05/10/23

1. What is Java?

Java is high level object Oriented platform independent programming language.

2. Explain how java works has platform independent?

or

Explain & WORA architecture

After succesful compilation byte code gets generated. The byte code is independent or common for all different platforms.

The JRE of JVM ~~is~~ interprets the bytecode & converts the bytecode

3. What is bytecode?

It is the instruction generated by the compiler after succesful compilation.

It is an intermediate code.

4. Explain java compilation.

Compilation is two step process

1. check the syntax
2. If java program is syntatically correct bytecode gets generated

5. Diff b/w JDK, JRE, JVM

JVM is specification (rules, regulation) for JRE.

JRE is a physicaly form (implementation) of JVM.

JDK is a form of SDK it internaly contains the ~~all~~ java compiler, ~~JRE~~ the necessary resources to develop as well as to execute java programs or applications.

6. Components of JDK

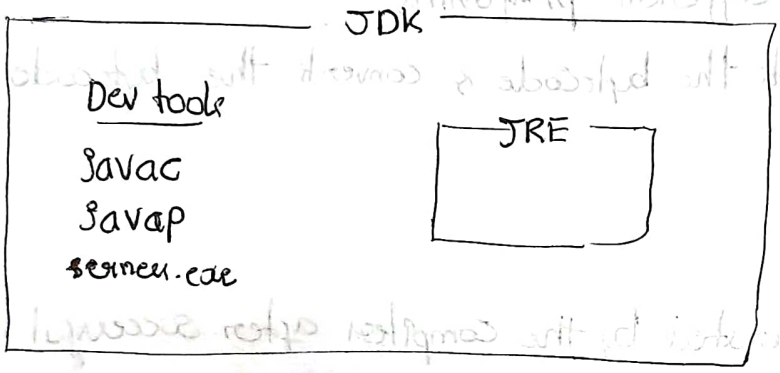
JRE, APIs, libraries

7. Is JRE platform dependent or independent

platform dependent, bcoz it is different for different platform.

8. Is JDK platform dependent or independent

The developmental tools of the JDK are platform independent. whereas JDK is platform dependent bcoz JDK internal contains JRE.



9. What are the object oriented concepts?

Class, object, Inheritance, polymorphism, Encapsulation, Abstraction & Association

10. What are object oriented principles?

Inheritance, polymorphism, Encapsulation, Abstraction

11. Is java 100% object oriented programming language?

NO, Because of the primitive datatypes, but primitive are supported with wrapper classes in java.

12. What is variable? what are the types?

It is a container or dataholder which is used to store data.

Local & Global

Instance (non-~~primitive~~ static) Static (class)

13. How do you initialize instance variables?

1. At the time declaration
2. Using object reference
3. Using Constructor
4. Using block
5. By using methods (getter methods)

14. Diff b/w. local & global variables.

Local	Global
* Have limited scope	* Having larger scope.
* Local variable is declared within the local scope i.e within the method, Constructor, block, etc,.	* Global variable is declared within the global scope i.e, within the cl & outside the local scope.
* Local variables can only be accessed within the local scope	* Global variable can be accessed within the same class, outside th class & even outside the package.
* Local Default initialization is not applicable for local variables.	* Default initialization is applicable
* Access modifiers are not applicable for local variables. (final can be applicable)	* Access Modifiers are applicable for global variables.
* Local variables can't be static.	* Global variables can be static
* Local variable can't be inherited.	* Global variable can be inherited
* Local variable can't be accessed using this keyword	* Can be accessed using this keyw
* Local variables are stored in stack memory.	* Global variables are stored in heap memory. (class area/ static pool is also part of heap memory) from JDK 1.8

15. Diff b/w Static Global Variables & Non-static Global Variables?

* Static variable belong to the class.	* Non-static variable belong to the Object.
* Static Global variable represents the data (with which) is common to all the objects of the same class.	* Non-static Global variable so called Instance variable represents the object specific data. (state of an object).
* Static global variable will have only one copy in the memory.	* Non-static Global variable will have multiple copies dependends on the objects.
* Static global variable is accessed using the class name.	* Is accessed using object reference.
* Is allocated at the time of class loading.	* Is created in memory at the time of object creation.

16. Is default initialization is applicable for static variable? → YES

17. What are the types of method?

1. Concrete method
2. Abstract method

18. Diff b/w final method & private method.

Final method can be inherited but cannot be overridden	private method can't be inherited.

19. Diff b/w Static & Non-static method.

* If the behaviour is polymorphic in nature then method should be non-static method.	* When the behaviour is same for different Objects then it should be Static method.
	* When there is mathmatical calculate.

* Non-static method can be inherited.
* Non-static method can be overridden
* Non-static method can be invoked using class name
* Nonstatic method gets loaded into memory when we invoke the method.

* static method can't be inherited.
* static method can't be overridden
* static method can be invoked using class name.
* static method gets loaded into memory at the time of class loading.

20. Explain Constructor with real time Example.

Constructor is one of the member of class where, Constructor name is same as class name, which is used for the initialising the resources (non-static / Instance variables)

Example: Constructor in ArrayList, LinkedList, Vector

21. what are the different way to invoke a Constructor?

1. using new keyword
2. using this() & super()
3. using super()

IS-A ⟹ Inheritance

HAS-A ⟹ Association
- → Aggregation
- → Composition

How we call A constructor in C Class?

```
class A
{
    A(){
    }
}
Class B extends A
{
    B(){
    }
}
Class C extends B
{
    C(){
    new A();
    }
}
```

We can't use super() for calling A()

22. What is Constructor Overloading & give real-time example.

Having multiple Constructors with change in the signature.

Ex: ArrayList, Vector, LinkedList.

23. Can the Constructor be Inherited? Why?

NO.

Because Constructor name should be same as the class name, if the Constructor is inherited to subclass then it creates Confusion, i.e., the subclass name & Constructor is different.

NOTE: But we can invoke the superclass constor from the subclass Constructor by using Constructor chaining using super().

24. Diff b/w Constructor & a Method.

Constructor	Method
* Constructor name should be same as ClassName.	* Method name can be anything.
* Constructor can't have returntype.	* Method must have returntype.
* Constructor can't be inherited.	* method can be inherited.
* Constructor can't be static & final.	* Method can be static or final.
* Constructor can be invoked using new keyword, this() and super().	* Method can't be invoked using new keyword using classname objectreference, methodname.
* purpose of Constructor is to initialize Variables or resources.	* purpose of method is to perform some functionality.
* Constructors can't be overridden.	* Method can be overridden.
* Constructor can't be abstract.	* method can be abstract.

* There is a default constructor generated by compiler.

* There is no default method generated by Compiler

25. Which is the most used Object Oriented principle?

Inheritance, because every class we write inherits the Object class

26. Where do we find Object class?

Inside the JRE library as part of java.lang package.

27. Where do we find all the inbuild class & interface of java?

(jar file) jre/lib/rt.jar

28. Diff b/w open source & free software

Free Software	Open Source
* Source code is not available	* we get the sourcecode.

29. Can we convert bytecode into java ~~program~~ code.

Yes, using decompiler

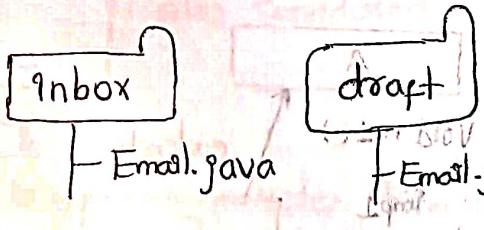
30. Why to set environment variable (path) after installation of java?

To locate java resources.

31. Is it possible to import two different classes with the same name bu from different packages?

NO, not possible

```
Import inbox.Email;
class Test{
    void doSomething(){
        Email e1 = new Email();
        Email e2 = new Email();
    draft.Email e3 = new draft.Email();
    } Email e4 = new Email();
```



32. Can you name few inbuilt packages of java?

java.util, java.lang, java.io, java.sql, java.math

33. Default package of java → java.lang

34. What is Inheritance, Explain with real time Example.

The process of acquiring the properties of one class to another class.

35 Why is Inheritance?

→ TO Code reusability
→ TO achieve generalization
→ TO avoid code duplication
→ Indirectly helps to achieve polymorphism.

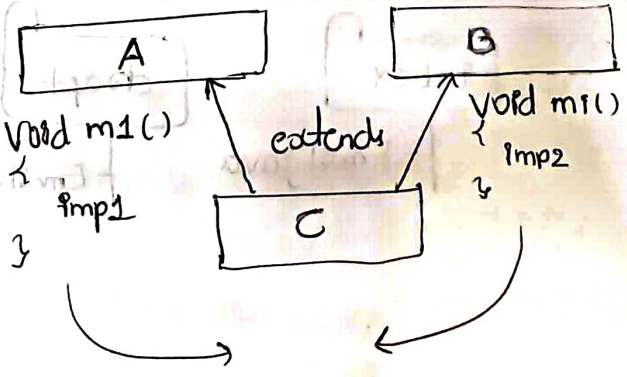
* Every class is example for Single level because Inherit Object class.
* NullpointerException Example of multilevel Inheritance.

36. Is multiple Inheritance is possible in java?

By using interface it is possible but not using class.

37. Why multiple Inheritance is not possible in java?

* Ambiguity while method binding.
* Ambiguity while Constructor chaining



C ref = new C();

ref.m1(); method-call

38. How do you avoid inheritance in a class?

1. make a class has final so that it can't inherit.

2. Write a private constructor in superclass. so that subclass can't invoke using super().

39. Why Constructor chaining? / How constructor chaining helps/Why CC Imp?

* TO enhance code reusability.
* TO avoid code duplication.

40. What is method Overriding? Explain with real time Example.

changing the subclass specific method implementation of inherited method.

Covariant returntype:

↳ returntype of reduced type (subclass type)

Example: Overriding toString(), hashCode(), equals(), (getMessage(), printStackTrace(). are method defined in throwable class and overridden in all its subclass).

41. Diff b/w Method Overloading & Method Overriding.

* For overloading inheritance is not mandatory.	* For Overriding inheritance is mandatory.
* We ignore returntype in overloading. (name & signature is considered)	* Returntype is also considered
* Overloading demonstrates compileTime polymorphism.	* Overriding demonstrates, RunTime polymorphism.
* Static method can be overloaded.	* Static method can't be overridden.
* private method can be overloaded.	* private method can't be overridden

* method overloading doesn't have any annotation.

* method overriding as annotation

6/10/23

42. Explain MethodOverloading with real-time Example.

Having multiple methods in a class with the same name but different signatures is called Method Overloading.

↳ no of arguments, change in datatypes, sequence or order of datatypes.

Method overloading presents different forms of doing the same activity. Hence, method name must be same.

Realworld Ex: ① class Website{

```
void login(String userName, String pwd)
{
}
void login(long contactNum, int otp)
{
}
}
```

② Unlocking / ~~Opening~~ the phone in different ways like pattern, pin, facelock, fingerprint

7/10/23

Realtime: 1. Sort method in ~~list~~ a collections

```
public static void sort(List list)
public static void sort(List lst, Comparator cfr)
```

2. public boolean add(<E> element)
   public void add(int index, <E> element)

3. println()
   println(String val)
   println(int val)
   println(boolean b)

43. Can we overload abstract method?

Yes we can overload

44. Can we overload one Constructor & one method together?

NO

45. Is overloading is possible in two different class?

YES

46. What is the difference b/w this & this() , super & super()

* this is a keyword
* which is use to invoke method & Variables of same class

* this() is not a keyword
* this() is use to invoke overload Constructor of the same class

---

* super is a keyword which is use to invoke the super class variables & methods.

* super() is use to invoke the super class Constructor

47. Why main() is static?

There are three reasons why main method is static

1. Main() doesn't exhibit polymorphic behaviour.
2. Main() is the first method that should be executed, hence it must be loaded first. So, main() is static because ~~it hence~~ it get stored loaded at the time of class loading
3. JVM can invoke the main() without creation of object

48. Why main() gets String[] as a argument?

To accept the command line arguments

(For main() arguments can be passed using command prompt by providing space)

Command line argument:

The multiple data that we can pass to the main method as an argument th the command prompt.

String[] can take any type of data.

Command prompt

javac Demo.java

java Demo abbs anoop true 25 828.56 (command line argument)

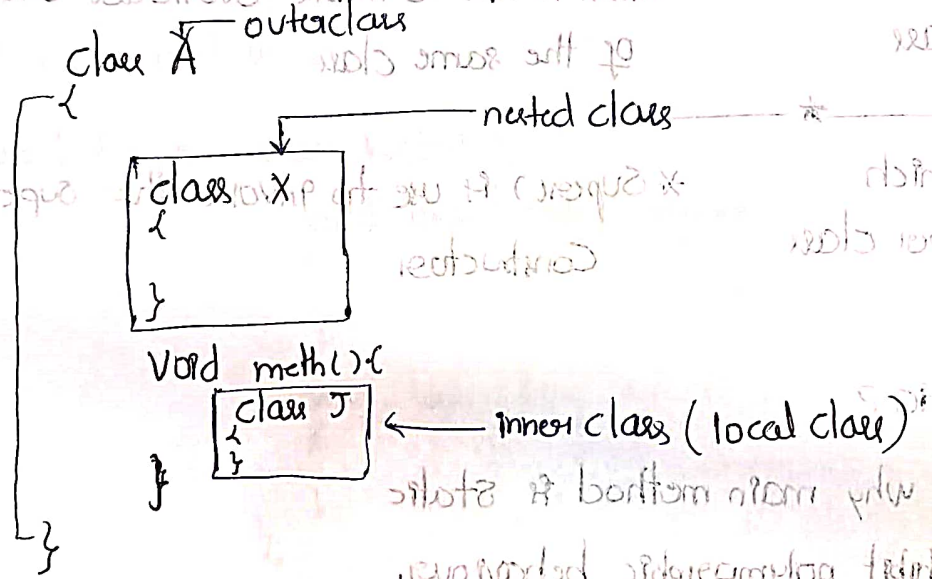
abbs	anoop	true	25	828.56
0	1	2	3	4

```
class Demo{
    psvm(String[] args){
        s.o.p(args.length); //5
    }
}
```

49. If we don't pass command line argument that what is the value of args?

It is an empty array (size of array != 0) but not null.

50. What are access modifiers apply for a class?



* Outerclass can be either default or public but nested class can be either public or private or default or protected

* Access modifiers can't be applicable for any local class members so can't applicable for inner class.

51. Explain System.out.println

```
public class Demo{
    public static void main(String[] args){
        System.out.println();
    }
}
```

System.out.println();

- → inbuilt final class java.lang
- → overloaded method printStream
- ↓ static final reference variable of type printStream

52. System.out.println(null); → Compilation error

```
class PrintStream{
println(char[] param)
{

}
println(String param)
{

}
System.out.println(null);    // Compilation error
}
```

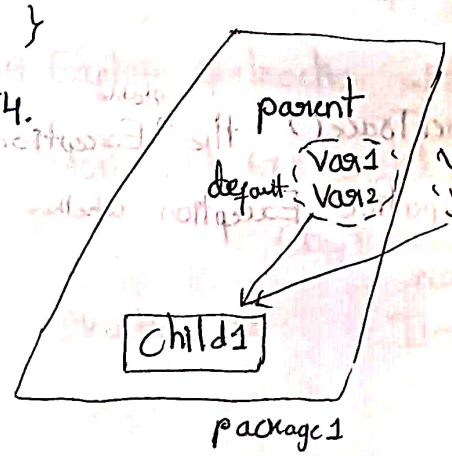
```
String s = null;
println(s);
```

⇒ println is overloaded method which accepts char[] & String as parameters when we pass null in println statement it create ambiguity during method calling so, it leads to Compilation error.

53. meth(per p) ← meth(null); // always subclass is get preferred over the superclass.

```
{
}
meth(Object o)
{
}
```

54.



55. Explain method Overriding rules?

56. What types of methods can't overridden?

→ static method

→ Final method

57. can we overload main method? ⇒ YES
(but not a good practice)

58. Can we override main method? ⇒ NO (because it is static)

59. Can we explicitly invoke main method? ⇒ YES

60. What are access modifiers that can used with Constructors?

public, private, protected, default

61. What possible exception that we can except at the time of downcasting?

ClassCastException

62. Explain polymorphism with types

Real world: ⇒ Enter button in keyword.
→ switch on button

① Collection col = new ArrayList();
HashSet();
LinkedList();
TreeSet();

col.add("");

② try{
- → IOE
- → SQLE
- → NPE

}
catch(Exception e){
e.printStackTrace();
e.getMessage();
}

When we invoke printStackTrace() the ↑ Suitable Exception get executed to the respective Exception whether IO, SQL, NP Exception.

63. Explain Encapsulation. Explain
level of Encapsulation → by making default getter & setter methods.

64. How to make your object immutable

Realtime Ex of Encapsulation:

Java Bean ~~class~~ is best Ex for Encapsulation. In MVC (model view contro architecture we use Java Bean objects to carry the data from backend to frontend.

65. Diff b/w Abstract class and Interface.

66. can we define a Constructor inside a interface ⇒ NO

purpose of Constructor is to initialization

The data members of interface are automatically public, static, final hence Constructor is not required to initialize.

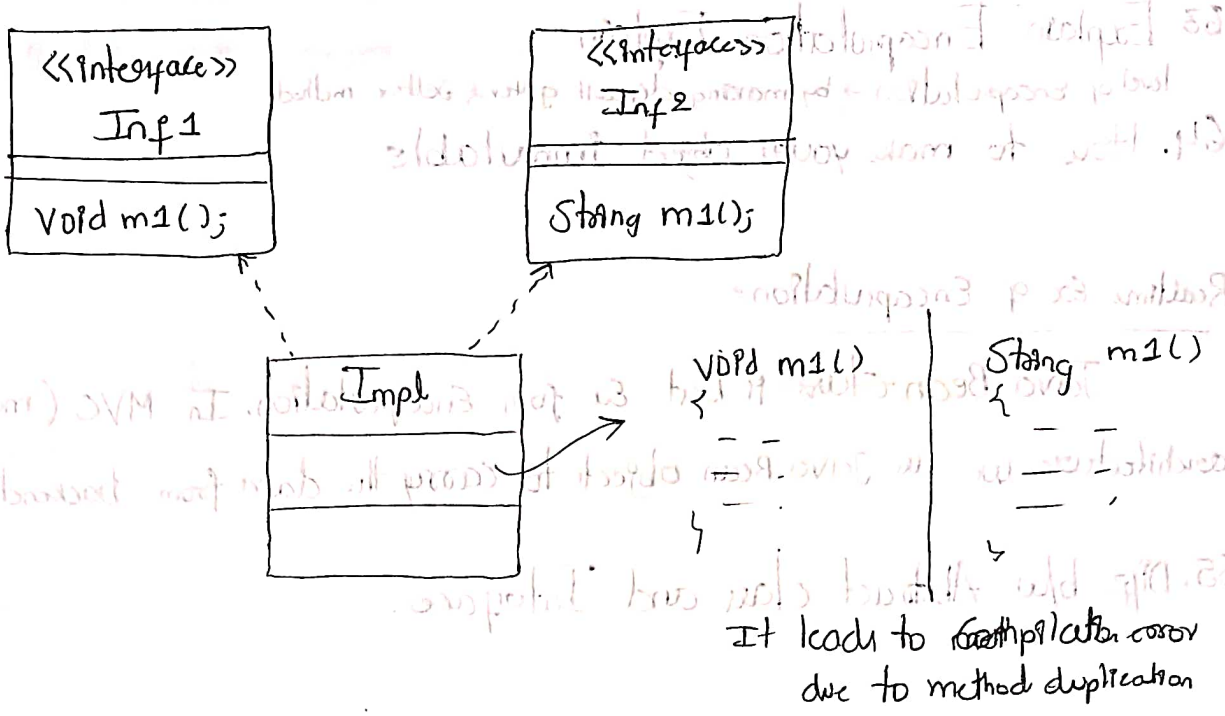
67. can an abstract class have Constructor? ⇒ YES
to initilize ~~abstract~~ instance variables.

68. How can you invoke Constructor of abstract class?
using super() of superclass ^by subclass Constructor.
constr

69. Explain Abstraction with realtime & realworld Example.

realworld Ex: watch, mouse
(with key)
(key is intermediate b/w watch & human)

Realtime ex : Iterator Ex



It leads to compilation error due to method duplication

70. What are the type of interface?

1. Regular interface
2. Marker interface
3. ~~Normal~~ Interface (SAM- single abstract method)
   Functional

Interface which doesn't ~~make~~ have any method → Marker Interface.

Ex: RandomAccess

71. Can u name marker interfaces in java?

Serializable, Cont Cloneable, RandomAccess, Remote, EventListner

72. Functional interfaces of java

Comparable, Runnable

10/10/23

Question related to Exception Handling :-

1. What is Exception?

Its a runtime interruption which stops program execution (or) a runtime error which stops program execution.

2. Can an exception occur at Compile Time?

NO, exception never occur at Compile Time

3. Diff b/w Exception & error

Exception	Error
* Exception occurs at Runtime	* Error occurs at Compile Time.
* JVM is responsible for the throwing Exception.	* Compiler is responsible for throwing an error.
* Exception occurs because of runtime interruption or sometimes because of logical mistakes.	* Error occurs because of syntatical mistakes
* Exceptions can be handled.	* Error can't be handled.

4. What are the types of Exception?

1. Unchecked Exception
2. Checked Exception

5. Name few exception that you experience in your project.

NullPointerException, ClassCastException, IO, SQL, ClassNotFoundException - - -

6. Write the Exception hierarchy

7. How do you handle an Exception? * using try & catch block

* using throws we can propagate the Exception to the caller.

8. Explain NullPointerException.

If we try to invoke either variable or method on the null reference then we get NullPointerException.

```
Pen p = null;
p.var
p.meth();        X not possible
```

9. How do you handle NullpointerException → By Conditional check

```
Pen p=null;
if(p!=null){
p.var;
p.meth();
}
```

10. Diff b/w checked & uncheckedException

11. Diff b/w throw & throws

throw is exorcised with object declaration.	throws is exorcised with method declaration or Constructor method

throws is used in the customException ⟶

12. In what situations we write only one catch & multiple catch block.

try → risky code
catch → solution/handling code
handling scenario

If the handling scenario is same (common) for multiple exceptions then we write only one catch block.

If the handling scenarios are different for different exceptions then we write seperate catch block.

13. Is it possible to handle multiple exceptions in single catch block.

YES

```
catch(IoException | SQLException | ArithE e)
{
}
```

14. What exceptions are called as Unchecked Exceptions?

Any exception category classes which extends RunTimeException is called unchecked Exception.

* After the throw the lines of code will be not executed.

System.exit(0); → forcefull way to terminate the JVM

15. In what situations finally block will not get executed?

In some situations where in JVM shut down that is abrupt terminations of JVM finally block will not executed.

or

System.exit(0);

```
Void meth()
{
    try{
    ≡
    }
    catch(){
    ≡
    }
    finally{
    ≡
    }
}
```

```
void meth(){
    try{
    ≡
    }
    catch(){
    ≡
    }
    ≡
}
```

5. How do you create, CustomException as checked or unchecked?

~~checked exception~~

If the customException class extends throwable or Exception then it belongs to checkedException.

If the customException class extends RuntimeException then it belongs to uncheckedException category.

7. Explain steps to create custom/userdefined Exception.

8. What are the imp methods are used to debug an exception

1. getMessage()
2. printStackTrace()

19. Diff b/w final, finally, finalise

final	finally	finalise
Keyword used which class, variable, method	block used in the case of exception handling	It is a method present in object class. which executes by JVM just before Garbage collection.

String

11/10/2

1. What is String?

String is a non-primitive datatype, final class, set of characters.

2. What are the different ways of String object creation?

1. Using new keyword & string constructor
2. Using double quotes

3. Explain how String is immutable?

2 object 1 object

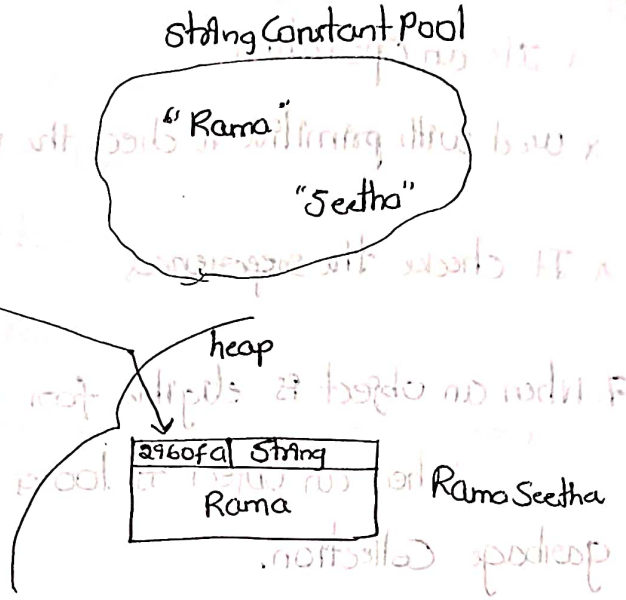
String s = new String("Rama");

(2 objects get created)

String is immutable means when we create a string object then the contents of the string object can't be changed. If we try to change the content of string object then, another new object gets created but

existing object will not get changed.

```
String s = new String("Rama");
s.concat("seetha");
S.O.P(s);  // Rama
```



4. Diff b/w String, StringBuilder, StringBuffer

String	StringBuilder	StringBuffer
String is Immutable	StringBuilder is mutable	StringBuffer is mutable
ThreadSafe (Single Threaded)	Not ThreadSafe (Multi Threaded)	Thread Safe
String is slower	StringBuilder is faster	Slower
present since JDK 1.0	JDK 1.5	present since JDK1.0
we can use concat	we can't use concat operator	we can't use concat operator
Scp applicable	Scp not applicable	scp not applicable
Two ways of object creation	only one way	only one way.

5. How do you create your own Immutable Object?

1. Define a final class
2. Declare all the data members as final & initialize at the time of declaration itself. so that the Object's state can't be modified.

6. Diff b/w == & equals()

==	equals()
* Its an Operator	* equals() method
* Used with primitive to check the value	* Used with only non-primitive
* It checks the references	* It checks the Content.

7. When an object is eligible for garbage collection?

When an object looses all its reference then it is eligible for garbage Collection.

8. Can we explicitly invoke garbage collector?

YES System.gc();

9. Name some final inbuilt Classes of Java?

String, System, & all wrapper classes.

10. How do you Convert String to primitive datatype?

By using wrapper classes methods.

primitive to String → Concat with ""

```
double primd = 9154.2;
String s = primd+"";
or
String s = Double.toString(pr
```

not a method object bcoz it doesn't parameter

11. How do you convert primitive-data into String?

There are two ways:

1. Concatenate the primitive with empty string — Ex:
2. Using static method called toString present in wrapper classes

```
String s = Integer.toString(33);
String s = Long.toString(69498166956 9);
```

primitive-data

Wrapper Classes

Every primitive datatype in java as a corresponding, non-primitive class, which is called as Wrapper Class.

Ex: int → Integer, short → Short ---etc---

There are mainly two uses of wrapper classes.

1. Converting primitive data into non-primitive using autoboxing.
2. Converting String into primitives.

* We can convert the string data into an appropriate primitive data using parse method of the wrapper class.
* All the wrapper classes except the character class as the parse method

Syntax: public static xxx parseXxx (String StringRepresentation)

String into primitive int — present in

Ex: public static int parseInt(String str) → Integer class
public static double parseDouble(String str) → Double class
public static long parseLong(String str) → Long
public static boolean parseBoolean(String str) → Boolean

int i = Integer.parseInt("25");
primitive 25 ← String 25

int i = Integer.parseInt("hello");
NumberFormatException (NFE)

```
class Demo{
    psvm(String[] args){
        S.O.P(Boolean.parseBoolean("hello")); // false
        S.O.P(Boolean.parseBoolean("1")); // false
        S.O.P(Boolean.parseBoolean("TRUE")); // true
        S.O.P(Boolean.parseBoolean("TRUE")); // true
    }
}
```

Case insensitive

Collections

1. Diff b/w Array & Collection.
2. Write Collection Hierarchy.
3. What is the super interface of Collection Hierarchy
   Collection. Interface which extends Iterable.
4. Diff b/w Collection & Collections & Map
5. Diff b/w Set & List
6. Use of Generics
   Type (datatype) Safety.
7. How do you make collection homogenous.
   By using generics.
8. Diff b/w ArrayList & LinkedList
9. Diff b/w ArrayList & Vector
10. How do you convert LinkedList to ArrayList. / How do you Convert one Collection to another Collection. By using overloaded Constructor

```
Import java.util.ArrayList;
Import java.util.LinkedList;

Class Main{
    LinkedList<String> ll = new LinkedList<String>();
    ll.add("Red"); ll.add("blue"); ll.add("orange");

    ArrayList<String> al = new ArrayList(ll);

    S.o.p(al);
}
}
```

11. Explain the data structure used in ~~ForceSet~~ ArrayList.
    ReSizeable/Growable Array.

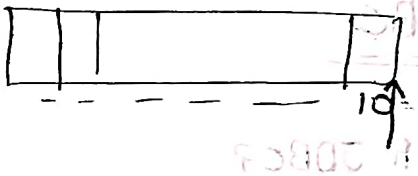
12. Diff b/w add() & set()

13. Diff b/w Iterator & for loop

14. Diff b/w for & forEach loop (enhanced for loop)

ListIterator

```
ListIterator<> itr = lst.listIterator();   // cursor point to the beginning of the list
                     lst.listIterator(11); // or list.size();  // cursor point to the end of list
while( itr.hasPrevious())
{
    itr.previous();
}
```



* When adding a data to the ArrayList we should make sure that the previous data is filled or atleast null. We can't add in the middle.

```
al.add(3, "TOM");
al.add(5, "JERRY");
```

* If the methods are not synchronized in the superclass, those can be made synchronized in the subclass while implementation.

15. Diff b/w HashSet & LinkedHashSet

16. Datastructure of TreeSet → BinaryTree

17. How do you make a TreeSet to store duplicate data.

18. How do you make TreeSet heterogenous. YES possible.

19. Diff b/w Comparable & Comparator.

20. What is Map?

21. Name the methods present in entrySet interface

getKey() and getValue()

entrySet() → method which is used to ~~we can~~ fetch both key & value from a map.

22. Diff b/w HashMap & LinkedHashMap.

23. Diff b/w HashMap & HashTable

↳ singleThreaded

↳ slow when compared to the HashMap

17/10/23

JDBC

1. What is JDBC?
2. Explain JDBC steps
3. Diff b/w JDBC API & JDBC Drivers
4. Which type of JDBC Driver we have used?

Type IV Driver

5. Can you name Driver class?

Com.mysql.cj.jdbc.Driver

6. Superinterface of all the Driver class

java.sql.Driver

7. What are different ways of loading & registering the driver?

1. using class.forName
2. creating an object of Driver class & registering the Driver with DriverManager using registerDriver method.

8. What are two important methods of DriverManager?

1. getConnection } static methods
2. registerDriver

few imp interfaces of JDBC API

Driver, Connection, preparedStatement, CallableStatement, Statement
ResultSet, Date

10. Where all the drivers are registered? ⇒ DriverManager

11. What are types of statement available in JDBC?

1. statement
2. preparedStatement
3. callableStatement

*** 12. Diff b/w statement & Callable Statement (Refer pdf notes)

13. Why callableStatements are used?

TO stored procedures

14. What is the resultSet represents?

Represents the processed data after execution of DQL queries.

15. What are the types of resultSet?

Bidirectional
1. ~~readOnly~~ resultSet
2. Forward only resultSet

16. Name few execute() available in JDBC

1. execute()
2. executeUpdate()
3. executeQuery()

17. Diff b/w execute, executeUpdate, executeQuery.

18. Name few important methods of resultSet.

next(), absolute, getXxx()

19. Name methods to set the data to placeholders.

20. Explain JDBC Architecture.

21. Can you implement Singleton design pattern for Database Connection.

```
Import java.sql.Connection;
Import java.sql.DriverManager;

public final class DBSingleton{

    private DBSingleton(){

    }
    private static Connection ONLY_ONE;

    static{
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            ONLY_ONE = DriverManager.getConnection("jdbc:mysql://localhost
                                :3306","root","root");
        }Catch(Exception e){
            e.printStackTrace();
        }
    }

    public static Connection getCon(){
        return ONLY_ONE;
    }
}
```

1. Why HashSet and HashMap doesnot maintain insertion order? (pdf notes)

Because, internally it uses Hashing algorithum, hence All hashing the order of insertion is purely based on the hash calculation.

2. Which of the collections are Homogenous?

hc of element % init-cap
↓
16
⇒ Remainder
↓
element position

File Handling: —

1. What are the types of Stream? (the sequence of data)
   1. byte Stream → the sequence of data in form byte.
   2. character stream
2. What is Serialization & deserialization?
3. What is transient keyword?

   used to not to serialize the ~~object~~.data

   which is specified as transient it can't be serialized.
4. What type of data can't be serialized?

   Static & transient data members can't be serialized.
5. What possible exceptions that we get at the time of deserialization?
   1. ClassNotFoundException
   2. FileNotFoundException
   3. IOException ~~Exception~~
   4. ClassCastException
6. What are Buffers? Why it is used?

   It improve the performance of IO operations.
7. What is SerialVersionUID? (identify)

   Its a unique id given to a class which helps the JVM to understand the matching class at the time of deserialization.

8. Can we create our own generics? YES

9. What are different ways to create an object in java?

Six ways:=

1. new keyword & the Constructor
2. Array
3. By using cloning & clone method
4. By using String literals
5. At the time of deserialization
6. Using new Instance

10. How do you create your own immutable object? (deprecated)

Multithreading:==

1. Explain Thread Lifecycle
2. Diff b/w notify() & notifyAll()
3. Explain Deadlock situation in thread
4. How do you create a thread?