JAVA

1. **Java Virtual Machine**
   - JVM stands for Java Virtual Machine which is a run-time environment for the compiled java class files.

2. **Are JavaScript and Java the same?**
   - Java is an OOP programming language while Java Script is an OOP scripting language.
   - Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only.
   - Java code needs to be compiled while JavaScript code are all in text.
   - They require different plug-ins.

3. **Java Runtime Environment**
   - JRE is what we need to run a Java program and contains set of libraries and other files that JVM uses at run time.
   - JRE = JVM + Library Classes

4. **Java Development Kit**
   - JDK is what we need to compile Java source code and contains JRE, development tools.
   - JDK = JRE + Development tools

5. **What are the popular topics for Java Interview?**
   Some of the popular topics for Java interview are:
   - OOPS Concepts
   - Java String
   - Collections Framework
   - Multithreading
   - Generics
   - Exception Handling
   - Stream API
   - Lambda Expressions
   - Latest Release Features
   - Java EE Frameworks – Spring, Hibernate etc.

6. **What are the advanced Java Topics?**
   Some of the popular topics for Java interview are:
   - Heap and Stack Memory
   - Garbage Collection
   - Reflection API
   - Thread Deadlock
   - Java ClassLoader
   - Java Logging API
   - Internationalization in Java
   - Java Module System

7. **Which class is the superclass of all classes?**
   - `java.lang.Object` is the root class for all the java classes, and we don't need to extend it.

8. **What is the method?**
   - Collection of statements that are grouped together to perform an operation. When you call the **System.out.println()** method, for example, the system actually executes several statements in order to display a message on the console.
   - A method is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name. Think of a method as a subprogram that acts on data and often returns a value. Each method has its own name.

9. **What is the constructor?**
   - A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created.
   - Each time an object is created using new() keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the data members of the same class.

10. Difference between a Constructor and a Method?

- Constructor doesn't have a return type and constructor's name must be same as the class name.
  - Constructor is called automatically when a new object is created. Constructor is invoked implicitly.
  - The Java compiler provides a default constructor if we don't have any constructor.
  - Constructors are not inherited by child classes
- Method have a return and the method's name may or not be same as the class name
  - Method is invoked explicitly.
  - Method is not provided by compiler in any case.
  - Methods are inherited by child classes.

11. What is the difference between a local variable and an instance variable?

- A **local variable** is typically used inside a method, constructor, or a block and has only local scope. Thus, this variable can be used only within the scope of a block.
- The best benefit of having a local variable is that other methods in the class won't be even aware of that variable.
  Example

```
if(x > 100){
    String test = "Alberto";
}
```

- An **instance variable** in Java, is a variable which is bounded to its object itself. These variables are declared within a class, but outside a method. Every object of that class will create its own copy of the variable while using it. Thus, any changes made to the variable won't reflect in any other instances of that class and will be bound to that particular instance only.
  Example

```
class Test{
    public String EmpName;
    public int empAge;
}
```

12. Object Oriented Programming (OOP)

- OOP is a programming language model organized around object rather than actions (logic and functions).
- In other words, OOP mainly focuses on the objects that are required to be manipulated instead of logic. This approach is ideal for the programs large and complex codes and needs to be actively updated or maintained.;
- It makes development and maintenance easier - It provides data hiding - It provides ability to simulate real-world.

OOP language follow 4 principles:

- **Encapsulation** : We can hide direct access to data by using private key and we can access private data by using getter and setter method.
- **Abstraction** : It is a process of hiding implementation details and showing only functionality to the user. Abstraction lets you focus on what the object does instead of how it does it.
- **Inheritance** : It is used to define the relationship between two classes. When a child class acquires all properties and behaviors of parent class known as inheritance. Child class can reuse all the codes written in parent class. It provides the code reusability.
- **Polymorphism** : It is an ability of object to behave in multiple form. The most common use of polymorphism in Java is when a parent class reference type of variable is used to refer to a child class object.

  Example

```
WebDriver driver = new ChromeDriver();
```

  We use method overloading and overriding to achieve Polymorphism.

13. **What is encapsulation and how did you use it?**
    - Data hiding by making variables private and providing public getter and setter methods.
    - In my project I created multiple POJO/BEAN classes in order to manage test data and actual data.
        - EX: I take JSON from API response and convert to object of my POJO class all variables are private with getters and setter.

14. **What is the concept of Abstraction?**
    - In OOP, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.
    - In other words, the user will have the information on what the object does instead of how it does it.
    - In Java, abstraction is achieved using Abstract classes and interfaces.
    - For example: when you log in to your bank account online, you enter your user-id and password and press the login. What happens then, how the input data sent to the server, how it gets verified are all abstracted away from you.

15. **Difference between Abstraction and Encapsulation?**
    - **Abstraction** lets you focus on **what** the object does instead of **how** it does it.
        - **Encapsulation** means hiding the internal details of how the object does something.
    - **Abstraction** is used for hiding the **unwanted** data and giving relevant data.
        - **Encapsulation** means hiding the code and data, and to protect the data from outside.
    - **Abstraction** can achieved by using Abstract class and Interfaces
        - **Encapsulation** can achieved by using "private" keyword.

16. **Difference between Abstract Class and Interface?**
    - Main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that implements a default behavior.
    - A class that is declared with abstract keyword, is known as abstract class. It can have abstract and non--abstract methods.
    - An Interface is a blueprint of a class. It is a template and it is declared with interface keyword. It can have abstract methods, default methods, static methods and public final static variables
    - When we want to use Abstract class, we use "**extend**" keyword. When we want to use Interface, we use "implement" keyword.
    - Abstract class and interface both are used to achieve abstraction Both cannot be instantiated; we cannot create an object.

17. **What is Polymorphism?**
    - Polymorphism is a very important concept in OOP because;
        - it enables to change the behavior of the applications in the run time based on the object on which the invocation happens.
        - by Polymorphism; one object can have different forms
    - Two types → **Compile Time** which is Static and **Run Time** Polymorphism which is related with child and parent class.
    - Polymorphism is implemented using the concept of Method overloading and method overriding. This can only happen when the classes are under the parent and child relationship using inheritance.

18. **What is Inheritance?**
    - Inheritance represents the **IS-A** relationship which is also known as a parent-child relationship.
    - It is the mechanism in java by which one class is allowed to inherit the features (fields and methods) of another class.
    - The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.
    - When you inherit from an existing class, you can reuse methods and fields of the parent class.
    - Moreover, you can add new methods and fields in your current class also.

- Code reuse is the most important benefit of inheritance because subclasses inherits the variables and methods of superclass.

19. Important terminology in Inheritance?

- **Class:** the group of objects which have common properties. It is a template or blueprint from which objects are created.
- **SuperClass**: the class being inherited from (or a base class or a parent class).
- **SubClass**: the class that inherits from another class(or a derived class, extended class, or child class).
  - The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability**: a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

20. Difference between Polymorphism and Inheritance

- Like in real world, Inheritance is used to define the relationship between two classes. It is similar to Father-Son relationship. In Java, we have Parent class (also known as super class) and child class (also known as subclass). Similar to the real-world, Child inherits Parents qualities, methods and codes.
  - A child class can reuse all the codes written in Parent class and only write code for behavior which is different than the Parent.
  - Inheritance is actually meant for code reuse.
- On the other hand, Polymorphism is an ability of object to behave in multiple form.
  - It is classified as overloading and overriding.
- By the way, they are actually related to each other, because its inheritance which makes Polymorphism possible, without any relationship between two class. It is not possible to write polymorphic code.
  - Dynamic Polymorphism → Overriding
  - Static Polymorphism → Overloading

21. Difference between method Overloading and method Overriding?

- First and most important difference between overloading and overriding is that,
  - in case of overloading , method name must be the same, but the parameters must be different;
  - in case of overriding , method name and parameters must be same
- Second major difference between method overloading and overriding is that;
  - We can overload method in the same class but method overriding occurs in two classes that have inheritance relationship.
- We cannot override static, final and private method in Java, but we can overload static, final and private method in Java.
- In method overloading , return type can be same or different. In method overriding , return type must be same or covariant type.

22. What is immutable ?

- Immutable means that once the constructor for an object has completed execution that instance can't be altered.
- This is useful as it means you can pass references to the object around, without worrying that someone else is going to change its contents.
  Especially when dealing with concurrency, there are no locking issues with objects that never change.

```
class Foo {
   private final String myvar;
   public Foo (final String initialValue)
   this.myvar = initialValue;
}
   Public String getValue (){
   return this.myvar;
   }
```

23. What is static binding vs dynamic/runtime binding?

- Static binding is overloading, and dynamic binding is method overloading

24. What is Access modifier and what are the different access modifiers?

- Java provides a number of access modifiers to set access levels for classes, variables, methods, and constructors.
  - Visible to the package, the default. No modifiers are needed.
  - Visible to the class only (private).
  - Visible to the world (public).
  - Visible to the package and all subclasses (protected).

25. Difference between Public, Private and Protected modifier in Java?

- In Java, access modifier which specifies accessibility of class, methods and variables. There are four access modifiers in Java namely Public, Private, Protected and Default.
- The difference between these access-modifies is that;
  - The most importantly is the level of accessibility.
  - Public is accessible to anywhere
  - Private is only accessible in the same class which is declared
  - Default is accessible only inside the same package
  - Protected is accessible inside the same package and also outside the package but only the child classes.
- We cannot use private or protected modifier with a top--level class.
- We should also keep in mind that access modifier cannot applied for local variable public, private or protected in Java.

26. Difference between Set, List and Map in Java?

- Set, List and Map are 3 important interface of Java collection framework.
  - List provides *ordered* and indexed collection which *may contain duplication* .
  - Set provides *un-ordered* collection of unique objects. Set *doesn't allowed duplication* . List and Set are both extend collection interface.
  - Map provides a data structure based on Key Value. Key is always unique, value can be dupl.

27. When to use List, Set and Map?

- If we need to access elements frequently by using index, List is a way to go ArrayList provides faster access with index.
- If we want to store elements and want them to maintain an order, List is an ordered collection and maintains order.
- If we want to create collection of unique elements without duplicates than choose any Set implementation. (HashSet…)
- If we want store data in form Key and Value than Map is the way to go. We can choose from HashMap, Hashtable…

28. What is Array?

- An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed. You have seen an example of arrays already, in the main method of the "Hello World!" application. This section discusses arrays in greater detail.
- Each item in an array is called an element, and each element is accessed by its numerical index. As shown in the preceding illustration, numbering begins with 0. The 9th element, for example, would therefore be accessed at index 8.
- **Advantage of Java Array**
  - Code Optimization: It makes the code optimized, we can retrieve or sort the data easily.
  - Random access: We can get any data located at any index position.
- **Disadvantage of Java Array**
  - Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

29. How do you find if ArrayList contains duplicates or not?

- There are several ways available. Shortest one is `.stream().distinct().count()` method
  `list.size() != list.stream().distinct().count()`
- Other methods:

```
//METHOD 1
public static <T> boolean containsUnique(List<T> list){ Set<T> set = new HashSet<>();
return list.stream().allMatch(t -> set.add(t));
}

//METHOD 2
public static <T> boolean containsUnique(List<T> list){ return list.stream().allMatch(new
HashSet<>()::add);
} // seems to be the best not only because it can handle pure streams, but also because it stops on
the first duplicate (while #1 and #2 always iterate till the end)

//METHOD 3
public static <T> boolean containsUnique(List<T> list){
    Set<T> set = new HashSet<>();
    for (T t: list){
        if (!set.add(t))
return false; }
```

30. Difference between Arrays and ArrayList in Java?

- Array is a part of core Java programming and has special syntax ArrayList is part of collection framework and implement List interface
- Major difference is that; Array is a fixed length data structure, so we can change length of Array one created, ArrayList is resizable.
- The other major one is that Array can contain both primitives and objects. ArrayList can only contain objects. It cannot contain primitive types.
- Also, we can compare Array and ArrayList on how to calculate length of Array or size of ArrayList. We use length for an Array, we use size() method for an ArrayList.

Array	ArrayList

int[ ] arr = {6,9,1};
• arr.length
• Arrays.sort(arry); //import java.util.Arrays
• Java also provides a convenient way to search, but only if the array is already sorted.
Arrays.binarySearch(arry, value);
• string[ ] [ ] marry = new string [3] [2];
• Arrays.asList(arry);
• Arrays.toString(arry);
• Arrays.deepToString(arry); //for multidimensional
	ArrayList list = new ArrayList();
	• list.add(obj);
	• list.add(index position, obj);
	• list.remove(obj);
	• list.set(index position, new obj); //replace object
	• list.isEmpty(); //boolean
	• list.size();
	• list.clear();
	• list.contains(obj);
	• list.get(int index);
	• list.toArray();
	• Sorting → Collection.sort(list);

31. What is thread safe or Synchronized?

- Thread safety is very important, and it is the process to make our program safe to use in multi-threaded environment, there are different ways through which we can make our program thread safe.
- **Synchronization** is the easiest and most widely used tool for thread safety.

- JVM guarantees that synchronized code will be executed by only one thread at a time.
- JAVA keyword **synchronized** is used to create synchronized code and internally it uses locks on Object or Class to make sure only one thread is executing the synchronized code.
- I mean Java synchronization works on locking and unlocking of the resource, so no thread enters into synchronized code.
- We can use synchronized keyword in two ways, one is to make a complete method synchronized and other way is to create synchronized block.

32. How do you sort an object that you created?

- Sort it will be able to sort.
- Also, I can store my objects into a TreeSet or TreeMap → Ex: NEXT PAGE
- Java provides a number of ways to sort a list.
  - COMPARABLE - COMPARATOR interfaces can be used for sorting. In these cases, We should override the compareTo method.
- Another way is List interface sort method which can use a comparator. With this method we can sort ascending or descending.

```
users.sort(Comparator.comparing(User::getUserID));
```

- If we don't want to modify the original list, but return a new sorted list; then we can use the sorted() method from the Stream interface…

```
List<User> sortedUsers = users.stream()
  .sorted(Comparator.comparing(User::getUserID))
  .collect(Collectors.toList());
```

33. Difference between Hashtable and HashMap in Java?

There are several differences between HashMap and Hashtable in Java:

- Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.
- Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.
- For example; one of HashMap's subclasses is LinkedHashMap, so in the event that you'd want predictable iteration order (which is insertion order by default), you could easily swap out the HashMap for a LinkedHashMap. This wouldn't be as easy if you were using Hashtable.

If synchronization is not an issue for me, I' prefer using HashMap. If it becomes an issue, then I prefer Collections.synchronizedMap() or ConcurrentHashMap.

- Both Hashtable and HashMap implements Map interface and both are Key and Value.
- HashMap is not thread-safe while Hashtable is a thread-safe collection.
- Second important difference is performance since HashMap is not synchronized.
  It performed better than Hashtable. → Collections.synchronizedMap(…Map…);

34. How would you handle Exception?

I would use try-catch-finally approach to handle the Exception

- 1- I would put my code that might generate an exception inside a try-catch block. With try-catch block I can rethrow an exception or try to perform my recovery steps. Also, If needed I can use multi or Union Catch blocks
- 2- I can also use throws keyword. BUT it does mean that anyone that calls my method now needs to handle it too!
- 3- Another way is AutoCloseable: When we place references that are AutoCloseable in the try declaration, then we don't need to close the resource ourselves. We can still use a finally block, though, to do any other kind of cleanup we want. try-with

35. TreeSet vs TreeMap

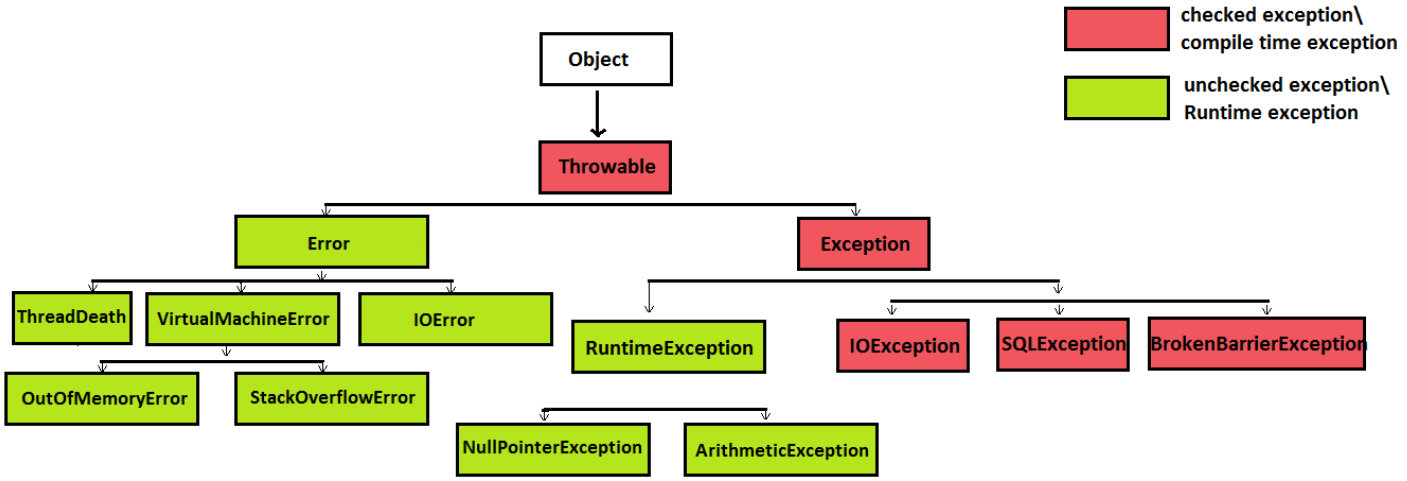
- TreeSet: Can contain only unique values - is sorted in ascending order
- TreeMap: can contain only unique keys. - keys are sorted in ascending order

36. final vs finalize vs finally ?

- **final** → is a keyword and used to apply restrictions on class, method and variable.
  - final Class CAN'T be Inherited
  - final Method CAN'T be Overridden
  - final Variable value CAN'T be changed.
- **finally** → is a block and used to place important code, it will be executed whether exception handled or not
- **finalize** → is a method and used to perform clean-up processing before Object is Garbage collected.

37. Difference between Error and Exception in Java?

- Both Error and Exception are derived from Throwable in Java.
- Error represent errors which are generally cannot be handled.
  For examples: OutOfMemoryError, NoClassDefFoundError
- On the other hand, Exception represent errors which can be catch and dealt.
  For examples> IOException, NullPointerException
- Exception is divided in two categories checked and unchecked Exception. Checked Exception require a mandatory try-catch code block to handle it. Unchecked Exception mostly represent programming errors (NullPointerException or RuntimeException)
- Errors are unchecked exception and the developer is not required to do anything with these
- **All the Errors are Exceptions, but the reverse is not true.**
- In general Errors are which nobody can control or guess when it happened, on the other hand Exception can be guessed and can be handled



38. Difference between RuntimeException and CheckedException in Java?

- Exception are divided in two categories Runtime (unchecked) Exception and CheckedException.
- Main difference between RuntimeException and CheckedException is that, it is mandatory to provide try-catch to handle CheckedException while in case of RuntimeException is not mandatory.
- Some of the most common Exception like NullPointerExceptio, ArrayIndexOutOfBound, ClassNotFoundException, IOException.

First I want to remind that Java Exceptions are divided in two categories RuntimeException also known as unchecked

Exception and checked (compile time) Exception.
Main difference between RuntimeException and checked Exception is that, It is mandatory to provide try catch or try finally block to handle checked Exception and failure to do so will result in compile time error, while in case of RuntimeException this is not mandatory.
Some of the most common Exception like NullPointerException, ArrayIndexOutOfBoundException are unchecked and they are descended from java.lang.RuntimeException.
Popular example of checked Exceptions are ClassNotFoundException and IOException and that's the reason you need to provide a try catch finally block while performing file operations in Java as many of them throws IOException.
If you I ask my personal opinion, I think Checked Exceptions makes our code code UGLY by adding boiler plate code in for of try-catch finally block.

39. Difference between throw and throws in Java?

- throw and throws are two keywords related to Exception feature of Java programming language.
- throw keyword is used to throw an exception explicitly, on the other hand, throws keyword is used to declare an exception which means it works similar to the try--catch block.
- If we see syntax wise than throw is followed by an instance of Exception class throws is followed by exception class names.
- throw new ArithmeticException ("Arithmetic Exception"); throws ArithmeticException;
- throw keyword is used to method body, while throws is used in method signature to declare the exception.

Both of them are two keywords related to Exception feature of Java. As I remember the main difference between throw and throws is in their usage and functionality.

- throws is used in method signature to declare Exception possibly thrown by any method, for example

```
public void shutdown() throws IOException{
        throw new IOException("Unable to shutdown");
 }
```

But throw is actually used to throw Exception in Java code.

```
            Throw new Exception("is Not able to initialized");
```

In other words; throws keyword cannot be used anywhere exception method signature while throw keyword can be used inside method or static initializer block provided sufficient exception handling.
Oh, I remember one other thing about throw, throw keyword can also be used to break a switch statement without using break keyword

40. Difference between Object and Class?

- Class is a blueprint or template which you can create as many objects as you like Object is a member or instance of a class
- Class is declared using class keyword, Object is created through new keyword mainly.

A class is a template for objects. A class defines object properties including a valid range of values, and a default value. A class also describes object behavior. An object is a member or an "instance" of a class and has states and behaviors in which all of its properties have values that you either explicitly define or that are defined by default settings.
Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
If we compare them there are many differences but let me tell you some of them which are important to know;

- There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization. There is only one way to define class in java using class keyword.
- Object is created many times as per requirement. Class is declared once.
- Object is an instance of a class. Class is a blueprint or template from which objects are created.
- Object is a physical entity. Class is a logical entity.

For Example:

Class: Human	**Object**: Man, Woman	**Class**: Mobile phone	**Object**: iPhone, Samsung, Moto
Class: Fruit	**Object**: Apple, Banana, Mango, Guava	**Class**: Food	**Object**: Pizza, Burger, Samosa

41. StringBuffer and StringBuilder?

- The main difference is StringBuffer is synchronized while StringBuilder is non-synchronized. So, StringBuilder can be called simultaneously. And this makes StringBuilder more efficient.
- StringBuffer is synchronized, StringBuilder is non-synchronized
- StringBuilder is more efficient than StringBuffer
- Constructor;
  - StringBuilder() → created an empty string with the initial **capacity of 16**.
  - StringBuilder(str str) → created an StringBuilder the specified string.
  - StringBuilder(int length) → created an empty string with the specified capacity as length.

● Method;

- StringBuilder str = new StringBuilder("Hello");
- str.append("Java"); → //Hello Java
- str.insert(1,"Java"); → //HJavaello
- str.replace(1,3,"Java"); → //HJavalo
- str.delete(1,3); → //Hlo
- str.reverse(); → //olleH

```
string str = "Hello";
string reversed = " ";

for (int i = str.length()-1; i>=0 ; i--){
   reversed += str.charAt(i);
}
sysout(reversed);
```

42. What is `finalize()`?

- finalize() method is a protected and non-static method of java.lang.Object class.
- This method is available in all objects that we create in java.
- This method is used to perform some final operations or clean-up operations on an object before it is removed from the memory.
- We can also override the finalize() method to keep those operations we want to perform before an object is destroyed. It can be called. object.finalize();

43. What is `final` keyword?

- final keyword is used with Class to make sure no other class can extend it, for example String class is final and we can't extend it.
- We can use the final keyword with methods to make sure child classes can't override it.
- final keyword can be used with variables to make sure that it can be assigned only once. However, the state of the variable can be changed, for example, we can assign a final variable to an object only once, but the object variables can change later on.
- Java interface variables are by default final and static.

44. What is `static` keyword?

- static keyword can be used with class level variables to make it global i.e all the objects will share the same variable.
- static keyword can be used with methods also. A static method can access only static variables of class and invoke only static methods of the class.

45. What is `system.gc()`?

- A request to JVM to run Garbage collector to free up memory
- Doesn't always work

The java.lang.System.gc() method runs the garbage collector. Calling this suggests that the Java Virtual Machine expend effort toward recycling unused objects in order to make the memory they currently occupy available for quick reuse.
It is not a command but is a request. It is up to garbage collector to honor this request

46. Important String Methods?

Method	Description
char charAt(int index)	returns char value for the particular index
int length()	returns string length
String substring(int beginIndex)	returns substring for given begin index
String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index
boolean contains(CharSequence s)	returns true or false after matching the sequence of char value
boolean equals(Object another)	checks the equality of string with object
boolean isEmpty()	checks if string is empty
String concat(String str)	concatinates specified string
String replace(char old, char new)	replaces all occurrences of specified char value
String replace(CharSequence old, CharSequence new)	replaces all occurrences of specified CharSequence
static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
String[] split(String regex)	returns splitted string matching regex
String[] split(String regex, int limit)	returns splitted string matching regex and limit
String intern()	returns interned string
int indexOf(int ch)	returns specified char value index
int indexOf(int ch, int fromIndex)	returns specified char value index starting with given index
int indexOf(String substring)	returns specified substring index
int indexOf(String substring, int fromIndex)	returns specified substring index starting with given index
String toLowerCase()	returns string in lowercase.
String toLowerCase(Locale l)	returns string in lowercase using specified locale.
String toUpperCase()	returns string in uppercase.
String toUpperCase(Locale l)	returns string in uppercase using specified locale.
String trim()	removes beginning and ending spaces of this string.
static String valueOf(int value)	converts given type into string. It is overloaded.

47. What's the difference between IS-A and HAS-A relationship?

- **IS-A** is based on inheritance → This thing is a type of that thing
- **HAS-A** relationships are based on usage
  - Ex: class A HAS -A B if code in Class A has a reference to an instance of class B

```
public Horse{
   private Halter myHalter;
   public void jump(){
            Sysout "im jumping"
```

- You are calling a Halter instance variable to use jump method that is coming from horse class - what this does is that it is means that Horse HAS-A Halter
- Horse class has a Halter, because Horse declares an instance variable of type Halter. When code invokes tie() on the Horse object's Halter instance variable -}
- Abstract class have constructors while interface don't have one

48. What is Iterator and difference between for each loop?

- Iterator works with ArrayList and not array.
- It will help us Iterate through the elements.
- Difference is with iterator you can make changes(remove item) to the list while iterating.
- within for each loop we cannot make changes to our list

49. Java Collection Framework

Two types of Collection (Be careful not to mix them up)

- **java.util.Collection** - interface from Set and List extend (not implement)
  - **Set** *(Unique things)* - DOES NOT ALLOW DUPLICATES. Classes that Implement Set;
    - **HashSet** → Use when you don't want any duplicates and you don't care about order when you iterate through
      - Unordered and Unsorted
    - **LinkedHashSet** → Ordered version of HashSet and Use over HashSet when you care about iteration order
    - **SortedSet**
    - **TreeSet** → Elements will be in ascending order, according to the natural order of the elements
      - Can also customize constructor to implement your own rules of the natural order
  - **List** *(list of things)* - cares about the index. Classes that implement List;
    - **LinkedList** → Ordered by index position and elements are doubly-linked to one another
      - It is a good choice for implementing stack and queue
      - Iterates more slowly than ArrayList but fast insertion and deletion
    - **Vector** → Same as ArrayList BUT vector methods are synchronized (thread-safe)
    - **ArrayList** → Fast iteration and Fast random access and ordered(by index)
      - Also unsorted (but can invoke Collections.sort() to sort it)
- **java.util.Collections** - a class that holds static utility methods for use with collections; Includes add, remove, contains, size, and iterator, etc.
  - **Map** *(things with unique ID)* → Important: none of the Map-related classes and interfaces extend form Collection. The implementation classes of Map are thought of "collections", not Collection. Classes that implement Map;
    - **Hashtable**
      - Same as HashMap BUT HashTable methods are synchronized (REMEMBER. ONLY METHODS ARE SYNCHRONIZED, NOT CLASSES OR VARIABLES)
      - Hashtable won't let you have anything NULL(NO NULLS AT ALL)
    - **LinkedHashMap**
      - Maintains insertion order(or optionally, access order)
      - Slower than HashMap for adding/removing elements but FASTER ITERATION
    - **HashMap** → Unsorted and Unordered & Allows one null KEY and multiple null values in a collection
      - KeySet()
      - Map.keySet() - returns a set of Keys
      - Map.keySet().size - return # of keys
    - **SortedMap** → TreeMap
- The implementation classes of Set,List, and Map can NEVER be both sorted but unordered, can be all other combinations.

50. How to convert float to String?

```
float f = Float.parseFloat("25");
String s = Float.toString(25.0f);
```

51. Let's say you have an "int b=3; and int a=4;" how can you swap them?

```
// one-line methods
a = a ^ b ^ (b = a);
b = (a + b) - (a = b);
a += b - (b = a);

int temp = a; // temporary variable
a = b; b = temp;
```

52. Do you know typecasting? What is casting?

- **Auto-boxing** → is a process when you take a primitive value and assign into wrapper class object int i=10;

```
Integer n=i;
Integer num=200;
Integer num2=new Integer(400);//NO BOXING
```

- **Un-boxing** → is a process when you take Wrapper class object and convert to primitive.

```
Integer num2=new Integer(400);
Integer num=200;
int i=num2;
```

- Assigning a value of one type to a variable of another type is known as Type Casting.

53. What is the output for this program?

```
for (int i = 0; i < 3; i++) {
   for (int j = 3; j >= 0; j--) {
            if (i == j)
            continue;
            System.out.println(i + " " + j);
   }
}
```

Output: 1 0 2 3 2 1 2 0

54. How do you use an abstract class in your project give me an example?

- These concepts are commonly used in framework development. Abstract class is used in defining a common super class while writing Page Object Model layer of the framework. We usually create an abstract class named BasePage to have all common members for every page written in this class example **`getPageTitle().`**
- Then each Page class (HomePage, LoginPage, DashboardPage etc.) inherit from BasePage. Sometimes one may need to change the behavior of methods implemented in superclass. So, subclass has freedom to override that method where we use polymorphism. This is how we use Abstract class in real projects.

55. What is the difference between pass-by-value and pass-by-reference? pass by value & pass by reference?

- Passing by value means that the value of the function parameter is copied into another location of your memory, and when accessing or modifying the variable within your function, only the copy is accessed/modified, and the original value is left untouched. Passing by value is how your values are passed on most of the time.
- Passing by reference means that the memory address of the variable (a pointer to the memory location) is passed to the function. This is unlike passing by value, where the value of a variable is passed on. In the examples, the memory address of myAge is 106. When passing myAge to the function increaseAgeByRef, the variable used within the function (age in this example) still points to the same memory address as the original variable myAge (Hint: the & symbol in front of the function parameter is used in many programming languages to get the reference/pointer of a variable).