
 If Java class is having static block and Main method then static block will execute before main method.
 We can provide multiple static blocks in a single Java class, in such a case all the static blocks will be executed in a sequential way.
 Static block executes before main method whereas main method will be executed after the class loading.

Q. What is a non-static block?

A. Block which is declared without any keyword is known as a non-static block.
B. Non-static block is used to develop a business logic which is common for all the objects and is executed after every object creation.
C. If we are creating multiple objects of the same class then multiple non-static blocks will be executed.
D. We can initialise non-static members inside the non-static block.

Q. What is the difference between a static and non-static block?

Static block	Non-static block
Block declared with static keyword is known as static block	Block which is declared without using static keyword is known as non-static block
Static block will be executed automatically at the time of class loading	It will be executed at the time of object creation
Static block can be used to initialise static members	Non-static block can be used to initialise non-static as well as static members
Static block will be executed once in a applicant lifecycle	Non-static blocks can be executed multiple times depending on the number of objects.

Q. Explain the difference between methods and blocks.

Methods	Block
Methods acts as container for Java statements	Block also acts as a container for executable Java statements
Methods are having identity	Blocks doesn't have identity
Methods can be executed explicitly. Methods have return type. It is possible to pass multiple arguments for a method.	Blocks are always executed by JVM. Blocks don't have a return type so values cannot be returned. We cannot pass runtime arguments for blocks.

Q. Explain the types of variables.

A. In Java variables are classified into 3 types:
    1. Class variable.
    2. Instance variable.
    3. Local variable.

- Variables declared inside the class outside the method using static keywords are known as class variables.
- Class variable can be accessible anywhere inside the same class as well as outside the class only through class name as a reference
- All the class variables will be having default values, for integer it will be 0, for Boolean it will be false, for object reference it will be null.

2. Instance variable:
- Variables declared inside the class j outside the method without using static keywords are known as instance variables.
- Instance variables can be accessible anywhere inside the same class as well as outside the class only through object reference.
- Instance variables will be having default values.

3. Local variable:
- Variables declared inside the method or block are known as local variables.

- Local variables can be static or non static depending upon the method type.
- We cannot access local variables outside the method or block.
- It's mandatory to initialise local variables because they will not be having default values.

Constructor:

Q. What is a constructor?

A. Construct is a special member of a class which can be used to initialise the state of an object.
   Class name and constructor name should be the same.
   It is not possible to provide return type for a constructor.
   Every Java class must and should have a constructor, either defined by compiler or else programmer.
   If the compiler defines the constructor then it is known as default constructor.
   Compiler can define only zero argument constructor.
   Compiler will provide a constructor only if the user defined constructor is not available.
   Programmer can define zero argument as well as argumented constructor.
   Constructor will be executed at the time of object creation.

Q. How can a constructor return the address of an object?

A. 1. Developers cannot return any explicit value from the constructor because the constructor doesn't have a return type.
   2. At the time of object creation a new operator is responsible to call the constructor of a specific class.
   3. When the constructor call takes place automatically the constructor will return the address of an object.
   4. It means it is not possible to return the address explicitly but by default constructor only return the address without support of return statement or return type.

Q. Why Java has provided 2 types of constructor?

1. Construct calls will happen at the time of object creation.
2. We can create objects for every Java class.
3. If the default constructor is not provided by Java then every time the developer has to provide an external constructor.
4. User defined Constructor is mainly used to pass the data at the time of object creation.
5. If the default constructor option is not there then we have to create an external constructor even if non static members are not available because of all these reasons two types of constructors are available.

Q. What is a constructor overloading?

1. The process of creating multiple constructors of a same class with different arguments is known as constructor overloading.
2. At the time of constructor overloading we have to pass different types of arguments for every constructor, agreements should be different either by augmented length or type.
3. Constructor overloading can be used to create different objects of the same class.

Q. What are the users of this keyword?

1. **This** keywords always point to the current object.
2. It means address of an object and **this** keyword will be the same.
3. **This** keyword is mainly used to differentiate local and instance variables.
4. We can use **this** keyword only inside the non-static context, for example: constructor or non-static method.

IDE.

Q. What is an IDE?

A. IDE stands for integrated development environment.
   It is a software application which enables users to more easily write and debug Java programmes.
   Most of the ide's features like:
    1. Syntax highlighting
    2. Code compilation

Which improves application performance and saves a lot of time.

IDE acts as a container for multiple components for example code editor, compiler, interpreter, debugger, etc.

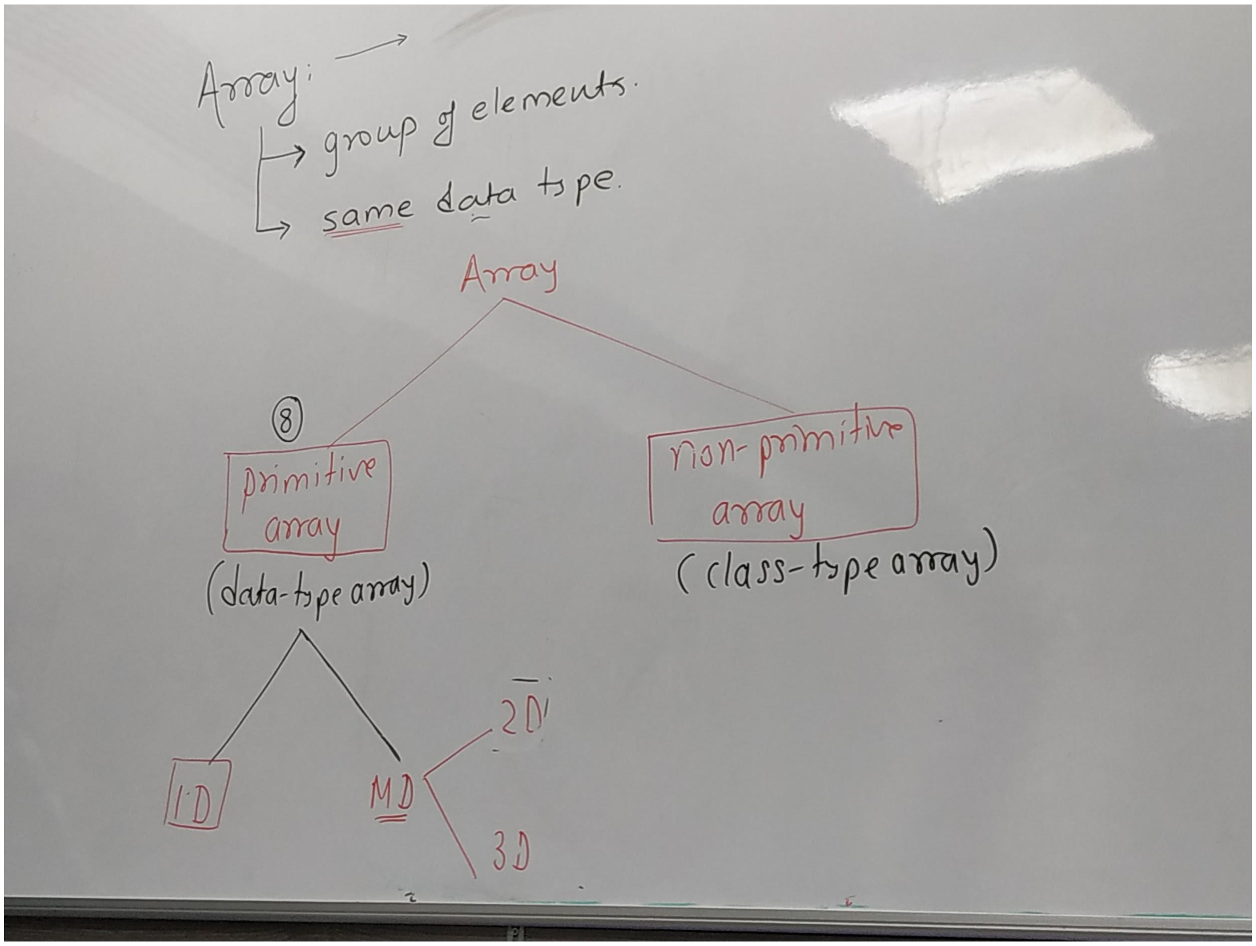
IDE also provides the support for testing components like maven, testNG, build, etc.

Examples of IDE:
1. NETBEANS
2. INTELLIJ IDEA
3. ECLIPSE
4. JCreator.

ARRAY:

Q. What is an array?

A. ARRAY is a collection of multiple elements.
B. ARRAY can be used to store a group of the same type of elements.
C. It means we can store only homogeneous values inside the array variable.
D. In Java array classified into 2 types:



Q. What is a primitive array?

A. An ARRAY variable created using primitive data type is known as primitive array.
B. Primitive array is also known as data type array.
C. Primitive array further classified into 2 types:
    1. One dimensional array.
    2. Two dimensional array.

Q. What is one dimensional array?

A. An ARRAY variable which stores data in terms of rows is known as 1D array.
B. 1D array is also known as a vector.
C. We can create one dimensional array by following 3 steps:
   1. Declaration: int[] arr1; or int arr1[];
   2. Size allocation: arr1 = new int[size];
   3. Initialisation:
      arr[0] = 10;
      .
      .
      .
      arr[4] = 50;

Q. What is a 3D array?

A. 3d arrays are known as multi dimensional arrays.
B. By using a 3D array we can store data in the format of rows, columns, and columns.
C. Following is the syntax of 3 dimensional array:
   int[][][] arr1;
   arr1 = new int [row][column][column of column];

String:

String class is an internal class declared inside the java.lang package.
String class can be used to declare string values.
Every string value will be considered as a String object.
We can create String variables by using 2 ways:

1. Without using a **new** operator.
2. With using **new** operators.

All String values will be stored in the string pool area situated inside heap memory.
String pool area classified into 2 sections:
   1. Constant Pool area
   2. Non-Constant Pool area.

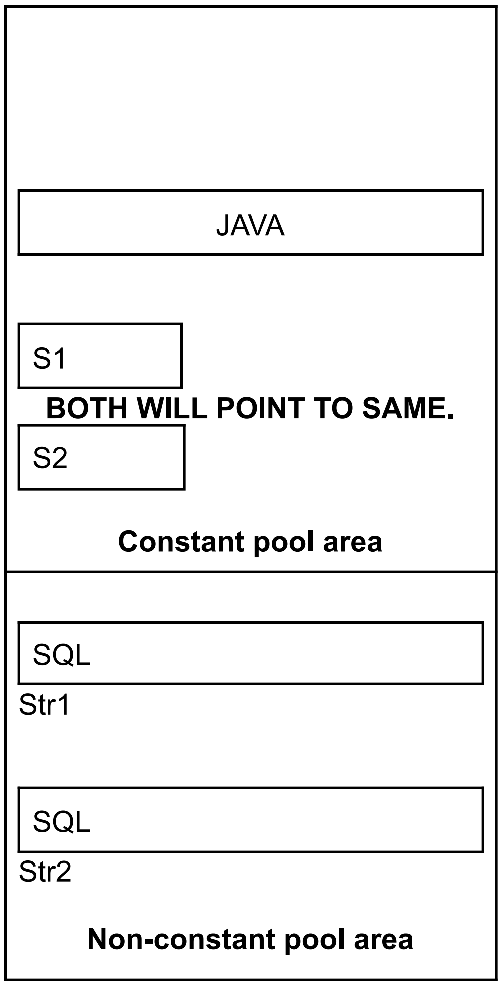
String Variables created without new operator will get stored inside the constant pool area.
String variable created using new operator will get stored inside the non-constant pool Area.

- Constant pool area does not allow duplicate values.
- Non-constant pool allows duplicate values.

String s1 = “JAVA”;
String s2 = “JAVA”;

String str1 = new String(“SQL”);
String str2 = new String(“SQL”);

String Pool area



Q. What is the difference between == operator and equals method?

A. == can be used to compare the address of an object whereas, equals method is used to compare content present inside the address.
B. We can use == for the comparison of primitive data types as well as non-primitive data types, it means we can compare the values of normal variables and addresses of reference variables.
C. Equals method is mainly used for the comparison of string values not for normal values.

Q. What is the use of equalsIgnoreCase method?

A. This method is also used for the comparison of 2 different string values.

B. equalsIgnoreCase method compares the content of 2 string values irrespective of case i.e uppercase and lowercase.

Q. Why String class immutable in nature?

1. Immutable means non-changeable.
2. Objects are classified into 2 categories i.e mutable and immutable.
3. If an object is immutable then modifications are not possible inside the existing copy, instead of that one separate copy will be generated for the modified string object.
4. String class is immutable in nature because after re-initialisation or modification new copy will be generated
5. After generating new copy old copy will be de-referred it means address variable will not be available for old copy
6. String class is immutable; it does not mean that we cannot reinitialize the value.

Q. Explain String Functions.

A. String functions can be used to perform various operations on String.
B. All the functions are **Case-sensitive**.
C. String class provides different inbuilt functions to operate string data.
    1. **length()**: Used to count the total number of characters(including space) in the string. It always returns integer values.
    2. **charAt(int index)**: This function can be used to find out the character present at a specified index. This function accepts index numbers as an input and returns character output .
    3. **indexOf(char ch)**: indexOf function can be used to find the index of a specified character. This Function always accepts character as an input and returns integer type value which represents index number. If the given character is having more than one occurrence then indexOf function will always return first occurence.
    4. **lastIndexOf(char ch)**: This function is also used to find out the index of a specified character. If the character is having more than one occurrence then this function will return last occurrence.
    5. **contains(String charseq)**: contains function is used to check whether the given character sequence is present inside the string value or not. If the character sequence is present then this function will return true else it will return false.
    6. **startsWith(String charseq)**: This function is used to check whether the given character sequence is present at start position or not. This function also returns boolean type of value, if sequence is present in the starting position the output will be true else it will return false.
    7. **endsWith(String charseq)**: This function is used to check whether the given character sequence is present in the ending position or not. If the sequence is present in the ending position then output will be true else output will be false.
    8. **subString(int startIndex)**: It can be used to extract a specified number of characters from the given string value. subString is also known as given String.

subString can be used with 2 types of parameters : 1. We can pass only the start index as a parameter for the subString function.
Syntax: we can pass start index as well as end index as parameter,
Ex. str.subString(startIndex, endIndex);

9. **toLowerCase().**
10. **toUpperCase().**
11. **toCharArray()**: It converts the given String input to a character array.
12. **split()**: It can be used to divide the given string value into multiple parts. This function accepts a string type of parameter as a condition and always returns an array of type String.
13. **isEmpty()**: This function is used to check whether the given string value is empty or not. If the string variable doesn't contain any data then this function will return output as true. If actual value is present then output will be false.
14. **trim()**: trim function can be used to remove the whitespaces from leading and trailing positions. This function will always return updated string value after removing whitespace.

Q. What is the use of while loop?

A. While loop can be used to perform multiple iterations, just like for loop.
B. While loop is executed until a certain condition is false.
C. It is also called an entry-control loop.
D. It is preferable to perform an unknown number of iterations.
E. Following is the syntax and working of while loop:

```
while(expression)
{
        Java statements;
        Java statements;
        Java statements;
        Java statements;
        .
        .
}
```

Q. What is the use of a do-while loop?

1. It is similar to while loop which can be used to perform multiple iterations.
2. It is also known as exit-control loop, it means first java statements will be executed then condition will be verified.
3. If we are using a do-while loop then at least one iteration will be performed every time.

```
do{
        Java statements;
        Java statements;
        Java statements;
        Java statements;
        ..
    }while(expression);
```

Q. What is the use of switch case?

1. It is one of the control statements, which can be used to apply restrictions to java statements.
2. We can use switch cases in order to create functions for the user.
3. Switch statements always accept choice as a parameter.
4. Inside the switch statement we can provide multiple cases.

   Syntax:

```
switch(choice)
{
        Case 1:
                Java statements;
                .
                .
                break;
        Case 2:
                Java statements;
                .
                .
                break;
        Case n:
                Java statements;
                .
                .
                break;
        Default:
                .
                break;
}
```

- Continue statement:
  - It can be used to skip the current iteration of for loop, while , do-while.
  - After the continue statement program moves to the end of the loop.
  - Syntax:

```
        for()
        {
                ----------------
                ----------------
                if(expression)
                {
                        Continue;
                }
        }
```

- Explain the difference between call by value and call by reference.
    1. The process of calling a method using primitive data type is known as call by value.
    2. It is applicable only for primitive data type reference(int,boolean,float,etc).

- Call by reference:
    1. The process of calling the method by providing an object reference is known as call by reference or reference variable.
    2. Call by reference is possible if the method accepts non-primitive parameters(Class-type parameters).
    3. If a method accepts a reference variable as a parameter then we have to pass object reference as a value. If object reference is not present then we have to pass null value.

Q. What are the object-oriented concepts?

1. Java is an Object-Oriented programming language.
2. By using OO Concepts we can develop different types of java applications.
3. Following are the pillars of Object-Oriented Programming.
    - Inheritance.
    - Polymorphism.
    - Encapsulation
    - abstraction.

Q. What is meant by object composition?

1. In java objects can have 2 types of relationships
    - Has-A(Object Composition)
    - Is-A(Inheritance)
2. If one object holds the properties of another object then it will be considered as object composition.
3. It is possible to composite multiple objects inside the single object.
4. Reference variables of an object will be considered as members of the class which can be static or non-static.

Q. What is an inheritance?

1. If one class acquires the properties of another class, then it is known as an inheritance or **Is-A** relationship.
2. The class which acquires the properties of another class is known as subclass.
3. The class from which properties are going to inherit is known as super class.
4. Inheritance is possible by using **extended** keywords.
5. Sub-class can inherit non-static members of superclass.

Q. Explain the types of inheritance.

1. Inheritance classified into 4 types.
   1. **Single Level Inheritance:** If one sub-class extends the properties of only one superclass then it will be considered as single level inheritance.
   2. **Multi Level Inheritance:** If one subclass extends the properties of one superclass and that super class acquires the properties of its superclass this process will be considered multi-level Inheritance.
   3. **Multiple Inheritance**: If one subclass is trying to extend properties of more than one superclass then it is known as multiple inheritance.
   4. **Hierarchical Inheritance:** If one superclass is having more than one sub-classes is known as hierarchical inheritance. In other words hierarchical inheritance means multiple subclasses can extend the properties of only one superclass.

Q. What is Constructor Calling?

1. The process of calling super class constructor using sub class constructor is known as constructor calling.
2. If 2 classes are having **IS-A** relationship then constructor call is mandatory.
3. constructor calls can be done by using **super()** statements.
4. super() statement should be the first statement of constructor calling.it means multiple super() statements are not allowed, because java does not support multiple inheritance.
5. Constructor call can be done implicitly as well as explicitly.

- **Implicit constructor call:**
  1. If Compiler calls the constructor by providing a super statement inside the sub-class statement in body then it is known as implicit constructor call.
  2. Compiler can call only zero parameterised constructor of superclass.
- **Explicit constructor call:**
  1. If a programmer calls the constructor of superclass by providing a super statement then it is known as an explicit constructor call.
  2. If the superclass constructor is a parameterised constructor and constructor call must be done explicitly.

Q. Why does java not support multiple Inheritance?

1. Multiple Inheritance means one subclass is acquiring more than one superclass.
2. If one class is acquiring properties of more than one superclass then we have to provide multiple super statements inside the sub class constructor body which is not possible because super statements must be the first statement of the constructor.
3. Because of all these reasons, java does not support multiple inheritance.
4. If a subclass is trying to access properties of more than one superclass then it creates diamond ambiguity.

Q. What is a Constructor Chain?

1. Subclass constructor calls the constructor of superclass that super class constructor calls the constructor of its super class constructor, this process is known as constructor chaining.

2. It is possible only through inheritance .
3. Every class in java will be having super class either provided by developer or else compiler.
4. If the developer is not providing a super class then by default the compiler provides the object as a superclass.
5. It means every class in java will be having properties of an object class.

Q. What is the use of this statement?
1. This statement can be used to call another constructor of the same class.
2. We can use this statement in case of constructor overloading.
3. This statement should be the first statement of the constructor body.
4. Multiple statements are not allowed inside the constructor.

Q. What is the use of super keywords?
1. **super** keyword can be used to access the non-static members of superclass.
2. If the superclass and subclass have the same identifier then we can use super keyword to differentiate super class and subclass identifiers.
3. We can access multiple members by using super keywords.

Q. Differentiate between super(),super,this(),this.

super()	**super keyword**	**this()**	**this keyword**
Super statement can be used to call the constructor of super class.	Super keyword can be used to call non-static members of super class(**v&m()**).	This statement is used to call another constructor of same class	This keyword can be used to differentiate between instance and local variable.
Super statement must be declared inside the constructor body	Super keywords can be declared anywhere inside the method.	This statement must be declared inside the constructor.	This keyword can be declared inside the non-static method or constructor
Super statement must be the first statement of constructor	It's not mandatory to provide the super keyword at first line	This statement should be the first statement of constructor body	It is not mandatory to provide this keyword at first line
Multiple super statements are not allowed inside the constructor	We can use super keyword multiple times	Multiple this statements are not allowed inside the constructor	We can use this keyword multiple times

Q. What is method overloading?

1. The process of creating multiple methods with the same name but different arguments is known as method overloading.
2. At the time of method overloading, parameters must be different either by parameter length or parameter type.
3. Method Overloading can be used to achieve the same functionality, with different parameters.
4. We can overload static as well as non-static methods.
5. It is possible to achieve compile-time polymorphism, by using method overloading.
6. Subclass can overload the method of superclass if two classes are having **is-A relationship**.

Q. Is it possible to overload the main method?

1. Yes, it is possible.
2. If the class is having multiple main methods in such a case, JVM will execute the standard Main Method.
3. The method which accepts the String[] type of arguments will be considered as the Standard main method, remaining methods will be treated as external methods.
4. We have to call external main methods inside the standard main method for execution.

Q. What is method overriding?

1. The process of inheriting a method from subclass and changing its implementation in subclass is known as method overriding.
2. In case of method overriding method signature should be the same, but method implementation can be different.
3. It is used to achieve different functionalities with the same parameters.
4. It is possible only through **IS-A** relationship.
5. By using method overriding we can achieve run-time polymorphism.

Q. Which methods cannot be overridden?

1. We cannot override 3 types of methods:
    - Static: they are the class members and they will be having a single copy inside the JVM Memory hence it is not possible to override static methods.
    - Private: They are not accessible outside the class so it is not possible to override private methods.
    - Final: If the method is final, then it is not possible to change the definition of the respective method, hence we cannot override the final method.

Q. What is the use of the final keyword?

1. Final keyword can be used to declare constant values.
2. We can use the final keyword at 3 different locations.1. Final variable 2. Final Method 3. Final Class.
3. If the variable is final then re-initialisation is not possible.
4. If the method is final then overriding is not possible.
5. If class is final then inheritance is not possible.

Q. Difference between overloading and overriding.

overloading	**overriding**
It means creating multiple methods with the same name but different signatures.	Overriding means creating multiple methods with different functionality
Used to achieve the same functionality with different parameters.	Used to achieve different functionality with same parameters
It can be done inside the same as well as different classes	It must be done in a different class
Overloading can be used to achieve compile-time polymorphism	Overriding can be used to achieve run-time polymorphism.
We can overload static and non-static methods	We can only override non-static methods.
It is possible without inheritance	It must be done with inheritance.

Q. What is Casting?

1. The process of converting one type of information into another type is known as casting.
2. In java casting is mainly classified into 2 types:
    1. Primitive casting
    2. Non-Primitive casting.

CASTING

PRIMITIVE(DATA-TYPE**)** **NON-PRIMITIVE(**CLASS-TYPE**)**

NARROWING **UPCASTING**
WIDENING **DOWNCASTING**

Q. What is primitive casting?

- It is also known as data-type casting.
- The process of converting one primitive information into another primitive information is known as data type casting or primitive casting.
- Primitive casting is mainly classified into 2 types:
    1. Narrowing
    2. Widening

1. Narrowing: The process of casting higer types of information into lower types of information is known as narrowing. It must be done **explicitly**.

2. Widening: The process of casting lower types of information into higher types of information is known as widening. It can be done **implicitly** as well as **explicitly.**

Q. What is Class type-casting?

1. The process of casting one class type information into another class is known as class type casting.
2. To perform class type casting we have to follow some rules:
    a. 2 classes should have **IS-A** relationships.
    b. The object which has to be casted into another class should contain the properties of the target class.
    c. Class type casting is classified into 2 types:
        1. Up-casting
        2. Down-casting

Q. What is Up-casting?

1. The process of casting subclass type information into its superclass is known as an Up-casting.
2. It can be done implicitly as well as explicitly.
3. If the compiler performs up-casting then it is known as implicit up-casting.
4. If a programmer performs up-casting the unit will be considered explicit up-casting.
5. After up-casting we can access the properties of only super class.

Q. What is **ClassCastException**?

1. It is a type of run-time Exception or unchecked exception.
2. JVM is responsible for throwing **ClassCastException.**

Q. When does it occur?

1. JVM will throw a ClassCastException if the developer is trying to convert one class type object into another class type and the respective object does not contain the properties of the target class.

Q. Wny Compiler not throw a ClassCastException?

1. Because the compiler is responsible for checking syntax and rules of the language.
2. Compiler will not throw a ClassCastException because syntactically it is correct.

Q. How to avoid **ClassCastException**?

1. We can avoid **ClassCastException** by using **instanceof** operator.

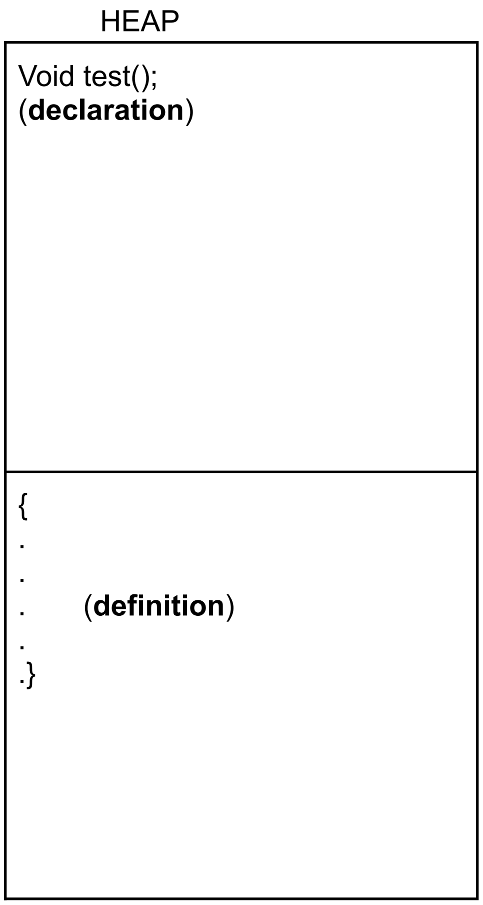
NOTE: METHOD OVERRIDING IS THE EXCEPTIONAL CASE FOR THE RULE OF UPCASTING. IT means, if a method is overridden, the latest implementation will always be executed.

Q. What is Downcasting?

1. The process of casting superclass-type information into its subclass is known as downcasting.
2. Downcasting must be done explicitly.
3. To perform downcasting, first we have to upcast an object, it means without upcasting downcasting is not possible.
4. After downcasting we can access the properties of subclass as well as superclass

Q. What is Binding?

1. Binding is the process of attaching the declaration with its definition.
2. It can be done at the time of compilation or execution.



Q. What is Polymorphism?

1. An object showing different behavior in different stages is known as polymorphism.
2. Polymorphism has 2 types:
   1. Compile Time Polymorphism:
      - If the compiler binds the method declaration along with its definition at the time of compilation is known as compile-time polymorphism.
      - Compile Time Polymorphism is also known as early binding or static binding.
      - Static binding cannot be re-binded.
      - Method Overloading can be used to achieve compile time Polymorphism.

2. Run-time Polymorphism:

- If **JVM** binds the method declaration and definition at the time of execution then it is known as run-time polymorphism.
- Run-time Polymorphism is also known as late binding or dynamic binding.
- Dynamic binding can be rebonded.
- To achieve Run-time Polymorphism we have to use following concepts:
    1. Method Overriding.
    2. Inheritance
    3. up-casting.
- By using Run-time Polymorphism we can achieve generalization.

Q. What is Generalization?

1. It is the process of extracting common properties of all classes and combining into one class.
2. Generalization can be achieved through run-time polymorphism.

Q. What is an abstract class?

1. The method which is having declaration as well as definition is known as concrete method.
2. The class which contains concrete methods is known as concrete class.
3. If a method is having only a declaration then it will be considered as an abstract method.
4. If a method is an abstract then it must be declared inside the abstract class.
5. Abstract class can act as a container for abstract as well as concrete methods.
6. Abstract methods and classes must be declared by using abstract keywords.
7. It is not possible to create an object of abstract class, because a new operator does not work with an abstract keyword.
8. A sub-class can extend the properties of super class which is abstract in nature, in such a case, sub-class is responsible to provide an implementation for all the abstract methods of super class. If not then sub-class will also become an abstract class.
9. Abstract class can be used to achieve **partial abstraction.**

Q. What is an Interface?

1. Interface is a java type definition block which can be used to declare abstract methods.
2. We can not provide concrete methods inside the interface because it is completely abstract in nature.
3. It is possible to declare static concrete methods from **JDK1.8.**
4. It is possible to declare data members inside the class and by-default all the data members will be having **public static and final access**.
5. A class can implement properties of interface.
6. In such a case class is responsible to provide implementation for all the abstract methods of interface. If not then class will become an **abstract class**.

7. It is not possible to create an object of interface, because it contains only abstract methods.
8. One interface can extend the properties of another interface.
9. We can create a reference variable of type interface in such a case, we have to provide an object of implementation class.
10. One class can implement the properties of more than one interfaces.

Q. Why is it not possible to create an object of INterface?
1. Interface is a java type definition block which does not extend the properties of Object Class.
2. We can not create an object of Interface, because it contains **no-static abstract Methods.**
3. At the time of object creation we have to use a new operator which does not work with abstract keywords.
4. Since Interface is a definition block, which does not contain constructor and object creation is not possible without constructor.
5. Because of all these reasons we cannot create objects on the Interface.

Q. How to achieve multiple Inheritance by using Interface?
1. The class can implement properties of more than one interface because the interface does not extend the properties of the object class.
2. To achieve multiple inheritance we have to create multiple interfaces and all interfaces should be implemented by using java class.
3. We can extend a class and implement an interface by using a single sub-class.

Q. What is the use of default methods?
1. Interface is a collection of non-static abstract methods and static concrete methods.
2. We cannot provide implementation for non-static methods, inside the interface because it is purely abstract in nature.
3. We cannot inherit static concrete methods of super class into the subclass because static members are the class members.
4. From **JDK1.8** we can create a default concrete method inside the interface which can be inherited in subclass.
5. Default methods are the special methods which can be inherited in a subclass and can be accessed by using object reference

Q. What is meant by access modifier in Java?
- Access Modifiers can be used to set the visibility for the members of the class.
- In java access modifiers are mainly classified into 4 types:
    1. **Private:** If members are having private access then we cannot access those members outside the class.
    2. **Package-Level:** If members are having package level access then we can access those members anywhere inside the same package.

3. **Protected:** protected members can be accessible anywhere inside the same package as well as outside the package only through Inheritance.
4. **Public:** we can access public members anywhere inside the project as well as outside the project.

Q. What is Encapsulation?

1. The process of binding members of the class along with a class body and protecting them by using access modifiers is known as an encapsulation.
2. In other words encapsulation means the process of wrapping data members and function members as a single unit.
3. Encapsulation can be used to achieve **data hiding**.

Q. Rules of Encapsulation.

Section-1:

1. We Can Declare One class inside the other class, then it will be considered as an inner class.
2. Inner class can have anyone of 4 access modifiers whereas outer class can have either public or package level access.
3. By default all the members of class will be having package level access whereas interface members will be having public level access.

Section-2:

1. One java file can act as a container for multiple classes. In such a case only one class should have public level access. File name should be same as class name which is having public access
2. If all classes are having package level access then we can provide any class name as a filename.

Section-3:

1. A constructor can have anyone of 4 access modifiers.
2. Default constructor will be having the same access as that of a class.
3. If the constructor is private then object creation is not possible.

Q. What is a Java Bean Class?

1. The class which contains private data members and public getter, setter.
2. Java Bean class can be used to achieve encapsulation along with data hiding.
3. Getters method in the class provides the read access for private data members it means we can access private members outside the class through getter method.
4. Setter method provides write access for private data members, it means we can modify private data members outside the class by using the setters method.

Q. What is an inner class?

1. One class declared inside the other class known as **inner class or nested-class.**
2. Inner class can be used to encapsulate properties of one object inside the other object.

3. It is possible to create multiple inner classes inside the single outer class.
4. Inner class can be static or non-static because the inner class will be treated as a normal member of the class.
5. To access properties of the inner class we have to create an object of the outer class.
6. Syntax:
    a. class Outer
    ```
    {
            static class inner
            {
                    -----;
            }
    }
    ```
    b. class outer
    ```
    {
            class inner
            {
                    ----;
            }
    }
    ```

Q. What is java anonymous class?

1. In java we can declare one class inside the other class, then it will be considered an Inner class.
2. The inner class declared without any name is known as anonymous class.
3. We can create anonymous classes that provide implementation to all abstract methods of an interface at the time of object creation.
4. After creating an anonymous class we don't have to provide implementation for abstract methods of an interface by using sub-classes.

Q. What is a Singleton Design Pattern?

1. The process of creating only one object throughout the application of life cycle is known as singleton design pattern.
2. The class which contains a single tone object is known as a singleton class.
3. To create a singleton object we have to declare a private constructor and one static method for object creation.
4. If an object is created then we have to provide the address of that object to the new reference variable and if the object is not present then the new object will be created.

Q. What is an Abstraction?

1. Abstraction is the process of hiding an implementation of an object without exposing it to the utilisation layer.
2. In other words abstraction means hiding object functionality and providing an interface or media to access that functionality.

3. Abstraction can be achieved by using 3 steps:
    a. Generalise all the properties of an object into an interface.
    b. Provide implementation for object functionalities by using classes.
    c. Create a reference variable of type interface to access object functionalities.
4. Abstraction can be used to achieve **loose-coupling.**

Q. What is coupling?

1. Coupling represents dependency between 2 classes.
2. In java there are 2 types of coupling.
    a. Loose Coupling
    b. Tight Coupling
3. If one class is independent of another class then it is known as **loose-coupling**.
4. If one class is completely dependent on another class then it is known as **tight coupling.**

Example:

1. super class→ Car**(TIGHT COUPLING)**
   Sub class→ vehicle.
2. Super class→ Vehicle**(LOOSE COUPLING)**
   Subclass→ Bike,Car,etc

Q. What is Factory-Design Pattern?

1. The process of creating objects of implementation classes without exposing it to utilisation layer is known as factory design pattern.
2. The method which contains objects of implementation classes is known as factory method.
3. The class which contains the factory method is known as factory class.
4. Abstraction is the backbone of factory-design patterns.
5. By using factory design pattern we can create **Loosely Coupled** Objects

Q. What is the difference between Abstract class and Interface?

Abstract Class	Interface
It can be used to store abstract methods as well as concrete methods	It is a special type of java definition block which acts as a container for only abstract methods.
We can create non-static concrete method inside the abstract class	Interface does not allow to store non-static concrete methods.
We cannot create objects of abstract class because the new operator does not work with abstract keywords.	We cannot create object of interface because it will not be having constructor

By default members of abstract class are having package level access	By default members of interface are having public level access
We can declare static as well as non-static variables inside the abstract class	All interface variables are **public static** and **final**
Abstract class can be used to achieve partial abstraction	Interface can be used to achieve complete abstraction
Sub-class has to extend the properties of abstract class.	Sub-class has to implements the properties of an Interface

Q. Difference between Abstraction and Encapsulation?

Abstraction	Encapsulation
It is the process of hiding implementation of an object from it’s utilisation layer	It means binding data members and function members as a single unit
Abstraction can be used to solve the problem at design level	Encapsulation can be used to solve the problem at implementation level
Abstraction is all about hiding unwanted details and giving essential details	Encapsulation means hiding the code and data into single unit
Abstraction can be used to achieve loose coupling between classes	It cannot provide surety that loose coupling will happen in encapsulation.
Abstraction mainly focuses on what an object should behave instead of how it behaves	Encapsulation means hiding the internal details of an object by providing single-unit to the end user.

Q. What is a Non-primitive array?

1. Non-primitive array is also known as an object array or class-type array.
2. Non-primitive arrays can be used to store **multiple objects of the same class.**
3. After creating a non-primitive array we can easily access object details of specific classes.

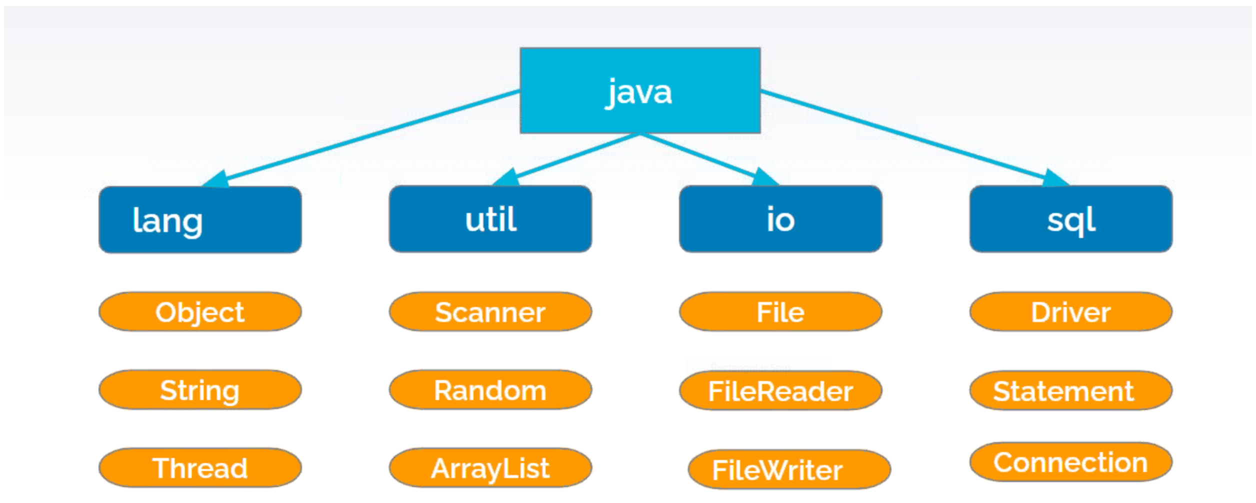
Example:

Enter how many Customers
Enter Name: **Enter Account Opening Balance:**
1. **Get all Customers.** 2. **Search Customer by Acc.no** 3. **Deposit**→ Accept AccNo 4. **WithDraw**→ Accept AccNo 5. **Check Balance**→ Accept AccNo 6. **EXIT**

Bank Array:
AccNO,Name,Balance.

Section-3

Java Libraries:



1. Java Library is a collection of Multiple Classes and interfaces.
2. It is also known as API, because it provides the platform for application development.
3. Java Language has provided various packages which can be used to develop different types of java applications.
4. For Ex: DataBase Application, File Applications, Collection Applications, etc.

5. Language package is the default package for every java application. It means every java program will be having properties of the language package.
6. To use the features of other packages, we have to import them explicitly.

Object Class:

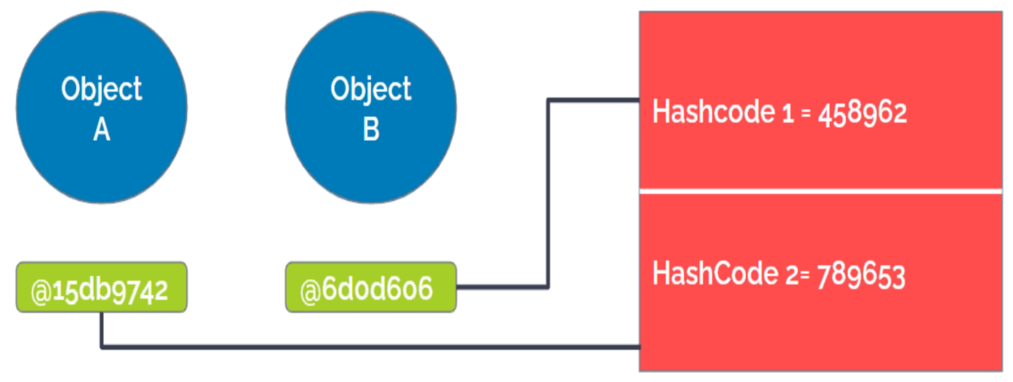
1. It is declared inside the java.language package.
2. Every java class will be having properties of an object class.
3. A developer can inherit properties of an object class implicitly as well as explicitly.
4. Object class has provided few important methods which can be used to get the different types of information.
    a. toString()
    b. hashCode()
    c. clone()
    d. equals(String args)
    e. getClass()
    f. finalize()
    g. wait()
    h. notify()
    i. notifyAll()

Q. Explain the use of toString method.

1. toString method is declared inside the object class.
2. This method can be used to represent an object as a string value.
3. For the string representation of an object we have to override the toString method inside the respective class.
4. toString method always returns String type of value.
5. Whenever we are printing a reference variable internally the toString method will be executed and by default it will return the address of an object.

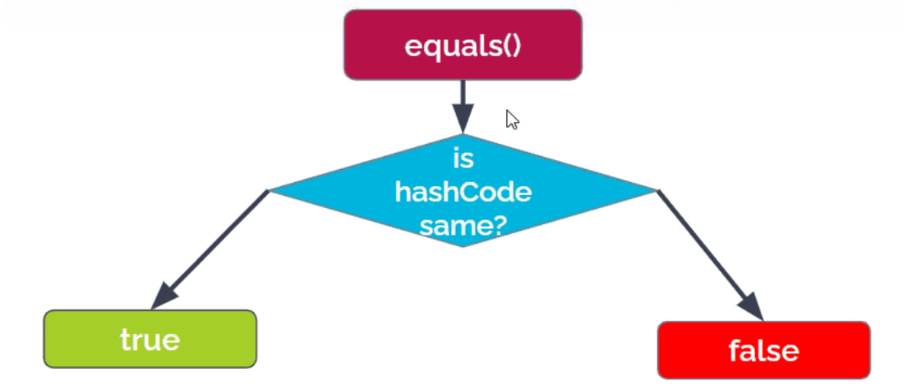
Q. What is the hashCode method?

1. Hashcode method is declared inside the object class.
2. Hashcode method can be used to identify a specific object.
3. Every object will be having a specific hashcode value which is present in integer format.
4. Hashcode value will be generated by JVM immediately after object creation.



Q. What is equals method?

1. Equals method is also declared inside the object class.
2. This method can be used for the comparison of 2 objects.
3. At the time of comparison the hashcode of an object plays an important role.
4. If 2 objects are having the same hashcode then output will be true else output will be false.

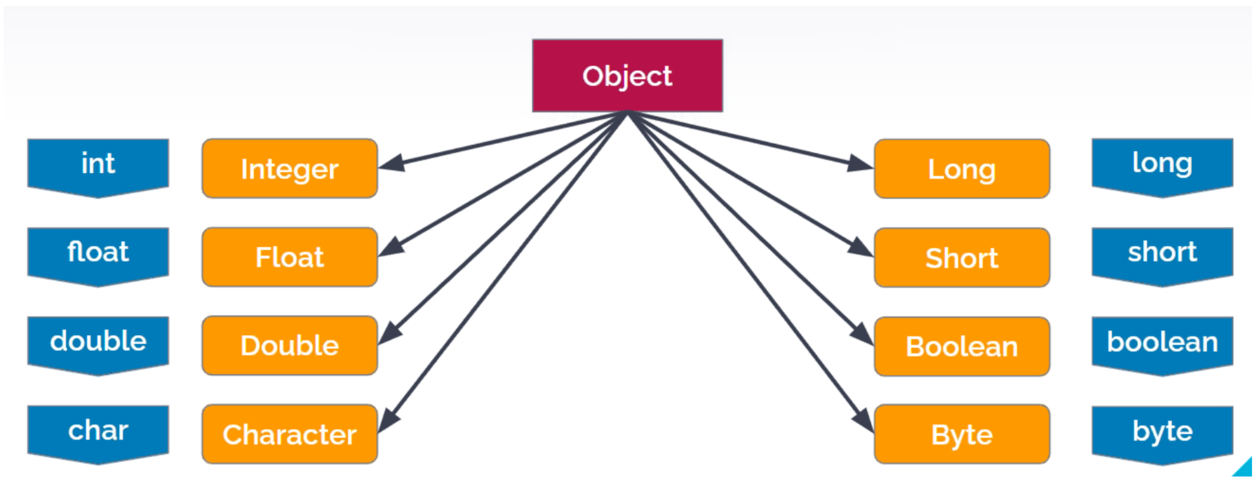


Q. What is the getClass method?

1. It is declared inside the object class.
2. This method always returns a **full-qualified class_name** of an object.
3. We can call this method by using the reference of a specific object.

Wrapper Class

1. Every primitive data type will have a wrapper class.
2. All wrapper classes extend the properties of object class.
3. Wrapper classes are declared inside the java.language package.
4. Wrapper classes are also known as non-primitive data types which can be used to represent information in terms of objects.

5. 

Autoboxing:

1. The process of converting primitive information into non-primitive(Object Form) is known as autoboxing.
2. Autoboxing is applicable only for wrapper classes.
3. After autoboxing we can use the features of object class.
4. At the time of autoboxing we have to create the reference variable of a specific wrapper class.
    a. Primitive Info:
        i. int a = 20;
    b. Non-primitive Info:
        i. Integer i = new Integer(20);
    c. Autoboxing:
        i. int a =50;
        ii. Integer i = a OR Integer i = new Integer(a);

Auto-Unboxing:

1. The process of converting non-primitive information to primitive is known as auto unboxing.
2. Unboxing can be done only on wrapper class.
3. After unboxing we cannot use the features of object class.
4. Example:
    a. Integer i = new Integer(50);
    b. int a=i;

Parsing Techniques:

1. Is used for string value conversion.
2. We can parse the string values by using wrapper class.

3. It is possible to convert string value into the specific primitive and non-primitive data type.
4. Syntax:
    a. String to primitive or non-primitive:
        i. Wrapper_class.parse()
5. Similarly we can convert primitive data type value to the string value by using **toString()** method.
6. Following are the methods that can be used to convert string values to the primitive data type.
    a. Integer.parseInt(String s)
    b. Double.parseDouble(String s)
    c. Float.parseFloat(String s).
    d. Long.parseLong(String s).
    e. Short.parseShort(String s)
    f. Boolean.parseBoolean(String s)
    g. Byte.parseByte(String s)

7. To convert primitive data values in string values we can use following methods:
    a. Integer.toString(int a).
    b. Double.toString(double b).
    c. Float.toString(float c).
    d. Long.toString(long a).
    e. Short.toString(short a).
    f. Boolean.toString(boolean b).
    g. Byte.toString(byte b).
    h. Character.toString(char c).

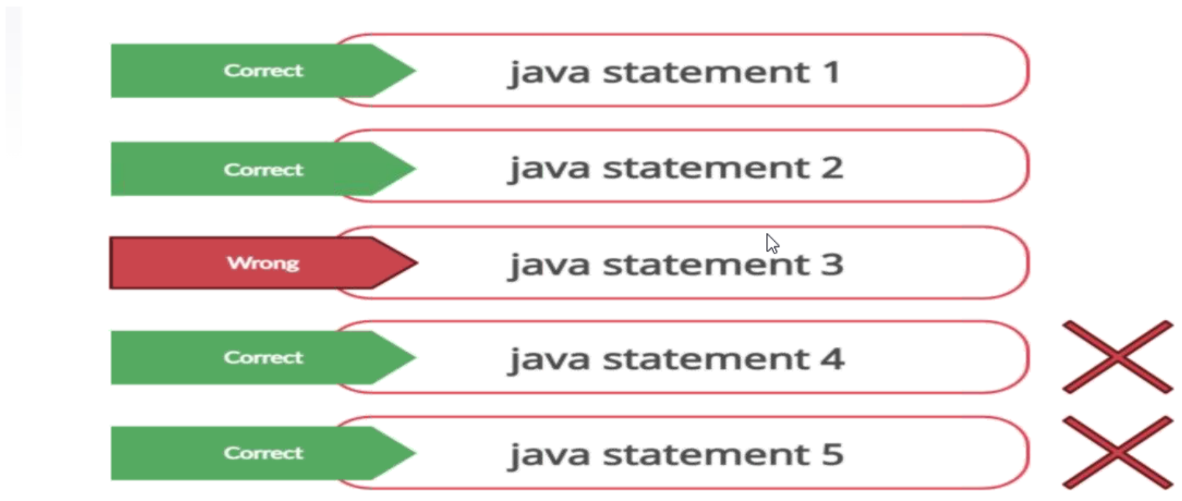
Difference between Casting AND Auto-Boxing/Unboxing AND Parsing

Casting	Auto Boxing/Unboxing	Parsing
The process of converting one type of information into another type is known as casting	The process of converting primitive information into non-primitive format and vice-versa is known as Auto Boxing/Unboxing	The process of converting string values into primitive data type and primitive value into string format is known as parsing.
Casting can be done on primitive data types or non-primitive data types	boxing /unboxing is possible only if primitive and non-primitive data types are involved	Parsing can be done only if string values are involved
Casting must be done through the same types of data type categories. For Ex: primitive to primitive or non-primitive to non-primitive	boxing/unboxing can be done only through different data type categories. For ex: primitive to non-primitive or vice-versa.	Parsing can be done only if string value and primitive data types are present. For Ex: String to primitive or primitive to string.
After casting input values might get changed.	After Boxing/Unboxing input values won’t be changed.	After parsing input value remains constant but there type will be different
Casting is possible without wrapper classes.	Auto Boxing/Unboxing is not possible without wrapper class	Parsing is not possible without wrapper class and string values.

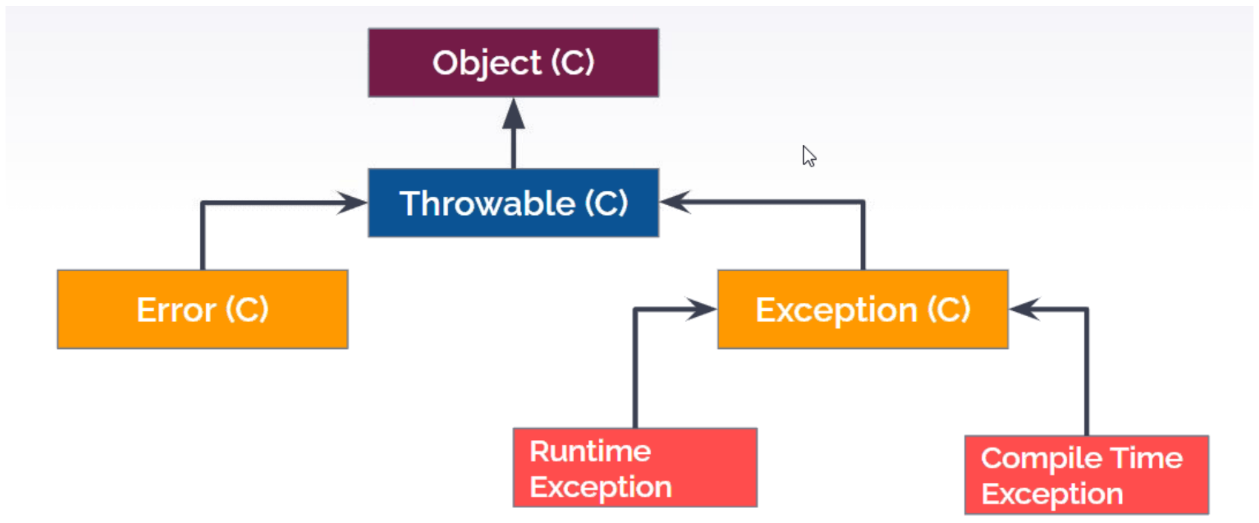
Exception Handling.

Q. Why do we have to use exception handling?

1. Exception handling can be used to handle abnormal situations present inside the java program.
2. If we are not using exception handling then java application will be terminated abnormally.



Q. Explain the Exception Hierarchy in java programming language.



A.

1. Exception hierarchy is a collection of different classes.
2. Throwable is the root class, present inside the exception hierarchy which extends the properties of an object class.
3. Throwable class is having 2 important sub-classes:
    a. Error
    b. Exception
4. Exception further classified into 2 types:
    a. Run-time Exception(Unchecked Exception).
    b. Compile-time Exception(Checked Exception).

Q. What is an Error?

1. Error indicates a serious problem due to lack of system resources.
2. Error is a subclass of throwable class.

3. Examples:
    a. StackOverflow error.
    b. OutOfMemory error.

Q. What is an Exception?
1. It is an abnormal situation or unwanted event which occurs during the execution of program and disrupts normal flow of the program instruction
2. df

Q. explain the difference between error and exception.

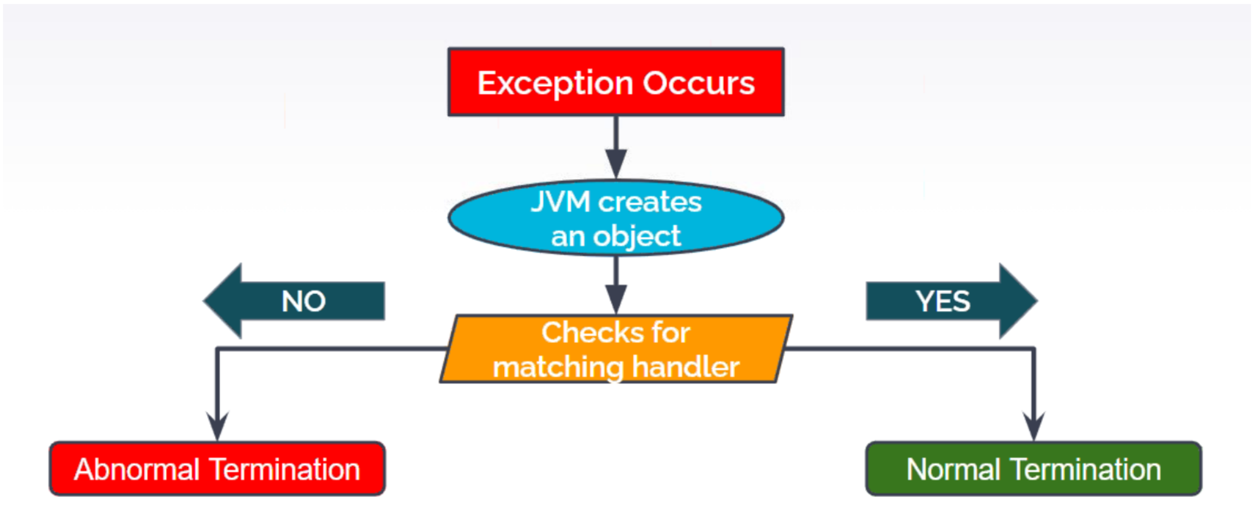
Error	Exception
Errors are not recoverable.	Exceptions are recoverable.
Errors are mostly caused by the environment in which the program is running.	Program itself responsible for causing exceptions.
Errors occur at runtime. They will not be known to the compiler.	Compile-time exceptions are known to the compiler. But runtime exceptions will not be known to the compiler.
Stackoverflow error, out of memory error,etc	SqlException, IOException, etc.

Q. Difference between checked and unchecked Exception.

Un-checked Exception	Checked Exception
They are also known as run-time exceptions.	Checked exceptions are also known as compile-time exceptions.
JVM is responsible to throw unchecked exceptions.	Compiler is responsible for throwing checked exceptions.
External resources are not involved in unchecked exceptions.	External resources are involved in checked exceptions.
It’s not mandatory to handle unchecked exceptions.	It is mandatory to handle checked exceptions.
Unchecked exceptions are having	Checked exceptions will be having high

low-priority because only java resources are involved.	priority because external resources are involved.
Examples: Arithmetic exception, array indexOutOfBounds exception, ClassCastException.	Examples: IOException, Interrupted Exception, SQLException.

Q. Execution Flow of Exception Handling:



1. Whenever an exception occurs in a java program JVM will create an object of respective exception class.
2. Try block is responsible to throw an exception object created by JVM towards the catch block.
3. Inside the catch block we have to provide a matching handler to handle exception objects thrown by try blocks.
4. If a matching handler is present then the exception object will be handled and normal termination will take place.
5. If a matching handler is not provided then the program will be abnormally terminated.

Q. How to use multiple catch statements with a single try block?

1. It is possible to provide multiple catch blocks with different handlers along with a single try block.
2. Inside the try block we can provide multiple java statements, which may cause an exception.

3. If the developer does not have an exact idea about the exception object then we can declare multiple catch blocks.
4. In case a try block is having multiple catch blocks then at a time only one catch block will be executed because one try block can throw only one exception.
5. If the try block is having multiple catch blocks then the subclass catch block must be declared before the superclass catch block.

Nested Try-catch block:

Q. What is the use of a nested try-catch block?

1. One try block will be able to throw only one exception object, it means we can handle objects of only one exception by using catch block.
2. To handle multiple exception objects, we can use nested try-catch blocks.
3. If the try-catch block is declared inside the other try block then it is known as a nested try block.
4. It is possible to declare multiple nested try-catch blocks inside the outer try block.
5. It is recommended to start the nested try block from the very first line of outer try block.

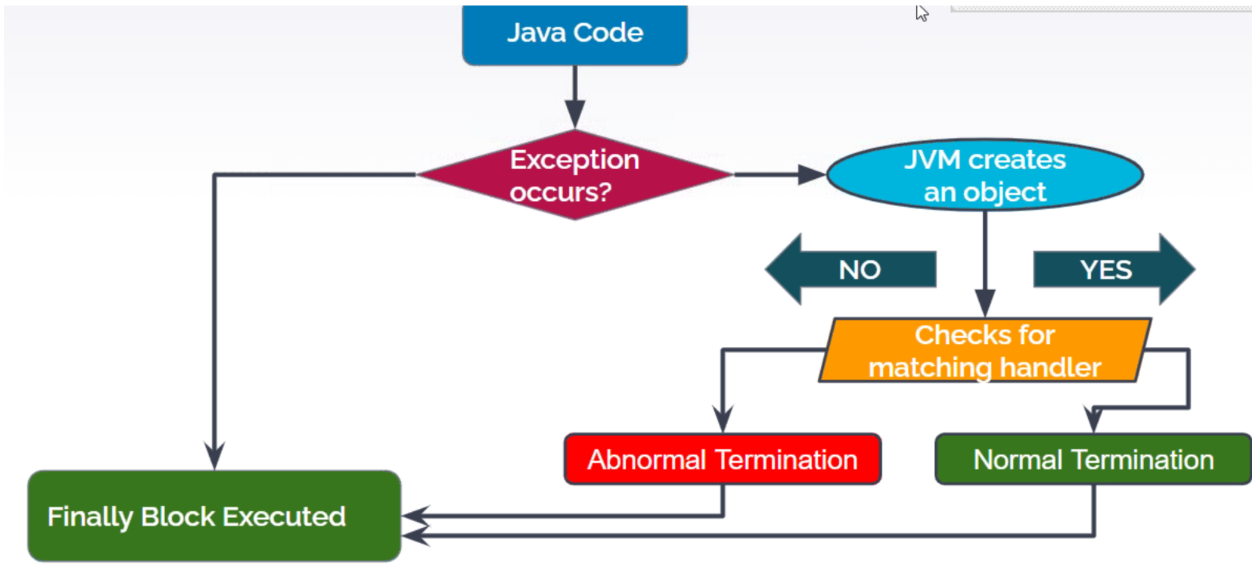
Q What is the use of Finally block?

1. Finally is the special block available in exception handling Mechanism.
2. Finally block can be used to close all the costly resources.
3. Finally block will be executed irrespective of exception scenarios.
4. It means this block will be executed in every possible situation so it does not matter whether an exception occurs or not.
5. All system resources will be considered as costly resources, which can be used for inter-application communication.
6. Finally block must be declared after try or else catch block.
7. We cannot provide multiple finally blocks after try-catch blocks.
8. Following are the possible ways to declare finally block:
    a.
    ```
    Try{
    }
    catch(){
    }
    Finally{
    }
    ```
    b.
    ```
    Try{
    }
    catch(){
    }
    ```

catch(){
}
Finally{
}

c. Try{
}
Finally{
}

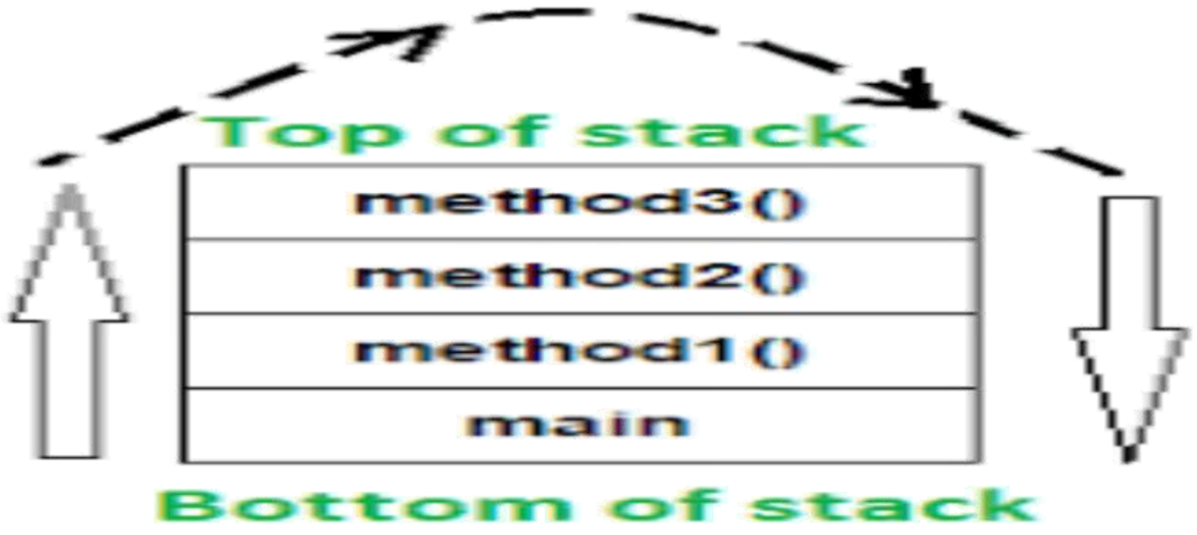
9. Finally block would not be executed only if System.exit() is invoked before it.



Exception Propagation:

Q. What is Exception propagation?

1. If an object is moving from one point to another point then it is known as propagation.
2. In case of exception handling exceptions thrown from top of the stack will drop down to the call stack, it means an exception thrown from one method will be forwarded to another method.
3. This process will be continued until the exception gets handled or else it reaches the main method.
4. The process in which an exception object moves from one method to another method is known as exception propagation.
5. How it works:-



Q. What is the use of printStackTrace()?

1. printStackTrace is the non-static method declared inside the throwable class, which can be used for the diagnosis of exceptions.
2. This method will print detailed information of the exception object. For example: exception name, exception class line number,etc.
3. We can call this method inside the catch block by using the reference of the exception object.

Q. What is the use of the throw keyword?

1. Throw keyword is the part of the exception handling mechanism.
2. Throw keyword mainly used to throw exceptions.
3. Developers can throw custom exceptions with the help of throw keywords.
4. It is mandatory to provide exception class object along with throw keyword

Q. What is Custom Exception?

1. Custom Exception is also known as user-defined exception.
2. Apart from Java Exception classes, developers can create their own exception classes.
3. If a developer is creating an exception class, then it is known as custom-exception.
4. Developer can create 2 types of exception:
    a. Checked Exception.
    b. Un-checked Exception.
5. To create checked custom exceptions we have to extend properties of Exception class.
6. To create custom unchecked exceptions we have to extend properties of RuntimeException class.

Q. What is the use of the throws keyword?

1. Throws keyword is a part of the Exception handling mechanism.
2. Throws keyword can be used to declare an exception of a specific class.
3. Exception Declaration means we have to provide an exception class name along with a throws keyword.
4. Throws keyword plays an important role, in case of propagation of checked exceptions.
5. If you are using throws keyword then no need to provide try-catch block inside that particular method.

throw	throws
Throw keyword is used to create an exception object.	Throws keyword is used for declaration of an exception
Throw keywords must be declared inside the method body.	Throws keyword must be declared along with method signature.
Along with throw keyword we have to provide object of specific exception class	Along with throws keyword we have to provide name of an exception class
We cannot throw multiple objects inside the single method.	We can declare multiple exceptions along with method signature by using throws keyword.
Throw keyword is able to propagate only unchecked exceptions	Throws keyword is able to propagate checked as well as unchecked exceptions

Q. What is the difference in collection, collections, and collection frameworks?

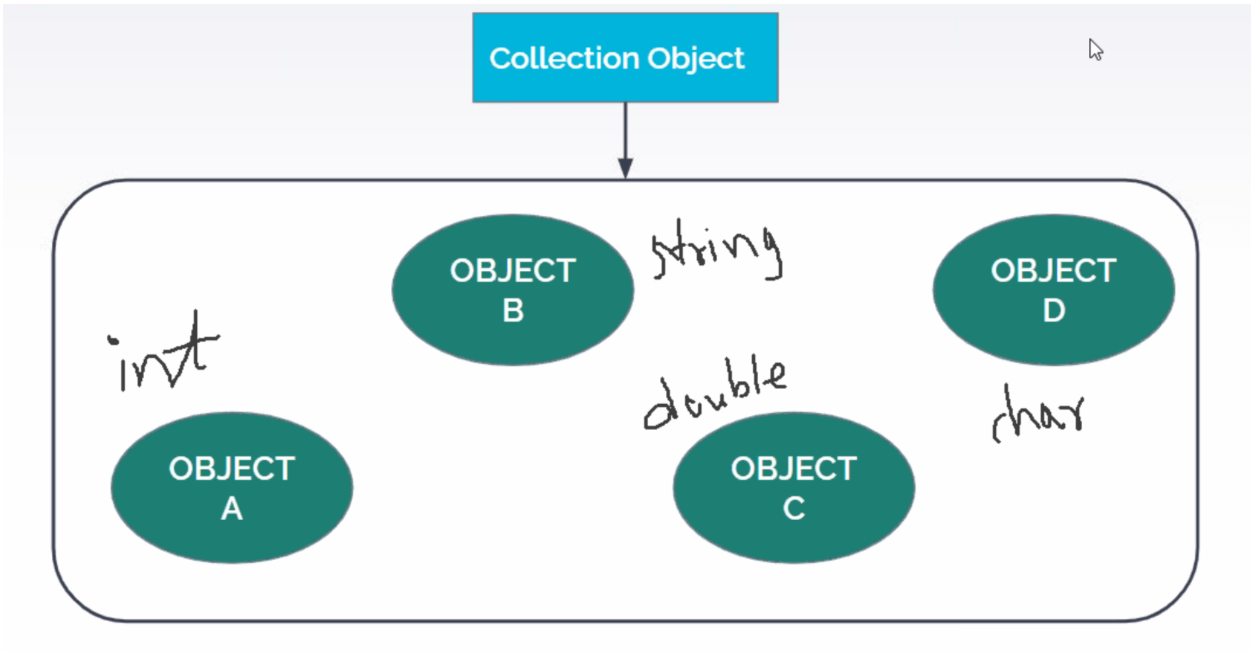
1. Collection is an interface available in the java.util package.
2. Collections is a utility class which contains different methods.
3. Collection framework is a collection of interfaces and classes which can be used to manipulate data.

Q. What are the drawbacks of arrays?

1. Arrays can be used to store only the same type of data i.e homogeneous data.
2. Array variables are fixed width or static width, meaning array is not resizeable.
3. Since arrays are not resizeable there is a chance of memory wastage.
4. Array manipulation is a time consuming process because the array does not have an add or remove method.

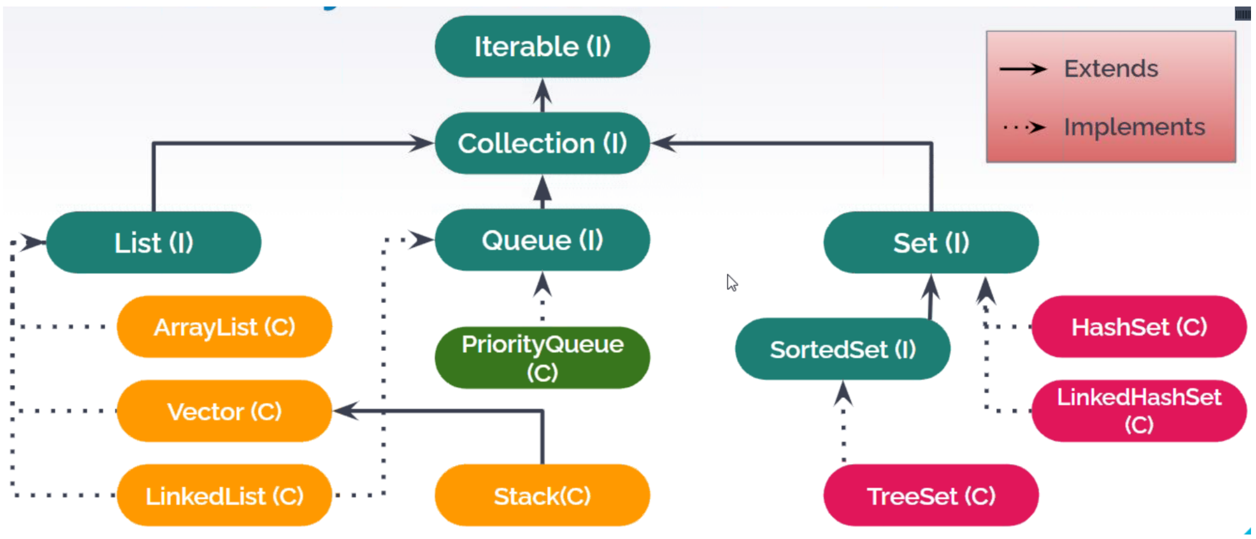
Q. What is Collection?

1. Collection is a group of multiple objects.
2. In other words an entity which acts as a container for multiple objects, is known as a collection.
3. It is possible to store the same as well as different types of information inside the collection object, it means collection classes and interfaces are capable of storing homogeneous and heterogeneous information.
4. We can efficiently process and store the data by using a collection framework.



Q. Benefits of collection framework.

1. Collection framework is resizable in nature it means dynamic structured.
2. It increases reusability and interoperability.
3. Reduced development effort by using core collection classes rather than implementing our own classes.
4. Collection framework will reduce efforts for the code maintenance.



List(Interface:

1. List interface is a type of collection interface.
2. List interface is declared in the java.util package.
3. List interface can be used to store the data in sequential order.
4. List interface allows to store null and duplicate values.
5. Following are the implementation classes of list interface:
    a. ArrayList
    b. Vector
    c. LinkedList

ArrayList:

1. ArrayList is an implementation class for list interface.
2. ArrayList class is declared inside the java.util package.
3. ArrayList internally uses the concept of dynamic array to store the elements.
4. Since it is an implementation class of List interface data will be stored in a sequential manner.
5. ArrayList allows to store null as well as duplicate values.
6. ArrayList implements following marker interfaces:
    a. Serializable.
    b. Clonnable.
    c. RandomAccess.
7. Default capacity of ArrayList is 10 elements.
    a. New Capacity= oldcapacity * 3/2+1(JDK 1.7)
    b. New Capacity= oldcapacity * 3/2(JDK 1.8+)
        i. 10*3/2→ 15
        ii. 15*3/2→ 23

8. Capacity will grow and shrink dynamically. It means if you are going to insert elements in arrayList memory will be allocated automatically, similarly memory will be released if you are going to remove elements from ArrayList.

Q. Methods of ArrayList:

1. add(Object o):
   a. This method can be used to add a new element inside the array List.
   b. Every element should be represented in the form of an object.
2. add(int index, Object o):
   a. This method can be used to add a new element at a specific position.
3. remove(Object o):
   a. This method can be used to remove specific elements from the arraylist.
4. remove(int index):
   a. This method always removes the elements from the specified position.
5. set(int index, Object o):
   a. This method is used to modify the element present at specified index.
6. indexOf(Object o):
   a. This function can be used to get the index number of the specified element.
7. get(int Index):
   a. This function always returns an element present at a specified index.
8. size():
   a. This function always returns the capacity of arrayList.
9. contains(Object o):
   a. This function always checks whether a specified element is present or not inside the arrayList.

Q. What is the difference between generic and non generic arrayList?

1. Generic array list allows to store only specific types of information or elements. Whereas non generic allows to store the same type as well as different types of elements.
2. To create a generic arraylist we have to specify the respective class name, whereas to create a non-generic array list we don't have to specify any class name.

Q. What is the difference between iterator and ListIterator?

1. Both are the interfaces declared in the java.lang package, which can be used to iterate collections.
2. Iterator interface allows to traverse the collection oly in a forward direction, whereas ListIterator supports forward as well as backward traversing