OOP

1. **What is Abstraction?**
   - ⇨ Hiding internal implementation and sharing set of services is called as abstarction.
   - ⇨ We can achieve abstraction by using “Interface” and “Abstarct Class”.
   - ⇨ We can achieve 100% abstraction using “Interface” and partial by using “Abstract Class”.
   - ⇨ We can achieve security using abstraction.
   - ⇨ E.g. ATM machine, Car, Mobile, etc.
2. **What is Encapsulation?**
   - ⇨ Wrapping of data member and methods called as Encapsulation.
   - ⇨ We can achieve it by making data members “private”.
   - ⇨ POJO class is good example of encapsulation.
   - ⇨ In a class if it has every data member as a “private” then such class is called as tightly encapsulated.
   - ⇨ E.g. Engine, Gear box within Car,etc.
3. **Difference between Abstraction and Encapsulation?**

Abstraction	Encapsulation
Hiding internal implementation and sharing set of services.	Wrapping of data member and methods.
We can achieve abstraction by using “Interface” and “Abstarct Class”.	We can achieve it by making data members “private”.
It increases code.	It decreases code.
It solves problem at Design level.	It solves problem at implementation level.

4. **What is Inheritance?**
   - ⇨ Acquiring properties of Parent class is called inheritance.
   - ⇨ It is also called as IS-A relationship.
   - ⇨ It can be achieved by using “extends” keyword, by making Parent-Child relationship.
   - ⇨ It helps in reusability of code.
   - ⇨ E.g. New Car version inherits properties from old versions.
5. **What is Polymorphism?**
   - ⇨ It means one name many forms.
   - ⇨ It makes reusability simple and also makes code understanding easy.
   - ⇨ There are two types of Polymorphism :
     - Run-time Polymorphism (Dynamic Binding, Overriding)
       - Decision making at Runtime by using runtime object.
       - Used for adding additional functionality into existing one.
       - Useful only in Parent-Child Relationship.
     - Compile-time Polymorphism (Static Binding, Overloading)
   - ⇨ Best example of polymorphism is “println” method of “printstream” class.
6. **Difference between IS-A relationship & HAS-A relationship?**

IS-A relationship	HAS-A relationship
It is also known as inheritance, it is acquiring parent class properties.	It is acquiring class properties by creating instance of that class.
It is achieved by “extends” keyword.	It is achieved by “new” keyword.
Method with same name, signature but different return types can’t be created in parent-child class.	Method with same name and different return type can be created.

7. Difference between Overloading & Overriding?

Overloading	Overriding
Decision making done at compile-time, where compiling is based on reference type.	Decision making done at run-time, where JVM run is based on runtime object.
We can overload method and constructor both.	We can override method but we can't override constructor.
We can overload static, private and final method.	We can't override static, private and final method.
E.g. "println" method of Printstream class.	E.g. 'equals' method of object class and 'equals' method of String class.

8. What is Static?

- ⇨ Static is keyword.
- ⇨ We can apply static keyword with variables, methods, blocks and classes.
- ⇨ The static variable gets memory only once in the class area at the time of class loading. It can be used to refer to the common properties of all objects, for example, the company name of employees, college name of students, etc.
- ⇨ A static method belongs to the class rather than the object of a class. It can be invoked without the need for creating an instance of a class. It can access static data member and can change the value of it. It can be used to setup database connection.
- ⇨ Static block is used to initialize the static data member. It is executed before the main method at the time of class loading.

9. What is non-static block?

- ⇨ It is used for non-initializing content.
- ⇨ Before calling constructor non-static block is executed.

10. What is constructor?

- ⇨ A constructor in Java is a special method that is called when an object of a class is created.
- ⇨ It is used to initialise object as well as non-static variables.
- ⇨ Constructors can also take parameters to initialize data members.

11. What is Object?

- ⇨ An object is an instance of a class.
- ⇨ Objects have states and behaviours. Example: A dog has states - color, name, breed as well as behaviours – eat, bark, smell.
- ⇨ Ways we can create an object :
    - By using new Operator :
      Test t = new Test();
    - By using newInstance() :(Reflection Mechanism)
      Test t=(Test)Class.forName("Test").newInstance();
    - By using Clone() :
      Test t1 = new Test();
      Test t2 = (Test)t1.clone();
    - By using Factory methods :
      Runtime r = Runtime.getRuntime();
      DateFormat df = DateFormat.getInstance();
    - By using Deserialization :
      FileInputStream fis = new FileInputStream("abc.ser");

```
ObjectInputStream ois = new ObjectInputStream(fis);
Test t = (Test)ois.readObject();
```

EXCEPTION HANDLING

12. What is Exception?

- ⇨ In our application, there are chances of abnormal condition, normal flow of program get disturbed, such situation is called as an exception.
- ⇨ All types of exceptions only occurs at runtime.
- ⇨ So avoiding such abnormal termination we need to handle exception.
- ⇨ Exception will occur because of wrong user input.

13. What is Exception Handling?

- ⇨ To avoid occurring exception we need to provide alternative way to execute rest of program, this is called as exception handling.
- ⇨ Keywords to handle exception: try, catch, finally, throw, throws.

14. Difference between Exception and Error?

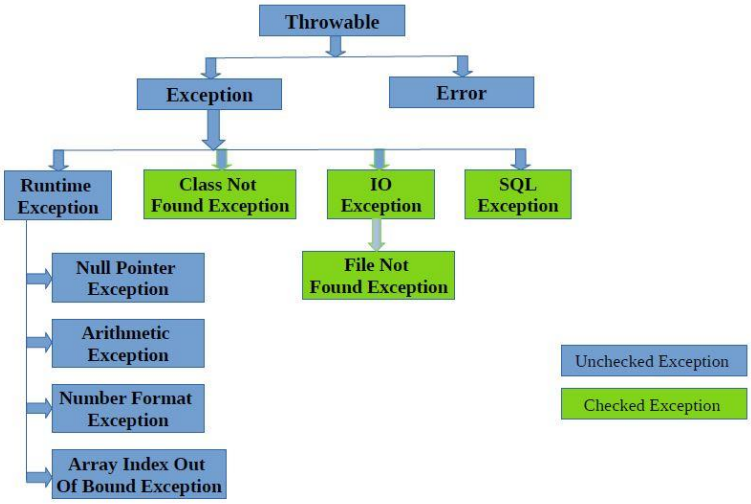
Exception	Error
Classified as checked and unchecked type.	Classified as an unchecked type.
It belongs to java.lang.Exception.	It belongs to java.lang.error.
We need to handle the exceptions.	We need to solve the errors.
It can occur at run-time & compile-time both.	It doesn't occur at compile-time, only occurs at run-time.
E.g. NullPointerException , SqlException, etc.	E.g. OutOfMemoryError , AbstractMethodError, etc.

15. Default Exception handling in Java?

- ⇨ Step-1: Problem Analysed.
- ⇨ Step-2: Problem Finding.
- ⇨ Step-3: Create Object of an Exception found.

  E.g. ArithmeticException e=new ArithmeticException ( );
- ⇨ Step-4: Throw e ;
- ⇨ Step-5: JVM will catch.
- ⇨ Step-6: JVM will display Exception message.

16. Explain Exception hierarchy?



17. Difference between Checked Exception & Unchecked Exception?

Checked Exception	Unchecked Exception
Exceptions that compiler forces user to write handling code before compilation, it is called checked exception.	Exceptions that compiler doesn't forces user to write handling code are called checked exception.
Compulsory need to handle them.	Not compulsion to handle them.
It increases code.	It reduces code.
E.g. IOexception, SqlException, FileNotFoundException, etc.	All Runtime Exception as well as all Errors are example of unchecked exception.

18. What are methods in Exception class?

- ⇨ public String getMessage()
  Returns a detailed message about the exception that has occurred.
- ⇨ public Throwable getCause()
  Returns the cause of the exception.
- ⇨ public String toString()
  Returns the name of the class concatenated with the result of getMessage().
- ⇨ public void printStackTrace()
  Prints the result of toString() along with the stack trace, the error output stream.
- ⇨ public StackTraceElement [] getStackTrace()
  Returns an array containing each element on the stack trace.
- ⇨ public Throwable fillInStackTrace()
  Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

19. What is use of finally block?

- ⇨ "finally" block code always be executed either there is problem inside "try" block or there is no problem inside "try" block. In both situation, "finally" block code will be executed.
- ⇨ Resources which are open inside "try" block, those resources should be closed inside "finally" block. E.g. If database connection open then it should be closed inside "finally" block.
- ⇨ Syntax :

```
try {                          try {
    }                              }
finally {                      catch ( ) {
        }                                      }
                               finally {
                                       }
```

- ⇨ We cannot write two or more "finally" block for one "try" block, only one "finally" block can be written with "try" block.
- ⇨ We cannot write "finally" block before "catch" block, we can write it only after "catch" block like try-catch-finally.

20. How to stop executing finally block?

- ⇨ Before finally block get executed, System.exit(0); line get executed.
- ⇨ Before finally block get executed, unending loop get executed.

21. What is final, finally, finalized?

- ⇨ The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be used for variables, methods, class.

- Final variable once assigned can't be changed after.
- Final method can't be rewritten, can't be inherited.
- Final class can't be accessed by creating child of it.

⇨ The finally is a block that always be executed either there is exception occur inside "try" block or there is no exception occur inside "try" block. In both situation, "finally" block code will be executed.

⇨ The finalize() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object, it is used to perform clean-up activity.

22. What is throws keyword?

⇨ By using "throws" keyword we can give a chance to caller method to handle the exception.

⇨ The "throws" keyword is used for propagating the exception.

⇨ Whenever unchecked exception will occur, it will automatically propagated.

⇨ Whenever checked exception will occur, we need to write "throws" keyword for propagating the exception.

⇨ e.g. public void m1( ) throws IO Exception, SQL Exception
{ }

23. What s throw keyword?

⇨ The "throw" keyword is used for occurring our own exception.

⇨ To occur custom exception we use "throw" keyword.

⇨ Syntax :

```
public void m1( )
{
Arithmetic Exception e=new Arithmetic Exception( ) ;
throw e;}
    OR
public void m2( )
{
throw new Arithmetic Exception( ) ;
}
```

24. Difference between throw & throws?

throw	throws
It is used to occur custom exception.	It is used to give a chance to caller method to handle the exception.
This keyword is used while creating new instance of exception class.	This keyword is used with method name to declare the exception that need to propagate.
Can be used only with single exception at a time.	Can be used with multiple exceptions at a time.
You cannot throw multiple exceptions.	You can declare multiple exceptions

25. Why to create Customise Exception?

⇨ Customise Exception tells user user-friendly message, as user can't understand programming language code for exception.

26. Tell me Customise Exception for any Checked Exception?

- ⇨ First create user-define class for customise exception, extend it to corresponding checked exception class, and also create parameterised constructor that accepts string input and forward that variable using super keyword.
- ⇨ Now use throw keyword and create new instance of custom class you have created where you want to handle checked exception in your application.

27. Can we create Customise Exception for Checked Exception?

- ⇨ Yes we can but not necessary, because customise exception used for telling user user-friendly message as user doesn't know programming language.
- ⇨ But programmer is aware of programming codes, so programmer doesn't require customise exception.

28. New Exception related feature in JDK 1.7 version?

- ⇨ Try with Resource.

29. Which type of statements can be written in try with resources?

- ⇨ We can write those classes/interfaces which have implemented/extended Autoclosable interface only.
- ⇨ For user-defined class/interface, we need to implement/extend Autoclosable interface explicitly. So that we can write that class in try with resources context.

30. Difference between ClassCasteException and NoClassDefFound Error?

ClassCasteException	NoClassDefFound Error
It occurs when we are trying to type cast parent object to Child.	It occurs when it is unable to find required .class file at runtime.
It can be avoided by correcting type while type casting, by changing in code.	It can be avoided by providing required .class file.
It is child class of Runtime Exception.	It is child class Error.

31. Different Scenarios in Exception Handling?

- ⇨ > Once problem will occur inside "try" block, then remaining line of code of "try" block will not be executed.
  - ➢ If there is no problem inside "try" block, then "catch" block will be skipped.

  In "catch" block we can write same class of exception and its parent classes only.
  - ➢ We can write two or more "catch" for one "try" block, but we can write child classes of exception in first catch blocks and then parent class exception, we cannot write first parent exception before child exception.
- ⇨ > If "try" with "finally" block and "try" have return statement, then before control return back to caller first of all "finally" block code will be executed.
  - ➢ If "try" with "finally" block, "try" have return statement and if "finally" block change return statement value, then it will be changed only for "finally" block, it will not be changed for caller.
  - ➢ If "try" and "finally" both have return statement, then "finally" block statement will be executed.
- ⇨ "throws" with method calling, in case of unchecked exception classes :
  - ➢ For unchecked exceptions, it is not needed to write explicitly throws exception, the exception propagation is done implicitly to the caller.
  - ➢ In case of checked exception classes, needed to write explicitly throws exception for propagating exception to the caller.
- ⇨ "throws" keyword with method Overriding:
  - ➢ (In case of unchecked exceptions) If Parent class method throws unchecked exception or doesn't throws any exception then at the time of overriding no need to

write throws keyword, if we want we can write ay of unchecked exception class but not checked exception class.

- (In case of checked exceptions) If Parent class method throws checked exception then at the time of overriding no need to write throws keyword, if we want then we can write same exception class or their child class but not parent class.

⇨ "throws" keyword with Super Class constructor and Sub Class constructor:

- (For unchecked exception) If Parent class constructor doesn't throws any exception or "throws" unchecked exception then child class constructor no need to write "throws keyword, if we want then any of exception class they can write.
- (For checked exception) If parent class constructor throws exception then at a time of inheritance child class constructor must write "throws" keyword with same exception class or their parent class.

32. How do you handled Exception in your project?

⇨ We created separate package for Exception classes needed in project where handling code was written, then at a time of exception handling scenario in project used throws keyword.

Collection

1. What is Collection?

⇨ The Collection is a framework that provides an architecture to store and manipulate the group of objects.

⇨ It provides operations that you perform on a data such as searching, sorting, insertion and deletion on the group of objects.

⇨ Collection represents a single unit of objects as a group.

2. Why do we use Collection?

⇨ (Explain Difference between Array & Collection)

Array	Collection
It is Fixed in Size.	It is Growable in nature.
It can hold only Homogeneous Data Elements.	It can hold both Homogeneous and Heterogeneous Elements.
With Respect to Memory Arrays are Not Recommended to Use.	With Respect to Memory Collections are Recommended to Use.

3. What is Difference between Collection & Collections?

Collection	Collections
It is an interface.	It is class.
It can be used to Represent a Group of Individual Objects as a Single Entity.	It is used to sort and synchronize the collection elements.
It provides the methods that can be used for data structure.	It provides the methods which can be used for various operations on a collection.

4. Which Collection object you have used in your Project?

⇨ List - ArrayList for getting lists of object from Database.

⇨ Set – HashSet for mapping POJOs using hibernate. (One-to-Many & Many-to-One)

5. How Arraylist works?

- ⇨ When we create object of Arraylist, it create Arraylist instance with default capacity 10.
- ⇨ Arraylist capacity increases with formula – New Capacity = ((3/2) x Old Capacity)+1
- ⇨ When Arraylist increments with new capacity then data from old Arraylist is copied into new instance and old instance is destroyed.
- ⇨ When we add or delete data into the Arraylist then multiple data shift operations are performed.
- ⇨ Arraylist follows Indexing.

6. Difference between Arraylist & Vector?

Arraylist	Vector
It is not Synchronised.	It is Synchronised.
It is not Thread Safe	It is Thread Safe
Its Default Capacity :- 10 and It increases after by (Old+Old*(3/2))+1	Its Default Capacity :- 10 and It increases after by Old+Old*100%
Performance is high.	Performance is low.
Enumeration cannot be used.	Enumeration can be used.

7. How we synchronize Arraylist?

- ⇨ By Default ArrayList Object is Non- Synchronized but we can get Synchronized Version ArrayList Object by using the following Method of Collections Class.
- ⇨ public static List synchronizedList(List l)
- ⇨ E.g. ArrayList al = new ArrayList ();
  List l = Collections.synchronizedList(al);

8. Explain all Constructors of Arraylist?

- ⇨ ArrayList l = new ArrayList();
  It creates an Empty ArrayList Object with Default Initial Capacity 10.
- ⇨ ArrayList l = new ArrayList(intinitialCapacity);
  It creates an Empty ArrayList Object with specified Initial Capacity.
- ⇨ ArrayList l = new ArrayList(Collection c);
  It creates an equivalent ArrayList Object for the given Collection Object.

9. Why Arraylist is fast for retrieval operation?

- ⇨ It implements RandomAccess Interface, hence Arraylist is fast for retrieval operation.

10. Why set doesn't allow duplicates?

- ⇨ Set internally uses HashMap.
- ⇨ HashMap object is created in every Set implemented class.
- ⇨ Here HashMap stores key as all inserted elements and value as a dummy object created with new keyword.
- ⇨ As HashMap doesn't accept duplicate keys, so set don't allow duplicate values.

11. How add () of set works?

- ⇨ Its return type is Boolean, it returns true or false value.
- ⇨ It uses equals ( ) method of object class.
- ⇨ It implements HasMap internally.

12. How Linkedlist works? (Why insertion & deletion is fast in Linkedlist?)

- ⇨ When we create an object of Linkedlist and add an element to it.
- ⇨ It stores element as a node in which previous & next node address is also stored.
- ⇨ Node format = ||prev. node addr.| (value) |next node addr.||
- ⇨ Due to previous & next node address is stored, hence while updation or insertion & deletion operation data shift operation need not to perform and it makes Linkedlist fast.

13. Define Linkedlist?

- ⇨ It is one of implemented class of List interface in collection framework.
- ⇨ It allows duplicate values, Insertion order is preserved & indexing is maintained.
- ⇨ It implements Clonable, Serializable interfaces.
- ⇨ It follows doubly linked list structure.
- ⇨ It is mostly preferable for insertion & deletion operations.

14. What is Map?

- ⇨ Map is used for store different object in the pair of "key" and "value".
- ⇨ In map, "key" should be unique.
- ⇨ Insertion order will not be maintained in Map.

15. Difference between Hashmap & Hashtable?

HashMap	Hashtable
It is not Synchronised.	It is Synchronised.
It allows multiple threads at a time.	It allows single thread at a time.
It is not thread safe.	It is thread safe.
Null key (once) & Null value is allowed.	Null key & Null value is not allowed.
Its performance is fast.	Its performance is slow.

16. How HashMap works?

- ⇨ When we create HashMap object, HashMap instance as per default capacity 16 buckets is created.
- ⇨ When we perform add (put ( )) operation, it accepts data in key & value format.
- ⇨ Internally hashing technique is used, that generates hashcode for key and also calculate index to find bucket location for inserting data in HashMap instance.
- ⇨ It will store element at that location as a node format.
  ||previous node address| (Key) | (Value) |next node address||
- ⇨ Now when we perform retrieval (get ( )) operation, it asks for key.
- ⇨ Again hashing technique is used and bucket location is identified, then equals ( ) method is used to compare key content and if it returns true then value is retrieved.

17. What is Hash Collision?

- ⇨ In HashMap, if two keys have same hashcodes then such situation is called as hash collision.
- ⇨ In such case, while adding data, doubly Linkedlist is created to insert data.
  ||prev. node addr.| (Key1) | (Value1) |next node addr.|| → || | (Key2) | (Value2) | ||
- ⇨ And retrieval operation is performed using equals ( ) method.

18. What happens when we put same keys in Map?

- ⇨ If we add a key-value pair where the key exists already, put method replaces the existing value of the key with the new value.

19. What is Contract between equals () & hashcode ()?

- ⇨ If equals ( ) returns true, then objects must have same hashcodes.
- ⇨ If equals ( ) returns false, then objects may or may not have same hashcodes.
- ⇨ If hashcodes of objects are same, then we can't conclude output of equals ( ), it may be true or may be false.
- ⇨ If hashcodes of objects are different, then output of equals ( ) must be false.

20. Difference between Hashmap & Synchronised (or Concurrent) Hashmap?

HashMap	Synchronised or Concurrent HashMap
It is non-Synchronized in nature.	It is Synchronized in nature.
It is not Thread-safe.	It is thread-safe.
Performance is high.	Performance is low.

It can throw ConcurrentModificationException.	It doesn't throw ConcurrentModificationException

21. Difference between Comparable & Comparator?

Comparable	Comparator
This interface is from java.lang package.	This interface is from java.util package.
It is used for Default sorting.	It is used for Custom sorting.
It has only one method i.e. compareTo.	It has two methods i.e. compare & equals.
Programmer decides how sorting is to be done.	User decides how sorting is to be done.

22. What is Identity Hashmap?

- ⇨ In IdentityHashMap JVM will Use == Operator to Identify Duplicate Keys, which is meant for Reference Comparison.
- ⇨ e.g. Integer i=new Integer (5);
  Integer i1=new Integer (5);
  Map m=new IdentityHashMap();
  m.put(i, "java");
  m.put(i1, "cjc");
  System.out.println(m); => {5=java, 5=cjc}

23. What is fail safe & fail fast iterator?

- ⇨ Using iterators we can traverse over the collections objects. The iterators can be either fail-safe or fail-fast.
- ⇨ Fail-safe iterator means they will not throw any exception even if the collection is modified while iterating over it.
- ⇨ Whereas Fail-fast iterator throw an exception (ConcurrentModificationException) if the collection is modified while iterating over it.
- ⇨ Fail-Fast Iterators internal working:
  - Every fail fast collection has a modCount field, to represent how many times the collection has changed/modified.
  - So at every modification of this collection we increment the modCount value. For example the modCount is incremented in below cases:
    1. When one or more elements are removed.
    2. When one or more elements are added.
    3. When the collection is replaced with other collection.
    4. When the collection is sorted.
  - So everytime there is some change in the collection structure, the mod count is incremented.
  - Now the iterator stores the modCount value in the initialization as below:
    int expectedModCount = modCount;
  - Now while the iteration is going on, expectedModCount will have old value of modCount. If there is any change made in the collection, the modCount will change and then an exception is thrown
- ⇨ Unlike the fail-fast iterators, fail-safe iterators traverse over the clone of the collection. So even if the original collection gets structurally modified, no exception will be thrown.

24. What is WeakHashmap?

- ⇨ In Case of WeakHashMap if an Object doesn't contain any References then it is Always Eligible for GC Even though it is associated with WeakHashMap.
- ⇨ Garbage Collector Dominates WeakHashMap.
- ⇨ Both null values and null keys are supported in WeakHashMap.
- ⇨ It is not synchronised.

Hibernate

1. **What is Hibernate & ORM tool?**

(Hibernate)

- ⇨ It is Java Frame work that simplifies development of Java Application to interaction with database.
- ⇨ It creates SQL (Structured Query Language) queries at runtime according to database.
- ⇨ It provide automatic table creation feature, relationship mapping like IS-A relationship (Default, Single Table, Joint, table per Class) & HAS-A relationship (one to one, one to many, many to one, many to many).
- ⇨ It provide primary key auto increment feature.
- ⇨ It converts all checked exception into uncheck exception.
- ⇨ It provides cache mechanism.
- ⇨ It provide HQL (Hibernate Query Language) query and Criteria API.

(ORM)

- ⇨ It is a programming technique that maps the object to the data stored in the database.
- ⇨ It overcomes the mismatch between OOP language & database.
- ⇨ It reduces developer's efforts, time and cost.

2. **What is disadvantage of hibernate?**
   - ⇨ Can't perform multiple insert operations.
   - ⇨ Debugging is difficult as compare to JDBC.
   - ⇨ Contain lots of boiler plate code.
   - ⇨ Can't be used for small type of applications.
   - ⇨ Slow execution as it perform SQL queries at runtime.
3. **Difference between get( ) & load( )?**

get ( )	load ( )
Eager Loading	Lazy Loading
If value is absent in database then it returns null.	If value is absent in database then hibernate exception (ObjectNotFoundException) occurs.
It always hit database.	It may or may not be hit to database.

4. **What is Session Factory?**
   - ⇨ SessionFactory is an interface.
   - ⇨ It contain all DB related property details (pulled from either hibernate.cfg.xml file or hibernate.properties file)
   - ⇨ SessionFactory is a factory for Session objects. (allow to create any number of session objects developer wants)
   - ⇨ We can create one SessionFactory per database in any application.
   - ⇨ It is usually created during application start up.

- ⇨ It holds 2nd level cache data.

5. What are methods present in Session?

(Session)

- ⇨ It wraps the JDBC connection.
- ⇨ It is factory of Transaction, Query and Criteria.
- ⇨ The session object provides an interface between the application and data stored in the database.
- ⇨ It holds a first-level cache (mandatory) of data.
- ⇨ The org.hibernate.Session interface provides methods to insert, update and delete the object.

(Session Method)

- ⇨ beginTransaction()
- ⇨ save()
- ⇨ update()
- ⇨ persist()
- ⇨ delete()
- ⇨ saveOrUpdate()
- ⇨ createQuery()
    - Create a new instance of Query for the given HQL query string.
- ⇨ createSQLQuery()
    - Create a new instance of SQLQuery for the given SQL query string.
- ⇨ merge()

6. Difference between save( ) & persist( )?

Save ( )	Persist ( )
Its return type is Serializable object.	Its return type is void.
It can save object within transaction boundaries and outside boundaries.	It can only save object within the transaction boundaries.
It is only supported by Hibernate.	It is supported by Hibernate and also by JPA (Java Persistence API).
It will create a new row in the table for detached object.	It will throw persistence exception for detached object.

7. What are the states of an object? Explain them?

- ⇨ In Hibernate following are the states of an object: -
- ⇨ Transient state
    - The transient state is the initial state of an object.
    - Once we create an instance of POJO class, then the object entered in the transient state.
    - Here, an object is not associated with the Session. So, the transient state is not related to any database. Hence, modifications in the data don't affect any changes in the database.
    - The transient objects exist in the heap memory. They are independent of Hibernate.
- ⇨ Persistent state
    - When object associates with the Session, it entered in the persistent state.
    - Hence, we can say that an object is in the persistence state when we use save( ), persist( ), update( ), saveOrUpdate( ), lock( ), merge( ).

- Here, each object represents the row of the database table. So, modifications in the data make changes in the database.

⇨ Detached state

- Once we either close the session or clear its cache, then the object entered into the detached state.
- As an object is no more associated with the Session, modifications in the data don't affect any changes in the database.
- However, the detached object still has a representation in the database.
- If we want to persist the changes made to a detached object, it is required to reattach the application to a valid Hibernate session.
- To associate the detached object with the new hibernate session, use any of these methods - load(), merge(), refresh(), update() or save() on a new session with the reference of the detached object.

8. What is hibernate transaction management?

⇨ There are some methods we can use within transaction boundaries like update & delete method.

⇨ Also Commit & Rollback need transaction's object always.

⇨ Commit = to save object into database after save method we need to commit that.

⇨ Rollback = checked to unchecked exception but hibernate does that internally.

9. What is cache? What is 1st level cache?

⇨ 1st level cache is associated with session, it is always enabled.

⇨ When query is fired to get data from database, then that data is stored in local session for that session object.

⇨ Then next time if we try to get same data for same session object then that time it doesn't hit database it get data from local session.

⇨ For every session object newly query get fired.

10. What is 2nd level cache?

⇨ It is associated with session factory, it need to be enabled.

⇨ It is available globally for all sessions.

⇨ It stores data from database to 1st level cache then to 2nd level cache also, then if we try to get same record then first it see within 1st level cache if data present or not, if not then it goes to 2nd level cache, if there also data not found then it goes to fire query from database.

⇨ For another session object it has its own local session (1st level cache) but 2nd level cache is same as before because it acts globally.

11. How to remove particular object from cache?

⇨ Session has Evict( ) method used for removing particular cache.

12. How to clean cache?

⇨ Session has Clear( ) method to clear all cache.

13. Which version of Hibernate, Spring, Springboot you have used?

⇨ Hibernate-4, Spring-4, Springboot-2.

14. In One-to-Many & Many-to-One, how many tables are created by-default and if mapped by is used?

⇨ By-default it will create 3 tables.

⇨ For mapped by it will create 2 tables.

15. In Many-to-Many, how many tables are created by-default and if mapped by is used?

⇨ By-default it will create 4 tables.

- ⇨ For mapped by it will create 3 tables.

16. **What is Dirty Checking?**
    - ⇨ If we get record & we set again then it is updated without calling update method, its because of dirty checking.
    - ⇨ This can be avoided by using @Immutable annotation.
17. **What is process for Automatic ID generation from any random number?**
    - ⇨ `@SequenceGenerator(name = "mySeqGen", sequenceName = "mySeq", initialValue = 500, allocationSize=1)`
    - ⇨ `@GeneratedValue(generator = "mySeqGen")`
18. **How to show SQL queries at run time?**
    - ⇨ While specifying hibernate properties, add Show_SQL property as a true, e.g. <property name="show_sql">true</property>
19. **What does hbm2ddl does?**
    - ⇨ It validates number of column.
20. **How to disable 1st level cache?**
    - ⇨ We can't disable cache but we can clear all cache using clear( ) method of session, then it works as disabled.
21. **How to enable 2nd level cache?**
    - ⇨ Add 3 rd party jar files. (ehcache)
    - ⇨ Add Annotation below to Entity class. @Cache(usage=CacheConcurrencyStrategy.READ_WRITE)
    - ⇨ Include two extra lines in settings in 'HibernateUtil.class' settings.put(Environment.USE_SECOND_LEVEL_CACHE, "true"); settings.put(Environment.CACHE_REGION_FACTORY, "org.hibernate.cache.ehcache.EhCach-eRegionFactory");
22. **How you have used Hibernate (XML, Annotations, Java Based)?**
    - ⇨ (Always Answer as) Annotation Based.
23. **Named Queries-Definition, syntax, advantages?**
    - ⇨ Instead of writing multiple case, HQL query or Native query we can write in a single place which is entity class.
    - ⇨ Because of that HQL query and Native query will not be scattered.
    - ⇨ It is easy to reuse and maintain.
    - ⇨ There are Four Annotations for Named query :

    1. @NamedQuery.
    2. @NamedQueries.
    3. @NamedNativeQuery.
    4. @NamedNativeQueries.

24. **Native Queries-Definition, syntax, advantages?**
    - ⇨ You can use native SQL to express database queries if you want to utilize database-specific features.
    - ⇨ Hibernate 3.x allows you to specify handwritten SQL, including stored procedures, for all create, update, delete, and load operations.
    - ⇨ Your application will create a native SQL query from the session with the createSQLQuery() method.
    - ⇨ You need to pass a string containing the SQL query to the createSQLQuery() method.

Spring Framework

1. **Why do we use spring?**
   - ⇨ It is light weight & open source.
   - ⇨ It is complete end (applicable for all layers in application) & modular framework (applicable for particular layer in application).
   - ⇨ We can achieve loose coupling.
   - ⇨ Non-invasive framework (doesn't force to extend or implement any base class or interface).
   - ⇨ We can develop easy to test kind of applications.
2. **Which server Spring has by-default?**
   - ⇨ Spring doesn't have server, we need to add it externally.
3. **List out the modules of spring?**
   - ⇨ There are total 6 modules in spring-
     - Spring Core
     - Spring Context/J2EE
     - Spring DAO/JDBC
     - Spring ORM
     - Spring AOP
     - Spring Web MVC
4. **What is IOC in spring?**
   - ⇨ IOC stands for Inversion of Control.
   - ⇨ It is heart of spring framework.
   - ⇨ It converts object from tight coupling to loose coupling which is achieved by dependency injection.
     (Tight Coupling -> Change in one class forces to change in other (dependent) class
     Loose Coupling -> Change in one class doesn't forces to change in other (dependent) class)
5. **What are the types of Dependency?**
   - ⇨ Primitive, Secondary and Collection.
6. **What is Dependency and what is Dependency Injection?**
   - ⇨ Data member or properties are dependency and assigning values to those data member is called as dependency injection.
7. **What are the types of dependency injection (DI)?**
   - ⇨ Setter based DI & Constructor based DI
8. **Difference between Setter based and Constructor based?**

Setter Based DI	Constructor Based DI
It allows partial injection.	It does not allow partial injection.
It overrides constructor based DI.	It doesn't override setter based DI.
It is mutable.	It is immutable.
It increases line of code.	It decreases line of code.

9. **Inject Secondary reference injection bean file?**
   - ⇨
```
<bean id="e" class="com.demo.beans.Engine">
        <property name="engineNumber" value="A123GAAG" />
        <property name="modelYear" value="2018" />
</bean>
```

```
<bean id="c" class="com.demo.beans.Car">
        <!-- passing reference of Engine class -->
        <property name="engine" ref="e" />
        <property name="carName" value="Honda" />
</bean>
```

10. Inject Primary and Secondary array injection bean file?

⇨ public class Engine {
    private String modelYear; // primitive string
    // generate getters and setters.
}

⇨ public class Car {
    private String[] carNames; // primitive string array
    private Engine[] engines; // secondary string array
    // generate setters.
}

⇨ In bean.xml file,

```
<bean id="c" class="com.demo.beans.Car">
        <property name="carNames">
                <list>
                        <value>Hindustan Moters</value>
                        <value>Tata Moters</value>
                        <value>Ashoka Leyland</value>
                </list>
        </property>
        <property name="engines">
                <list>
                        <ref bean="e1" />
                        <ref bean="e2" />
                        <ref bean="e3" />
                </list>
        </property>
</bean>
```

11. How to inject values using List, Set and Map?

⇨ <bean id = "javaCollection" class = "com.example.JavaCollection">

⇨
```
<property name = "addressList">
    <list>
      <value>India</value>
      <value>Maharashtra</value>
      <value>Pune</value>
    </list>
  </property>
```

⇨
```
<property name = "addressSet">
    <set>
      <value>India</value>
      <value>Gujrat</value>
      <value>Surat</value>
```

```
        </set>
      </property>
```

⇨
```
<property name = "addressMap">
    <map>
      <entry key = "1" value = "India"/>
      <entry key = "2" value = "Goa"/>
      <entry key = "3" value = "Panaji"/>
    </map>
  </property>
</bean>
```

12. How to inject Vector from List hierarchy (write bean.xml)?

⇨
```
<property name="empName">
  <util:list list-class="java.util.Vector">
    <value>ABC</value>
    <value>XYZ</value>
  </util:list>
</property>
```

13. How to inject Hashset from Set hierarchy (write bean.xml)?

⇨
```
<property name="empId">
<util:set set-class="java.util.HashSet ">
   <value>SO11</value>
   <value>SO12</value>
</util:set>
</property>
```

14. How to inject Hashmap from Map hierarchy (write bean.xml)?

⇨
```
<property name="empIdName">
<util:map map-class="java.util. HashMap ">
   <entry key="WO1" value="ABC" />
   <entry key="WO2" value="GHI" />
   <entry key="WO3" value="MNO" />
</util:map>
</property>
```

15. What are the types of Container?

⇨ Bean factory (Core) Container & Application Context Container.

16. Difference between Bean Factory Container & Application Context Container?

Bean Factory	Application Context
It is core or legacy container.	It is advance container, it extends bean factory & have additional advance properties.
It is used to develop desktop based applications.	It is used to develop web based & desktop based applications.
It is lazy loading.	It is eager loading.
It doesn't support annotations.	It supports annotations.

17. How to make Application Context Lazy loading?

⇨ By using scope as Prototype, we can make application context to act as a lazy loading.

18. What are bean scopes?

⇨ There are five bean scopes.

- Singleton (same instance per IOC container)
- Prototype (any number of instances per IOC container)
- Request (Valid for Spring based applications, used for httprequest)
- Session (Valid for Spring based applications, used for httpsession)
- Global Session (Valid for Spring based applications, used for global httpsession)

19. What is autowiring?

⇨ For injecting secondary type of dependency, we use autowiring.

20. How to enable Autowiring?

⇨ Add Spring context and <context:annotation-config /> in bean configuration file.

⇨ Include 'AutowiredAnnotationBeanPostProcessor' directly in bean configuration file.
<bean class = "org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor"/>

21. What does autowire supports by-default?

⇨ By-default autowire supports byType.

22. What is front controller?

⇨ It is also known as Dispatcher Servlet.

⇨ It manages entire process.

⇨ It find appropriate class as per request.

23. What are Stereotype annotations? Explain them?

⇨ @Controller: Identifies class as a controller class & marks it as a bean.

⇨ @Service: Identifies class as a service class & tells it has business logic.

⇨ @Repository: Identifies class as a dao layer class & tells it has a database connection.

24. Can we write @Repository for business logic class and @Service for dao logic class?

⇨ Yes we can shuffle @Repository & @Service.

⇨ It is just to understand other programmers to identify business logic class and dao layer class.

⇨ But we can't replace @Controller class.

25. What are annotations used in spring?

⇨ @RequestMapping:
- It is used at class level or method level.
- It is used to map URL.

⇨ @RequestParam:
- It brings single variable from client side.
- E.g. Username, Password get from client side.

⇨ @ModalAttribute:
- It brings complete POJO class from client side.
- E.g. Registration form data from client side.

26. What is server side validations or spring validations?

⇨ Some annotations we can use as validations.

⇨ Annotations like-
- @NotEmptyValidation
- @SizeAnnotation
- @EmailAnnotation
- @NotNullAnnotation
- @Valid => Data from client side using request, then this checks data is as per validations or not.

Springboot

1. **Why to use springboot rather than spring MVC?**
   - ⇨ Springboot has embedded Tomcat server, no need to add external server.
   - ⇨ It has starter form Maven Dependency. (Web Starter, JPA Starter, etc.)
   - ⇨ It automatically configure spring application by using @Autoconfigure based on the dependencies.
   - ⇨ It has annotations that's why it skips XML configurations.
2. **Different ways to create springboot annotations?**
   - ⇨ Using Spring Initializer (https://start.spring.io/)
   - ⇨ Using Spring STS (Spring Tool Suite)
   - ⇨ Using Springboot CLI
   - ⇨ Using Spring IDE (Eclipse)
3. **How does springboot application works?**
   - ⇨ Create new project of springboot application.
   - ⇨ First specify dependency starter in pom.xml
   - ⇨ Then dependencies are added to project.
   - ⇨ Executes program from main ( ) where auto configuration done.
   - ⇨ Executes SpringApplication.run( ) that launches program.
4. **What does @SpringBootApplication do?**
   - ⇨ It marks class as a bean.
   - ⇨ It has 3 annotations working inbuilt are @Configuration , @EnableAutoConfiguration and @ComponentScan.
   - ⇨ @Configuration = It tells that class is spring bean.
   - ⇨ @EnableAutoConfiguration = It configures application as per jars added.
   - ⇨ @ComponentScan = Scans other packages in application.
5. **SpringApplication.run ( ) statement does?**
   - ⇨ It bootstraps and launches the application.
6. **How do we monitor & manages our application?**
   - ⇨ To get production ready feature we use springboot actuator.
   - ⇨ It has inbuilt HTTP endpoints.
   - ⇨ We can create our endpoints.
   - ⇨ Enable this using springboot actuator.
   - ⇨ E.g. of endpoints /bean (name of beans in application), /health (up or down of application).
7. **What is springboot profile (@profile)?**
   - ⇨ It provide ways to divide application to spare them for particular environment.
   - ⇨ We specify @Profile to tell code is written for particular environment.
   - ⇨ Environments like Development environment, Production environment, Testing environment, etc.
   - ⇨ Each environment has different configurations.
8. **What is spring data JPA?**
   - ⇨ It has 3 repository interfaces in that-
     - CrudRepository (It is parent interface, used for CRUD operations)
     - PagingAndSortingRepository (It extends CrudRepository, used for Paging & Sorting purpose)
     - JPARepository (It extends PagingAndSortingRepository, used for all above and its additional services)

9. **How to create custom queries?**
   - ⇨ Using @Query annotation.
   - ⇨ E.g. @Query("select c from City c where c.name like %?1")
     List<City> findByNameEndsWith(String chars);

10. **How did you manage session in your project?**
    - ⇨ We have used spring securities.
    - ⇨ But I haven't implemented it, my senior has implemented it.
11. **How to configure Hibernate in springboot?**
    - ⇨ Add JPA & MySQL database dependency.
    - ⇨ In application.properties, write hibernate configurations to connect database.
12. **How application.properties file added to container?**
    - ⇨ @EnableAutoConfiguration annotation is responsible for connection between database & application while configuring hibernate by adding application.properties file to container from its fixed path or specified path.

RESTfull Web Service

1. **What is web service?**
   - ⇨ It is medium or way of communicating application over the network.
   - ⇨ It is a client-server application or application component for communication.
   - ⇨ The method of communication between two devices over the network.
   - ⇨ It is a software system for the interoperable machine to machine communication.
   - ⇨ It is a collection of standards or protocols for exchanging information between two devices or application.
2. **What are the advantages of web service?**
   - ⇨ Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
   - ⇨ Reusability: One web service can be used by many client applications at the same time.
   - ⇨ Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
   - ⇨ Easy to deploy and integrate, just like web applications.
   - ⇨ Multiple service versions can be running at same time.
3. **What are the types of web service and difference between them?**
   - ⇨ SOAP (Simple object Access Protocol) & REST (Representational State transfer) are the types of web services.

SOAP	REST
It supports XML data format only.	It supports XML, HTML, Plain text, JSON, etc. data formats.
It is protocol based.	It is architecture based.
SOAP cannot use REST.	REST can use SOAP.
It has its own security.	Have to add security.
It is less preferred.	It is more preferred.
Need JAX-WS API to implement SOAP.	Need JAX-RS API to implement REST.

4. **What are the ways to implement Restful web service? Which one you have used?**

- ⇨ In JAX-RS, there are 3 types to implement Restful web service.
  - Jersey.
  - REST easy.
  - Spring with REST.
- ⇨ We have used spring with REST type.

5. By using spring with REST, how do we create Restful web service?

- ⇨ Add REST dependency in pom.xml. (spring-boot-starter-data-rest)
- ⇨ Add @RestController annotation for controller class.
- ⇨ Map the methods written in controller class.

6. What is REST? Why do you choose Restful web service?

- ⇨ REST is the acronym for REpresentational State Transfer. REST is an architectural style for developing applications that can be accessed over the network.
- ⇨ REST is a stateless client-server architecture where web services are resources and can be identified by their URIs (Uniform Resource Identifiers).
- ⇨ We choose Restful web service because-
  - It is architectural style not protocol.
  - It is fast and has no restrictions like SOAP.
  - It uses data formats like XML, HTML, Plain Text, JSON, etc.
  - It also can use SOAP.
  - It is programming language and platform (OS) independent.

7. Annotations used in Restful web service?

- ⇨ @RestController
- ⇨ @GetMapping
- ⇨ @PostMapping
- ⇨ @DeleteMapping
- ⇨ @PutMapping
- ⇨ @PathVariable
- ⇨ @RequestBody
- ⇨ @ResponseBody

8. Difference between @Controller & @RestController?

@Controller	@RestController
It is used to mark a class as Spring MVC Controller.	It is used in RESTFul web services and the equivalent of @Controller + @ResponseBody.
It returns view.	It returns data.
Added in Spring 2.5 version.	Added in Spring 4.0 version.

9. Difference between @PathVariable & @PathParam?

@PathVariable	@PathParam
It is from spring framework.	It is from JAX-RS.
It will work in spring MVC and REST	It will work in REST only.
It is annotation on a method argument to bind it to the value of a URI template variable.	It is a parameter annotation which allows you to map variable URI path fragments into your method call.

10. Have you Consumed or Produced Restful web service? Explain?

- ⇨ Consume or Produce in Restful web service means specifying which type of data to be requested (it is just telling type of data accepted or send through application)

- ⇨ E.g. While consuming data type we use syntax, consumes="application/JSON", it specify that certain service accepts only JSON data format.
- ⇨ Produce is associated with @GetMapping annotation where we mention type of data while we are sending it when that method/service is called.
- ⇨ Consume is associated with @PostMapping annotation where we mention type of data is to be accepted for saving it into the database when that method/service is called.
- ⇨ E.g. Film Producer makes (Produce) Movies & Viewers watch (Consume) those movies.

11. How to consume and produce data using xml format?

- ⇨ First use @XmlRootElement on POJO class or need to add Xml data supporting dependency.
- ⇨ Then in Controller methods specify,
  - (For Produce) produces="application/xml"
  - (For Consume) consumes="application/xml"

12. In Restful web service, how to convert JSON format data to Object format or vice versa?

- ⇨ Object to JSON format:
  - ObjectMapper mapper=new ObjectMapper( );
  - String jsonResult=mapper.writeValueAsString(object);
- ⇨ JSON to Object format:
  - ObjectMapper mapper=new ObjectMapper( );
  - Student s=mapper.readValue (jsonResult, Student.class);

13. How to manage session in Restful web service?

- ⇨ We can't manage session in Restful web service.

14. Can we manage session in Restful web service?

- ⇨ We can't manage session in Restful web service.
- ⇨ Because Restful web services are stateless means for every request it takes it as a new request.

15. How do you apply or implement security in web service?

- ⇨ By using spring security we have achieved security in web services.
- ⇨ But I have not implemented it, my senior has implemented it.

16. Difference between API & web services?

API	Web Services
It is an interface between two different applications.	It is interaction between applications over the network.
All APIs are not web services.	All web services are APIs.
API can be used for any style of communication.	Web service uses styles like REST, SOAP for communication.
It can be used by a client who understands JSON or XML.	It can be used by any client who understands XML.
API has a light-weight architecture.	Web Services does not have a light-weight architecture.

17. How to test web service?

- ⇨ Using ARC & POSTMAN.
- ⇨ Locally –
  - Create war file.
  - Deploy on local Tomcat server.

18. How do you mock the web service?

- ⇨ Mocking Web service is testing application using dummy data.

- ⇨ We haven't done it.

19. **What are HTTP methods?**
    - ⇨ GET, POST, DELETE, PUT, etc.
    - ⇨ GET - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
    - ⇨ POST - The POST method is used to submit an entity to the specified resource, often causing a change in state.
    - ⇨ PUT - The PUT method replaces all current representations of the target resource with the request payload.
    - ⇨ DELETE - The DELETE method deletes the specified resource.
    - ⇨ PATCH - The PATCH method is used to apply partial modifications to a resource.
20. **What are components in HTTP?**
    - ⇨ Request, Body, Header, etc

Design Pattern

1. **Explain with example Factory Method Design Pattern?**
    - ⇨ It is used if a java class is having private constructor and outside of that class object creation of that class is not possible.
    - ⇨ Factory method is java method, having a capability of constructing and returning its own java class object.
    - ⇨ E.g. Example for pre defined instance factory method is:
      String s1="welcome to durga software solutions"; // Existing Object
      String s2=s1.substring(3,7); // New Object i.e. it gives come
    - ⇨ E.g. (Sample Code)

```
class Test
{
        int x;
//private zero argument constructor
        private Test()
         {
            System.out.println("Test: 0-arg Con");
         }
//static factory method
        public static Test staticFactoryMethod()
         {
           Test t=new Test();
            t.x=5;
           return t;
         }
//instance factory method
        public Test instanceFactoryMethod()
        {
          Test t=new Test();
          t.x=this.x+5;
         return t;
        }
        public String toString()
```

```
        {
          return "x="+x;
        }
    }

    public class FactoryEx
    {
        public static void main(String[] args) throws Exception
        {
          Test t1=Test.staticFactoryMethod();
           System.out.println(t1); // which internally calls toString()
           Test t2=t1.instanceFactoryMethod();
          System.out.println(t2);
        } }
```

2. **Explain with example Singleton Class Design Pattern?**
    - ⇨ Creating multiple objects for java class with same data is wastage of memory.
    - ⇨ So we can use Singleton java class, which says create only one object for java class and use it for multiple times on each JVM in order to minimize the memory wastage and to increase the performance.
    - ⇨ Singleton java class is java class, which allows us to create only one object per JVM.
    - ⇨ Sometimes it's important to have only one instance for a class.
    - ⇨ E.g. Most of the JDBC driver classes are given as singleton java classes for better utilization.
    - ⇨ E.g. (Sample Code)

```
class Demo
{
        private static Demo instance;
        private Demo()
        {
          System.out.println("Demo : 0-arg Con");
        }
//static factory method
        public static Demo getInstance()
        {
// Singleton Logic
          if (instance == null)
          instance = new Demo();
          return instance;
        }
//override Object class clone()
        public Object clone()
        {
          return instance;
        }
}
```

3. **How to break Singleton Class?**
    - ⇨ To Break Singleton Using Reflection

```
import java.lang.reflect.Constructor;

public class Test {
   public static void main(String[] args) throws Exception {
      Demo s = Demo.getInstance();
      Class clazz = Demo.class;
      Constructor cons = clazz.getDeclaredConstructor();
      cons.setAccessible(true);
      Demo s2 = (Demo) cons.newInstance();
   }
}
```

- ⇨ The singleton will only work per classLoader. To break it, use multiple classLoaders.
- ⇨ One more way is to Serialize the Object store it in a file and Deserialize it to get a new Object.

4. **What is Class Loader?**
   - ⇨ The Java ClassLoader is a part of the Java Runtime Environment that dynamically loads Java classes into the Java Virtual Machine. The Java run time system does not need to know about files and file systems because of classloaders.
   - ⇨ Java classes aren't loaded into memory all at once, but when required by an application. At this point, the Java ClassLoader is called by the JRE and these ClassLoaders load classes into memory dynamically.

5. **What is real time example of Singleton Class Design Pattern?**
   - ⇨ Music Player Software, It has class contains services like Play the song, Stop the Song, Change the song, etc. Single instance of this class can be used for playing multiple songs in this application.

Serialization

1. **What is Serialization?**
   - ⇨ The process of saving (or) writing state of an object to a file is called serialization.
   - ⇨ But strictly speaking it is the process of converting an object from java supported form to either network supported form (or) file supported form.
   - ⇨ By using FileOutputStream and ObjectOutputStream classes we can achieve serialization process.
2. Have you used Serialization in your project?
   - ⇨
3. **What is real time technical example of Serialization?**
   - ⇨ Flipkart/Amazon application, all buying section contains images of products, that are stored in binary format in their database using serialization technique. When client download them from their website then these images are deserialised into client's device.
4. **How to serialize particular states of objects of class?**

- ⇨ By using Externalization process.
- ⇨ To provide Externalizable ability for any object compulsory the corresponding class should implements externalizable interface.
- ⇨ Externalizable interface is child interface of serializable interface.

Agile

1. **Do you know Agile? Have you implemented in your project?**
   - ⇨ The Agile Method and methodology is a particular approach to project management that is utilized in software development, it is iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.
   - ⇨ We have not implemented 100 % agile, we have followed agile.
2. **What is Sprint?**
   - ⇨ It is of maximum 2 weeks or minimum 2 days.
   - ⇨ It is time duration, in which decided work is completed.
3. **What is Story?**
   - ⇨ Whichever task is assigned is called as Story.
4. **Who is Scrum Master?**
   - ⇨ Manager who carry all agile process from team.
   - ⇨ In our case team manager was Scrum Master as we can't afford resource for that as we were small scale industry.
5. **Who is Product Owner?**
   - ⇨ It is person who interacts with client.
   - ⇨ Gets requirements from client and convert it into stories in document format. So that it could be assigned to developers.
6. **What is your daily routine?**
   - ⇨ On every start of week, at start time of office at 9 AM, daily stand up or scrum meeting was held approximately 15 min or can be extended to 30 min.
   - ⇨ Scrum master manages this meeting and assigns stories to developers.
   - ⇨ In this meeting, road blocks or problems occurred in stories are discussed and resolved.
7. **What is Grooming Session?**
   - ⇨ It held by Product Owner.
   - ⇨ Also upcoming stories are explained to developers.
8. **What is Retrospective Call?**
   - ⇨ It happens at Sprint end.
   - ⇨ All issues that stopped working of stories are discussed and taken care of not repeating them in next sprint.
   - ⇨ Kudos & Thanksgiving done.
9. **What is Demo Call?**
   - ⇨ It held at the end of sprint.
   - ⇨ Its call with client to show demo of story completed.
   - ⇨ Client suggestions are added in next sprint stories by product Owner.
10. **How did you gather requirements from Client?**
    - ⇨ We had resource as a product owner, for gathering requirement who also divided it into stories to assign it to developer.

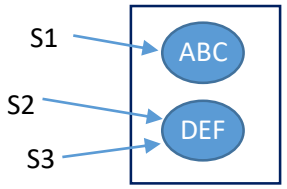
11. **How did Stories assign to you?**
    - ⇨ We used to receive mails from product owner with stories attached in document format.
    - ⇨ In big scale companies JIRA like softwares are used to handle all agile process.

GitHub

1. **Tell all the steps followed in Github in your project?**
    - ⇨ git clone url (by-default points master branch)
    - ⇨ git checkout sprint1
    - ⇨ git branch (new branch name)
    - ⇨ git checkout new branch (now it points to new branch created)
      (now make changes in code, after that follow below commands)
    - ⇨ git status
    - ⇨ git add
    - ⇨ git commit –m “commit message”
    - ⇨ git push origin “branch_name”
2. **What was Master Branch?**
    - ⇨ Branch name “develop”, that contains clean code which goes for execution.
    - ⇨ Team leader was developing new master branch for each sprint and then we clone that.

String

1. **What is String Constant Pool (SCP) concept?**
    - ⇨ When we create String object without using new operator the objects are created in SCP (String constant pool) area.
      String s1=”ABC”;
      String s2=”DEF”;
      String s3=”DEF”;
    - ⇨ When we create object in SCP area then just before object creation it is always checking previous objects.



SCP Area

    - ⇨ If the previous object is available with the same content then it won’t create new object that reference variable pointing to existing object.
    - ⇨ If the previous objects are not available then JVM will create new object.
    - ⇨ SCP area does not allow duplicate objects.
2. **When does object get created in SCP?**
    - ⇨ When String object is created with new keyword.
    - ⇨ When only String variable with content is created.
3. **Why String is immutable?**
    - ⇨ String object is immutable means its internal state remains constant after it has been entirely created. This means that once the object has been assigned to a variable, we can neither update the reference nor mutate the internal state by any means.

- ⇨ The String is widely used in Java applications to store sensitive pieces of information like usernames, passwords, connection URLs, network connections, etc. It's also used extensively by JVM class loaders while loading classes.
- ⇨ Hence securing String class is crucial regarding the security of the whole application in general.
- ⇨ Being immutable automatically makes the String thread safe since they won't be changed when accessed from multiple threads.
- ⇨ As we saw previously, String pool exists because Strings are immutable. In turn, it enhances the performance by saving heap memory.
- ⇨ E.g. Database connection properties are stored in String, so that no one can update that object to hack database data and database security can be achieved.

4. Difference between String & StringBuffer?

String	StringBuffer
String class is immutable.	StringBuffer class is mutable.
String is slow and consumes more memory when you concat too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when you cancat strings.
String class overrides the equals() method of Object class.	StringBuffer class doesn't override the equals() method of Object class.

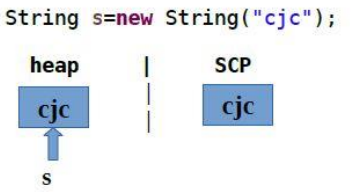
5. What are the String Scenarios related to String object creation while using Concat ( )?

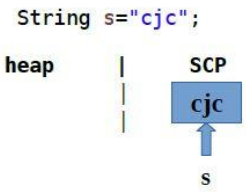
- ⇨ Concatenation of String :

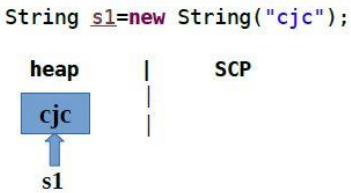
  String s="cjc";

  String s1=s.concat("class");

- ⇨ String scenarios :







6. How to use SCP objects in intern ( ) method?

- ⇨ It can be used to return string from memory, if it is created by new keyword. It creates exact copy of heap string object in string constant pool.
- ⇨ E.g. (Sample Code)

```
public class InternExample2 {
   public static void main(String[] args) {
      String s1 = "Javatpoint";
      String s2 = s1.intern();
```

```
        String s3 = new String("Javatpoint");
        String s4 = s3.intern();
        System.out.println(s1==s2); // True
        System.out.println(s1==s3); // False
        System.out.println(s1==s4); // True
        System.out.println(s2==s3); // False
        System.out.println(s2==s4); // True
        System.out.println(s3==s4); // False
    }
}
```

7. **How to create immutable class?**
    - ⇨ The instance variable of the class is final i.e. we cannot change the value of it after creating an object.
    - ⇨ The class is final so we cannot create the subclass.
    - ⇨ There is no setter methods i.e. we have no option to change the value of the instance variable.

Logical Problems

1. **Find Even-Odd Number?**

```
Public class Test1
{
 public static void main(String[ ] args)
  {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    if (n%2==0)
     {
          System.out.println(n+”is Even number.”);
     } else {
          System.out.println(n+”is Odd number.”);
           }
  }
}
```

2. **Find factorial of number?**

```
Public class Test2
{
  public static void main(String[ ] args)
   {
     Scanner sc=new Scanner(System.in);
     int n=sc.nextInt( );
     int sum=1;

     for(int i=1 ; i<=n ; i++)
      {
          sum=sum x i ;
      }
```

```
        System.out.println("Factorial of " + n + "is : "+ sum);
 }
}
```

3. Find Prime Number?

```
Public class Test
{
  public static void main(String[ ] args)
   {
     Scanner sc=new Scanner(System.in);
     int n=sc.nextInt( );
     int count=0;
     for(int i=1 ; i<=n ; i++)
      {
            if( n%i == 0)
             {
               count++;
             }
            if(count == 2)
             {
              System.out.println(n + "is prime number.");
             } else {
              System.out.println(n + "is not a prime number.");
                    }
      }

  }
}
```

4. Find Armstrong number?

E.g. 153 => $1^3+5^3+3^3 = 153$, 1634 => $1^4+6^4+3^4+4^4 = 1634$, etc.

```
Public class Test
{
  public static void main(String[ ] args)
   {
     Scanner sc=new Scanner(System.in);
     int n=sc.nextInt( );
     int m=n;
     int count=0;
    while(m>0)
     {
            m=m/10;
            count++;
     }
    int m=n;
```

```
   int sum=0, k=1;
   while(m>0)
    {
      m=m/10;
      int c=count;
      for(int i=1 ; i<=c ; i++)
      {
            k=k x m
            sum=sum + k;
      }
    }
   if(sum == n)
   {
            System.out.println(n + “is Armstrong number.”);
   } else {
              System.out.println(n + “is not a Armstrong number.”);
                    }
 }
}
```

5. Find Palindrome Number?

E.g. 121, 3443, 24542, etc.

```
Public class Test
{
  public static void main(String[ ] args)
   {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    int nrev=0, n1 = n;
    while(n>0)
    {
            nrev = nrev x 10 + (n % 10);
            n=n/10;
    }
   if(n1 == nrev)
   {
            System.out.println(n1 + “is Palindrome number.”);
   } else {
              System.out.println(n1 + “is not a Palindrome number.”);
                    }
  }
}
```

6. Find Perfect Number?

E.g. 6 => (3,2,1) => 3+2+1 = 6

```
Public class Test
{
  public static void main(String[ ] args)
```

```
  {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    int sum=0;
    for(int i=1 ; i<n ; i++)
     {
          if(n % i == 0)
           {
             sum=sum + i ;
           }
     }
     if(sum == n)
     {
          System.out.println(n + “is Perfect number.”);
   } else {
             System.out.println(n + “is not a Perfect number.”);
          }

 } }
```

7. Find Neon Number?

E.g. 9^2 =81 => 8+1 = 9

```
Public class Test
{
  public static void main(String[ ] args)
  {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    int m=n x n, sum=0;
    while(m>0)
    {
          sum = sum + m%10 ;
          m=m/10;
     }
     if(sum == n)
     {
          System.out.println(n + “is Neon number.”);
   } else {
             System.out.println(n + “is not a Neon number.”);
          }

 }
}
```

8. Find Spy Number?

E.g. 1124 => 1+1+2+4 = 1 x 1 x 2 x 4 = 8

```
Public class Test
{
  public static void main(String[ ] args)
```

```
  {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    int sum=0, mult=1;
    while(n>0)
   {
          sum = sum + n%10 ;
          mult = mult x n%10 ;
          n=n/10;
    }
    if(sum == mult)
     {
          System.out.println(n + “is Spy number.”);
      } else {
            System.out.println(n + “is not a Spy number.”);
            }

 }
}
```

9. Swap the numbers?

```
Public class Test
{
  public static void main(String[ ] args)
  {
    Scanner sc=new Scanner(System.in);
    int n1=sc.nextInt( );
    int n2=sc.nextInt( );
    int temp = n1;
    n1=n2;
    n2=temp;
  }
}
```

10. Find Automorphic number?

E.g. $\underline{5}^2 = 2\underline{5}$, $\underline{6}^2 = 3\underline{6}$, $\underline{76}^2 = 57\underline{76}$, etc.

```
Public class Test
{
  public static void main(String[ ] args)
  {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt( );
    int m = n x n, n1=n, temp=1;
    while(n1>0)
    {
          n1=n1/10 ;
          temp = temp x 10 ;
    }
```

```
    m = m % temp ;
       if(n == m)
    {
          System.out.println(n + “is Automorphic number.”);
     } else {
           System.out.println(n + “is not a Automorphic `snumber.”);
           }
 }
}
```

11. Print Fibonacci Series?

E.g. 1, 1, 2, 3, 5, 8, 13 . . .

```
Public class Test
{
 public static void main(String[ ] args)
  {
          System.out.println(“1, ”);
          int j = 1, k = 0 ;

          for(int i = 0; i<=10; i++)
          {
                  int m = j + k ;
                  System.out.println( m + “ , ”);
                  k = j ;
                  j = m ;
          }
  }
}
```

Angular

1) What is module in Angular 4?

- ⇨ Module in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application.
- ⇨ In case you are developing a website, the header, footer, left, center and the right section become part of a module.
- ⇨ To define module, we can use the NgModule.
- ⇨ Every Angular app has at least one module, the root module.

2) How to create project using NPM?

- ⇨ Step-1: Install angular cli:
  - `npm install - g @angular/cli`

Step-2: Create new project by this command:

- Choose yes for routing option and, CSS or SCSS.
- `ng new myNewApp`

Step-3: Go to your project directory:

- `cd myNewApp`

Step-4: Run server and see your application in action:

- `ng serve`

3) What is NPM?

⇨ NPM stands for Node Package Manager.

⇨ NPM is used to fetch any packages (JavaScript libraries) that an application needs for development, testing, and/or production, and may also be used to run tests and tools used in the development process.

⇨ NPM is 3rd party library or 3rd party package.

⇨ NPM is the package manager for the Node JavaScript platform. It puts modules in place so that node can find them, and manages dependency conflicts intelligently.

⇨ Most commonly, it is used to publish, discover, install, and develop node programs.

4) How to create modules, components, services, interfaces, classes in Angular 4 using NPM?

=> *for Module:

- ng g module newModule

*For Components:

- ng g component new-module/new-component

*for Service

- ng g service newservice
- ng g s newservice

* for Interface

- ng generate interface interfaceName

*for Class

- ng g class className

(Manually)

- Go to component (where new component to create) → Right Click to create component files → **.ts** file is compulsory needed, other than that **.html**, **.scss** & **.spec.ts** files are also created while creating component → add **@Component** Decorator in **.ts** file that contains metadata of component.

5) What is component in Angular 4?

⇨ Components are simply classes with decorators that mark their types and provide metadata which guide Angular to do things.

⇨ Every Angular application always has at least one component known as root component.

⇨ Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

6) How to run Angular 4 project?

⇨ By using **ng serve** command in Angular CLI.

7) What is service in Angular 4?

⇨ Angular services are singleton objects that get instantiated only once during the lifetime of an application.

⇨ They contain methods that maintain data throughout the life of an application, i.e. data does not get refreshed and is available all the time.

⇨ The main objective of a service is to organize and share business logic, models, or data and functions with different components of an Angular application.

8) What is Angular CLI?

⇨ Angular CLI is a command-line interface (CLI) to automate your development workflow.

⇨ It allows you to:

- Create a new Angular application
- Run a development server with LiveReload support to preview your application during development
- Add features to your existing Angular application
- Run your application's unit tests
- Run your application's end-to-end (E2E) tests
- Build your application for deployment to production.

9) How to call the data from service to component?

⇨ Through method (Has-A-Relationship).

10) What is Decorator in Angular 4?

⇨ Decorators are an excellent way to add metadata information to a class / method / property / parameter.

⇨ Angular uses quite a lot of decorators. There are decorators for classes, properties, methods and even parameters. Some of the important decorators are:

- NgModule
- Component
- Injectable
- Input
- Output

⇨ The Component decorator is used to decorate a class as an angular component and adds additional metadata information like the template, selector name, styles etc. to it.

11) Difference between Angular 1 vs 2 vs 4 vs 5 vs 6?

⇨ Angular 2

- Released in 2016
- Complete rewrite of Angular 1
- Written entirely in typescript
- Component-based instead of Controller
- More testable as component-based
- Support for Mobile/Low-end devices

⇨ Angular 3:

- Why we don't have Angular 3?

- Angular is being developed in a MonoRepo it means a single repo for everything. @angular/core, @angular/compiler, @angular/router etc are in the same repo and may have their own versions.
- The angular router was already in v3 and releasing angular 3 with router 4 will create confusion
- To avoid this confusion they decided to skip the version 3 and release with version 4.0.0 so that every major dependency in the MonoRepo are on the right track.

⇨ Angular 4
- Released in 2017
- Changes in core library
- Angular 4 is simply the next version of angular 2, the underlying concept is the same & is an inheritance from Angular 2
- Lot of performance improvement is made to reduce size of AOT compiler generated code
- Typescript 2.1 & 2.2 compatible — all feature of ts 2.1 & 2.2 are supported in Angular 4 application
- Animation features are separated from @angular/core to @angular/animation don't import @animation packages into the application to reduce bundle size and it gives the performance improvement.
- Else block in *ngIf introduced:

— Instead of writing 2 ngIf for else , simply add below code in component template:

*ngIf="yourCondition; else myFalsyTemplate"

"<ng-template #myFalsyTemplate>Else Html</ng-template>"

⇨ Angular 5
- Released 1st November 2017
- Build optimizer: It helps to removed unnecessary code from your application
- Angular Universal State Transfer API and DOM Support — by using this feature, we can now share the state of the application between the server side and client side very easily.
- Compiler Improvements: This is one of the very nice features of Angular 5, which improved the support of incremental compilation of an application.
- Preserve White space: To remove unnecessary new lines, tabs and white spaces we can add below code(decrease bundle size)

// in component decorator you can now add:

"preserveWhitespaces: false"

// or in tsconfig.json:

"angularCompilerOptions": { "preserveWhitespaces": false}`

- Increased the standardization across all browsers: For internationalization we were depending on `i18n` , but in ng 5 provides a new date, number, and currency pipes which increases the internationalization across all the browsers and eliminates the need of i18n polyfills.
- exportAs: In Angular 5, multiple names support for both directives and components
- HttpClient: until Angualar 4.3 @angular/HTTP was been used which is now depreciated and in Angular 5 a new module called HttpClientModule is introduced which comes under @angular/common/http package.
- Few new Router Life-cycle Events being added in Angular 5: In Angular 5 few new life cycle events being added to the router and those are:

- ActivationStart, ActivationEnd, ChildActivationStart, ChildActivationEnd, GuardsCheckStart, GuardsCheckEnd, ResolveStart and ResolveEnd.
- Angular 5 supports TypeScript 2.3 version.
- Improved in faster Compiler support:
- A huge improvement made in an Angular compiler to make the development build faster. We can now take advantage of by running the below command in our development terminal window to make the build faster.
  ng serve/s — aot

⇨ Angular 6

- Released on April 2018
- This release is focused less on the underlying framework, and more on tool-chain and on making it easier to move quickly with angular in the future
- No major breaking changes
- Dependency on RxJS 6 (this upgrade have breaking changes but CLI command helps in migrating from older version of RxJS)
- Synchronizes major version number of the:
  — Angular framework
  — Angular CLI
  — Angular Material + CDK
- All of the above are now version 6.0.0, minor and patch releases though are completely independent and can be changed based on a specific project.
- Remove support for <template> tag and "<ng-template>" should be used.
- Registering provider: To register new service/provider, we import Service into module and then inject in provider array. e.g:

```
// app.module.ts
import {MyService} from './my-service';
...
providers: [...MyService]
...
```

- But after this upgrade you will be able to add provided In property in injectable decorator. e.g:

```
// MyService.ts@Injectable({ providedIn: 'root'})
export class MyService{}
```

- The way ngModelChange event works:
  - Let's understand this with output produced by older and this version:
  - // Angular 5:
    <input [(ngModel)]='name' (ngModelChange)='onChange($event)' />
    onChange(value){ console.log(value); } // Would log updated value

    <input #modelDir='ngModel' [(ngModel)]='name'
    (ngModelChange)='onChange(modelDir)' />
    onChange(ngModel: NgModel){ console.log(ngModel.value); } // Would log old value, not updated
  - // Angular 6:

```
onChange(ngModel: NgModel){ console.log(ngModel.value); } // Would log updated value
```

CLI Changes: Two new commands have been introduced
— ng update <package>
* Analyse package.json and recommend updates to your application

* 3rd parties can provide update scripts using schematics
* automatically update code for breaking changes
* staying update and low maintenance
— ng add
* add new capablities to your applicaiton
* e.g ng add @angular/material : behind the scene it add bit of necessary code and changes project where needed to add it the thing we just told it to add.
* Now adding things like angular material, progressive web app, service workers & angular elements to your existing ng application will be easy.
CLI + Material starter templates: Let angular create code snippet for your basic components. e.g:
— Material Sidenav
* ng generate @angular/material:material-nav — name=my-nav
Generate a starter template including a toolbar with app name and then the side navigation & it's also responsive
— Dashboard
* ng generate @angular/material:material-dashboard — name=my-dashboard
Generates Dynamic list of cards
— Datatable
* ng generate @angular/material:material-table — name=my-table
Generates Data Table with sorting, filtering & pagination
It uses angular.json instead of .angular-cli.json
Support for multiple projects: Now in angular.json we can add multiple projects
initial release of Angular Elements which gives us ability to use our angular components in other environments like a Vue.js application. Its potential is truly amazing but unfortunately this release only works for angular application, we need to wait for next release to wrap out angular component into custom element and use it with framework like Vue.js

12) Difference between http and httpclient?

⇨ Http: The HttpClient is used to perform HTTP requests and it imported from @angular/common/http.

⇨ Httpclient: The HttpClient is more modern and easy to use the alternative of HTTP. HttpClient is an improved replacement for Http.

⇨ https://blog.hackages.io/angular-http-httpclient-same-but-different-86a50bbcc450

13) What is Rxjs?

⇨ RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous or event-based code.

⇨ RxJS can be used both in the browser and in the server-side using Node.

14) How to create single page application in Angular 4?

⇨ Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app. SPAs use AJAX and HTML5 to create a fluid and responsive Web apps, without constant page reloads. However, this means much of the work happens on the client side, in JavaScript.

⇨ By using Child Routing Process.

15) What is Observable?

⇨ RxJS introduces Observables, a new Push system for JavaScript. An Observable is a Producer of multiple values, "pushing" them to Observers (Consumers). A Function is a lazily evaluated computation that synchronously returns a single value on invocation.

⇨ Observable are just that — things you wish to observe and take action on. Angular uses the Observer pattern which simply means — Observable objects are registered, and other objects observe (in Angular using the subscribe method) them and take action when the observable object is acted on in some way.

16) What is Structural Directive & Different Structural Directives in Angular?

⇨ Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, or manipulating elements. As with other directives, you apply a structural directive to a host element.

⇨ Each structural directive does something different with that template.

⇨ There are 3 built-in structural directives – NgIf, NgFor and NgSwitch.

17) What is routing?

⇨ To handle the navigation from one view to the next, you use the Angular router. The router enables navigation by interpreting a browser URL as an instruction to change the view.

18) How routing works?

⇨ An Angular application that uses Angular Router only has one router service instance: It's a singleton. Whenever and wherever you inject the Router service in your application, you'll get access to the same Angular Router service instance.

⇨ To enable routing in our Angular application, we need to do three things:

a. create a routing configuration that defines the possible states for our application
b. import the routing configuration into our application
c. add a router outlet to tell Angular Router where to place the activated components in the DOM.

19) What is Angular 4 digest cycle?

⇨ Digest cycle is what Angular JS triggers when a value in the model or view is changed. The cycle sets off the watchers which then match the value of model and view to the newest value. Digest cycle automatically runs when the code encounters a directive.

20) How to pass data from one component to another?

⇨ Components can communicate with each other in various ways, including:

a. Parent to Child: via @Input()
b. Child to Parent: via @Output() and EventEmitter
c. Child to Parent: via @viewChild()
d. Unrelated Components: via a Service

⇨ https://fireship.io/lessons/sharing-data-between-angular-components-four-methods/