Module Name: - C

1)What do you mean by Dangling Pointer Variable in C Programming?

Ans: A Pointer in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the Pointer is still pointing to the same memory location, then that particular pointer variable is called as a Dangling Pointer Variable.

2)Differentiate between calloc() and malloc()

Ans: calloc() and malloc() are memory dynamic memory allocating functions. The only difference between them is that calloc() will load all the assigned memory locations with value 0 but malloc() will not.

3)Differentiate between Actual Parameters and Formal Parameters.

Ans: The Parameters which are sent from main function to the subdivided function are called as Actual Parameters and the parameters which are declared a the Subdivided function end are called as Formal Parameters.

4)What is Preprocessor?

Ans: A Preprocessor Directive is considered as a built-in predefined function or macro that acts as a directive to the compiler and it gets executed before the actual C Program is executed.

5) Differentiate between call by value and call by reference.

Call by reference in C

- In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.
- In call by value method, we can not modify the value of the actual parameter by the formal parameter.

- In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.
- The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.

Call by reference in C

- In call by reference, the address of the variable is passed into the function call as the actual parameter.
- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
- In call by reference, the memory allocation is similar for both formal parameters and actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.

Module Name: - CPP

1) Define inline function

If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. One of the important advantages of using an inline function is that it eliminates the function calling overhead of a traditional function.

2) Is deconstructor overloading possible? If yes then explain and if no then why?

No destructor overloading is not possible. Destructors take no arguments, so there's only one way to destroy an object. That's the reason destructor overloading is not possible.

3) What is the difference between virtual functions and pure virtual functions?

A virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword.

Example-

```
class base{
public:
 virtual void fun(){
 }
};
```

A pure virtual function is a function that has no implementation and is declared by assigning 0. It has no body.

Example-

```
class base{
public:
 virtual void fun()=0;
};
```

Here, = sign has got nothing to do with the assignment, and value 0 is not assigned to anything. It is used to simply tell the compiler that a function will be pure and it will not have anybody.

4) What is this pointer in C++?

The member functions of every object have a pointer named this, which points to the object itself. The value of this is set to the address of the object for which it is called. It can be used to access the data in the object it points to.

Example

```
class A{
 private:
  int value;
 public:
  void setvalue(int x){
   this->value=x;
  }
};
```

```
int main(){
 A a;
 a.setvalue(5);
 return 0;
}
```

5) How do you allocate and deallocate memory in C++?

The new operator is used for memory allocation and deletes operator is used for memory deallocation in C++.

For example-

```
int value=new int;              //allocates memory for storing 1 integer

delete value;                   // deallocates memory taken by value

int *arr=new int[10];           //allocates memory for storing 10 int

delete []arr;                   // deallocates memory occupied by arr
```

Module Name: - COP

1) Can we Overload or Override static methods in java?

Overriding : Overriding is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in superclass (or base class) at runtime.

Overloading: Overloading is related to compile time (or static) polymorphism. This feature allows different methods to have same name, but different signatures, especially number of input parameters and type of input parameters.

Can we overload static methods? The answer is 'Yes'. We can have two ore more static methods with same name, but differences in input parameters

Can we Override static methods in java? We can declare static methods with same signature in subclass, but it is not considered overriding as there won't

be any run-time polymorphism. Hence the answer is 'No'. Static methods cannot be overridden because method overriding only occurs in the context of dynamic (i.e. runtime) lookup of methods. Static methods (by their name) are looked up statically (i.e. at compile-time).

2) Why the main method is static in java?

The method is static because otherwise there would be ambiguity: which constructor should be called? Especially if your class looks like this:

```
public class JavaClass
{
  protected JavaClass(int x)
  {   }
   public void main(String[] args)
   {

    }
}
```

Should the JVM call new JavaClass(int)? What should it pass for x? If not, should the JVM instantiate JavaClass without running any constructor method? because that will special-case your entire class – sometimes you have an instance that hasn't been initialized, and you have to check for it in every method that could be called. There are just too many edge-cases and ambiguities for it to make sense for the JVM to have to instantiate a class before the entry point is called. That's why main is static.

3) What is the scope of variables in Java in following cases?

Member Variables (Class Level Scope) : The member variables must be declared inside class (outside any function). They can be directly accessed anywhere in class

Local Variables (Method Level Scope) : Variables declared inside a method have method level scope and can't be accessed outside the method.

Loop Variables (Block Scope) : A variable declared inside pair of brackets "{" and "}" in a method has scope within the brackets only.

4) What is the Final Keyword in Java?

Java final keyword is a non-access specifier that is used to restrict a class, variable, and method. If we initialize a variable with the final keyword, then we cannot modify its value.

If we declare a method as final, then it cannot be overridden by any subclasses. And, if we declare a class as final, we restrict the other classes to inherit or extend it.

In other words, the final classes can not be inherited by other classes.

Final Variable in Java

Once we declare a variable with the final keyword, we can't change its value again. If we attempt to change the value of the final variable, then we will get a compilation error.

Generally, we can consider a final variable as a constant, as the final variable acts like a constant whose values cannot be changed.

This rule was for the normal variables, what if we declare the reference variables as final? The restriction with the final reference variable is that we cannot change "what the object is referring to" but we can modify the object itself.

We can't just declare a variable as final without initializing it. That is, we have to give it an initial value while declaring a final variable. If we declare the final variable without initialization, then it is a blank final variable.

Final Method in Java

As earlier, we discussed the Final Keyword and How to declare the Final Variable. We can declare Java methods as Final Method by adding the Final keyword before the method name.

The Method with Final Keyword cannot be overridden in the subclasses. The purpose of the Final Method is to declare methods of how's definition can not be changed by a child or subclass that extends it.

Final Class in Java

We can also declare a class with a final keyword in Java. When we declare a class as final, then we restrict other classes to inherit or extend it.

In short, Java final class can't be extended by other classes in the inheritance. If another class attempts to extend the final class, then there will be a compilation error.

What is “super” keyword in java?

The super keyword in java is a reference variable that is used to refer parent class objects. The keyword “super” came into the picture with the concept of Inheritance. Whenever you create the instance of a subclass, an instance of parent class is created implicitly i.e. referred by super reference variable.

Various scenarios of using java super Keyword:

super is used to refer immediate parent instance variable

super is used to call parent class method

super() is used to call immediate parent constructor

5) How are Java objects stored in memory?

In Java, all objects are dynamically allocated on Heap. This is different from C++ where objects can be allocated memory either on Stack or on Heap. In C++, when we allocate object using new(), the object is allocated on Heap, otherwise on Stack if not global or static.

In Java, when we only declare a variable of a class type, only a reference is created (memory is not allocated for the object). To allocate memory to an object, we must use new(). So the object is always allocated memory on the heap.

Module Name: - OS

1) What is a Pipe and when it is used?

The pipe is generally a connection among two or more processes that are interrelated to each other. It is a mechanism that is used for inter-process communication using message passing. One can easily send information such as the output of one program process to another program process using a pipe. It can be used when two processes want to communicate one-way i.e., inter-process communication (IPC).

2) What is IPC? What are the different IPC mechanisms?

IPC (Interprocess Communication) is a mechanism that requires the use of resources like a memory that is shared between processes or threads. With IPC, OS allows different processes to communicate with each other. It is simply used for exchanging data between multiple threads in one or more programs or processes. In this mechanism, different processes can communicate with each other with the approval of the OS.

Different IPC Mechanisms:

Pipes

Message Queuing

Semaphores

Socket

Shared Memory

Signals

3) What is virtual memory?

It is a memory management technique feature of OS that creates the illusion to users of a very large (main) memory. It is simply space where a greater number of programs can be stored by themselves in the form of pages. It enables us to increase the use of physical memory by using a disk and also allows us to have memory protection. It can be managed in two common ways by OS i.e., paging and segmentation. It acts as temporary storage that can be used along with RAM for computer processes.

4) Explain file permission in Linux.

There are 3 kinds of permission in Linux:

Read: Allows a user to open and read the file

Write: Allows a user to open and modify the file

Execute: Allows a user to run the file.

You can change the permission of a file or a directory using the chmod command. There are two modes of using the chmod command:

Symbolic mode

Absolute mode

Symbolic mode
The general syntax to change permission using Symbolic mode is as follows:

$ chmod <target>(+/-/=)<permission> <filename>

where <permissions> can be r: read; w: write; x: execute.

<target> can be u : user; g: group; o: other; a: all

'+' is used for adding permission

'-' is used for removing permission

'=' is used for setting the permission

For example, if you want to set the permission such that the user can read, write, and execute it and members of your group can read and execute it, and others may only read it.

Then the command for this will be:

$ chmod u=rwx,g=rx,o=r filename

Absolute mode
The general syntax to change permission using Absolute mode is as follows:

$ chmod <permission> filename

The Absolute mode follows octal representation. The leftmost digit is for the user, the middle digit is for the user group and the rightmost digit is for all.

5) Explain Process Management System Calls in Linux

The System Calls to manage the process are:

fork () : Used to create a new process
exec() : Execute a new program
wait() : Wait until the process finishes execution
exit() : Exit from the process
And the System Calls used to get Process ID are:

getpid():- get the unique process id of the process
getppid():- get the parent process unique id

Module Name: - DBT

1) What is Normalization?

In SQL, normalization of data is a process through with data is organized in tables used for the reduction of redundancy and dependency of data. It divides large tables into smaller ones using some set of rules.

Types of normalization:

- **1NF:** The rules of 1NF are that each table must contain a single value and records are required to be unique.
- **2NF:** The rules of 2NF are that the table must be in 1NF and must possess a single-column primary key.
- **3NF:** The rules of 3NF are that the table must be in 2NF and must not have any transitive functional dependencies.
- **BCNF:** The rules of the Boyce Codd Normal form is that it must be in 3 NF and must not have more than one candidate key.

2) What do DDL, DML, and DCL stand for?

DDL is the abbreviation for Data Definition Language dealing with database schemas, as well as the description of how data resides in the database. An example of this is the CREATE TABLE command.

DML denotes Data Manipulation Language which includes commands such as SELECT, INSERT, etc.

DCL stands for Data Control Language and includes commands like GRANT, REVOKE, etc.

3) What is the difference between primary key and unique key?

While both are used to enforce the uniqueness of the column defined, the primary key would create a clustered index, whereas the unique key would create a non-clustered index on the column. The primary key does not allow 'NULL', but the unique key does.

4) What is the difference between LIKE and REGEXP operators in MySQL?

LIKE is denoted using the '%' sign. For example

SELECT * FROM user WHERE user name LIKE "%NAME"

On the other hand, the use of REGEXP is as follows:

SELECT * FROM user WHERE username REGEXP "^NAME";

5) What is the difference between the DELETE TABLE and TRUNCATE TABLE commands in MySQL?

Basically, DELETE TABLE is a logged operation, and every row deleted is logged. Therefore, the process is usually slow. TRUNCATE TABLE also deletes rows in a table, but it will not log any of the rows deleted. The process is faster here in comparison.

TRUNCATE TABLE can be rolled back and is functionally similar to the DELETE statement without a WHERE clause.

Module Name: - Core JAVA

1)What is Singleton Class in Java?

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time. After the first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.

Remember the key points while defining class as a singleton class that is while designing a singleton class:

Make a constructor private.
Write a static method that has the return type object of this singleton class. Here, the concept of Lazy initialization is used to write this static method.
Purpose of Singleton Class
The primary purpose of a Single class is to restrict the limit of the number of object creation to only one. This often ensures that there is access control to resources, for example, socket or database connection.

The memory space wastage does not occur with the use of the singleton class because it restricts the instance creation. As the object creation will take place only once instead of creating it each time a new request is made.

We can use this single object repeatedly as per the requirements. This is the reason why the multi-threaded and database applications mostly make use of the Singleton pattern in Java for caching, logging, thread pooling, configuration settings, and much more.

For example, there is a license with us, and we have only one database connection or suppose if our JDBC driver does not allow us to do multithreading, then Singleton class comes into the picture and makes sure that at a time, only a single connection or a single thread can access the connection.

How to Design/Create a Singleton Class in Java?
To create a singleton class, we must follow the steps, given below:
1. Ensure that only one instance of the class exists.

2. Provide global access to that instance by

Declaring all constructors of the class to be private.
Providing a static method that returns a reference to the instance. The lazy initialization concept is used to write the static methods.
The instance is stored as a private static variable.
Example of singleton classes is Runtime class, Action Servlet, Service Locator. Private constructors and factory methods are also an example of the singleton class.

Difference between Normal Class and Singleton Class
We can distinguish a Singleton class from the usual classes with respect to the process of instantiating the object of the class. To instantiate a normal class, we use a java constructor. On the other hand, to instantiate a singleton class, we use the getInstance() method.

The other difference is that a normal class vanishes at the end of the lifecycle of the application while the singleton class does not destroy with the completion of an application.
Inheritance.

2)Which are the Object Class methods in java?

Method	Description
public final Class getClass()	returns the Class class object of this object. The Class class can further be used to get the metadata of this class.
public int hashCode()	returns the hashcode number for this object.
public boolean equals(Object obj)	compares the given object to this object.
protected Object clone() throws CloneNotSupportedException	creates and returns the exact copy (clone) of this object.
public String toString()	returns the string representation of this object.
public final void notify()	wakes up single thread, waiting on this object's monitor.
public final void notifyAll()	wakes up all the threads, waiting on this object's monitor.
public final void wait(long timeout)throws InterruptedException	causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
public final void wait(long timeout,int nanos)throws InterruptedException	causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).
public final void wait()throws InterruptedException	causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).
protected void finalize()throws Throwable	is invoked by the garbage collector before object is being garbage collected.

The Object class is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

3) Explain override equals() method

The equals method is used to check the similarity between two objects. In case if the programmer wants to check an object based on the property, then it needs to be overridden.

4) What is Java Anonymous Class?

In Java, a class can contain another class known as nested class. It's possible to create a nested class without giving any name.

A nested class that doesn't have any name is known as an anonymous class.

An anonymous class must be defined inside another class. Hence, it is also known as an anonymous inner class. Its syntax is:

```
class outerClass {
   // defining anonymous class
   object1 = new Type(parameterList) {
       // body of the anonymous class
   };
}
```

Anonymous classes usually extend subclasses or implement interfaces.

Here, Type can be

a superclass that an anonymous class extends

an interface that an anonymous class implements

The above code creates an object, object1, of an anonymous class at runtime.

Note: Anonymous classes are defined inside an expression. So, the semicolon is used at the end of anonymous classes to indicate the end of the expression.

5) How are Java objects stored in memory?

In Java, all objects are dynamically allocated on Heap. This is different from C++ where objects can be allocated memory either on Stack or on Heap. In C++, when we allocate object using new(), the object is allocated on Heap, otherwise on Stack if not global or static.

In Java, when we only declare a variable of a class type, only a reference is created (memory is not allocated for the object). To allocate memory to an object, we must use new(). So the object is always allocated memory on the heap. Read more

Polymorphism

Q1: What are the advantages of Polymorphism in Java?
Ans:
Polymorphism, the name itself says one name and many functions. The flexibility of the code is achieved because of performing many functionalities by using different methods with the same names but different parameters.

The advantage of using Polymorphism is the ability to take more than one form by using a single interface. when single or many types are not defined by name it represents any type by using abstract symbols.

Q2: How does inheritance used in polymorphism in java?
Ans:
Inheritance means a process where a sub-class/child class acquires all the properties and behaviors of the parent class/superclass.

Polymorphism takes advantage of the parent-child relationship between two classes and makes the program more Dynamic by adding dynamic features in the code.

Q3: What is overloading in polymorphism in java?
Ans:
Overloading is also known by another name Static binding. In the process of overloading:

The method has the same names with different signatures and different types of parameters.
The method signature and number of parameters are the same but the types are different.
Overloading is also implemented when the different signature methods and parameters of numbers or types are used.
The overloading method is implemented at compile-time.

Q4: What is difference between Static binding and Dynamic binding?
Ans:
Static binding is one of the methods used or implemented in polymorphisms. Another name for static binding is Early binding. An object is linked with the function during compile-time is called Static binding.

In simple words, Static binding in polymorphism means the same method name with a different signature. In the same class we have multiple methods with the same name but with different parameters is called a Static class. The overloading method is implemented at compile-time to achieve static binding.

Dynamic binding is one of the methods used or implemented in polymorphism. Another name for Dynamic binding is Run-time. In Dynamic binding/Run-time, there are different classes where inheritance is implemented and methods with the same parameters.

In Dynamic polymorphism, the resolve of the file/code is done at a Run-time and not at a compile-time by using the Overriding method. Following the method of overriding is also called Dynamic

binding. In Dynamic binding polymorphism, the code is extended and becomes flexible and easy to manage.

Q5:Why binding of private, static, and final methods are always static binding in Java?
Ans:
Static, final, and private methods are early binding because they can't be overridden. The call to the object is already assigned to a defined location at the start of the execution of a program, which is known as early binding.

Late binding, on the other hand, selects the method during runtime. The method that will be called is determined by the class that the reference is pointing to. This is particularly true for interface methods, which cannot be bound early since they do not point to a specific implementation.

Abstraction

1. What is Abstraction in Java?

Ans: Abstraction in Java is a technique by which we can hide the data that is not required to users. It hides all unwanted data so that users can work only with the required data.

2. How to achieve or implement Abstraction in Java?

Ans: There are two ways to implement abstraction in java. They are as follows:

a) Abstract class (0 to 100%)
b) Interface (100%)

3. What is Abstract class in Java? How to define it?

Ans: An abstract class in java is a class that is declared with an abstract keyword.

Example of Abstract class:

```
abstract class Test {
abstract void show();
}
```

4. What is the difference between abstract class and concrete class?

Ans: There are mainly two differences between an abstract class and concrete class. They are:

a) We cannot create an object of abstract class. Only objects of its non-abstract (or concrete) sub classes can be created.

b) It can have zero or more abstract methods that are not allowed in a non-abstract class (concrete class).

5. What is Abstract in Java?

Ans: Abstract is a non-access modifier in java that is applicable for classes, interfaces, methods, and inner classes.

6. Can abstract modifier applicable for variables?

Ans: No.

7. What is Abstract method in Java?

Ans: A method which is declared with abstract modifier and has no implementation (means no body) is called abstract method in java.

It does not contain any body. It has simply a signature declaration followed by a semicolon. It has the following general form as given below.

Encapsulation

1. What is Encapsulation in Java? Why is it called Data hiding?

Ans: The process of binding data and corresponding methods (behavior) together into a single unit is called encapsulation in Java.

In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from being accessed by other classes, thereby we can keep variables and methods safes from outside interference and misuse.

If a field is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within the class. Therefore, Encapsulation is also called data hiding.

2. What are the important features of Encapsulation?

Ans: Encapsulation means combining the data of our application and its manipulation in one place. It allows the state of an object to be accessed and modified through behavior. It reduces the coupling of modules and increases the cohesion inside them.

3. What is the advantage of Encapsulation?

Ans: There are the following advantages of encapsulation in Java. They are as follows:
The encapsulated code is more flexible and easy to change with new requirements.
It prevents the other classes to access the private fields.
Encapsulation allows modifying implemented code without breaking other code who have implemented the code.
It keeps the data and codes safe from external inheritance. Thus, Encapsulation helps to achieve security.
It improves the maintainability of the application.

4. What are the main benefits of using encapsulation in Java?

Ans: The main benefits of using encapsulation are:
The main benefit of encapsulation is the ability to modify the implemented code without breaking the others code who have implemented the code.
It also provides us with maintainability, flexibility, and extensibility to our code.
The fields of a class can be made read-only or write-only.
A class can have total control over what is stored in its fields.
5. How to achieve encapsulation in Java? Give an example.

Ans: There are two key points that should be kept in mind to achieve the encapsulation in Java. They are as follows:

Declare the variable of the class as private.
Provide public setter and getter methods to modify the values of variables.

Inheritance

1. What is Inheritance in Java?

Ans: The technique of creating a new class by using an existing class functionality is called inheritance in Java. In other words, inheritance is a process where a child class acquires all the properties and behaviors of the parent class.

2. Why do we need to use inheritance?
Or, What is the purpose of using inheritance?

Ans: Inheritance is one of the main pillars of OOPs concept. Some objects share certain properties and behaviors. By using inheritance, a child class acquires all properties and behaviors of parent class.

There are the following reasons to use inheritance in java.

We can reuse the code from the base class.
Using inheritance, we can increase features of class or method by overriding.
Inheritance is used to use the existing features of class.
It is used to achieve runtime polymorphism i.e method overriding.

3. What is Is-A relationship in Java?

Ans: Is-A relationship represents Inheritance. It is implemented using the “extends” keyword. It is used for code reusability.

4. What is super class and subclass?

Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent class.

A class that inherits all the members (fields, method, and nested classes) from other class is called subclass. It is also called a derived class, child class, or extended class.

5. How is Inheritance implemented/achieved in Java?

Ans: Inheritance can be implemented or achieved by using two keywords:

extends: extends is a keyword that is used for developing the inheritance between two classes and two interfaces.
implements: implements keyword is used for developing the inheritance between a class and interface.

6. Write the syntax for creating the subclass of a class?
Ans: A subclass can be created by using the "extends" keyword. The syntax for declaring a subclass of class is as follows:

```
class subclassName extends superclassName
{
  // Variables of subclass
  // Methods of subclass
}
```

where class and extends are two keywords.

7. Which class in Java is superclass of every other class?

Ans: In Java, Object class is the superclass of every other class.

8. How will you prove that the features of Superclass are inherited in Subclass?

Ans: Refer to this tutorial: Inheritance in Java with Realtime Example

9. Can a class extend itself?

Ans: No, a class cannot extend itself.

10. Can we assign superclass to subclass?

Ans: No.

11. Can a class extend more than one class?

Ans: No, one class can extend only a single class.

12. Are constructor and instance initialization block inherited to subclass?

Ans: No, constructor and instance initialization block of the superclass cannot be inherited to its subclass but they are executed while creating an object of the subclass.

String

1. What is String in Java? Is it a datatype?
The string is a final class in Java defined in java.lang package. You can assign a sequence of characters to a string variable. For example String name = "Gaurav";

No, String is not a datatype like int, char, or long.

When you assign a sequence of character to String variable, you are creating a string object.

Every String literal is an instance of the String class, and its value can not be changed.

2. What is the difference between String in C language and String in Java?
If your resume contains something related to the 'C' language, they can ask you this question.

String in Java and C is completely different. In 'C' language String is a null-terminated character array.

In the image given below, I have shown the structure of the string in C and Java.

Showing String in C and Java
Showing String in C and Java

The string is more abstract in Java.

The string class comes with java.lang package and has lots of predefined methods that a programmer can use to operate on a string or get information about a String.

So String is more feature-rich in Java than C.

3. What is the String pool in Java?
The String pool is a special type of memory maintained by the JVM.

String pool is used to store unique string objects.

When you assign the same string literal to different string variables, JVM saves only one copy of the String object in the String pool, and String variables will start referring to that string object.

I have shown the pictorial explanation of the above sentence in the following diagram.

Two string variables pointing to the single string object from string pool
Two string variables pointing to the single string object from the string pool

The purpose of maintaining this special type of memory is memory optimization.

4. Why String is immutable?
In most of the Java Interviews, you will face this question. Why do you think Java language designers kept string immutable?

You can give the following reasons.

Java String pool is possible because the String is immutable.

If you assign the same string literal to many string variables, JVM will save only one copy of the string object in the Java string pool, and these variables will start referring to that string object.

If you were not asked about the String pool before this question, please give a little background about the string pool concept in Java. Please refer to the previous question.

Also, another reason can be Security. We know that almost every Java program contains a string, and it is used to save important data like usernames and passwords. So it should not be changed in-between. Otherwise, there will be a security problem.

5. How many objects will be created from the following code?

```
String firstString = "Gaurav";
String secondString = "Gaurav";
String thirdString =  new String("Gaurav");
String firstString = "Gaurav";
String secondString = "Gaurav";
String thirdString =  new String("Gaurav");
```

By seeing the above code, only two string objects will be created. The first two variables will refer to the same string object with the value "Gaurav". JVM uses the string pool concept to store only one copy of duplicate string objects to string constant pool.

But when we use a new keyword to create a new string, a new string object will be created and stored in the Java heap memory.

So for the third variable thirdString, a new string object will be created and stored in a Java heap space.

So there will be a total of two objects, one from the Java string pool and one from the Java heap memory.

Below, I have shown these two objects in the following diagram.

Showing the two string objects from the Java string pool and Java heap memory
Showing the two string objects from the Java string pool and Java heap memory

6. What is the intern() method?

intern() method is used to add the unique copy of the string object to the Java string pool manually.

We know, when we create a string using a new keyword, it will be stored in the heap memory.

We can store the unique copy of that string object in the Java string pool using the intern() method.

When you do such a thing, JVM will check if the string object with the same value is present in the string pool or not.

If a string object with the same value is present, JVM will simply provide the reference of that object to the respective string variable.

If a string object with the same value is not present in the string pool, JVM creates a string object with the same value in the String pool and returns its reference to the string variable.

7. What is the difference between the String and StringBuffer?
The String is a final class in Java. The String is immutable. That means we can not change the value of the String object afterword.

Since the string is widely used in applications, we have to perform several operations on the String object. Which generates a new String object each time, and all previous objects will be garbage object putting the pressure on the Garbage collector.

Hence, the Java team introduced the StringBuffer class. It is a mutable String object, which means you can change its value.

The string is immutable, but the StringBuffer is mutable.

8. What is the difference between the StringBuffer and StringBuilder?
We know String is immutable in Java. But using StringBuffer and StringBuilder, you can create editable string objects.

When Java Team realizes the need for the editable string object, they have introduced the StringBuffer class. But all the methods of the StringBuffer class are synchronized. That means at a time, only one thread can access a method of the StringBuffer.

As a result, it was taking more time.

Latter, Java Team realizes that making all methods of the StringBuffer class synchronized was not a good idea, and they introduced a StringBuilder class. None of the methods of the StringBuilder class are synchronized.

Since all the methods of the StringBuffer class are synchronized, StringBuffer is thread-safe, slower, and less efficient as compared to StringBuilder.

Since none of the methods of the StringBuilder class is synchronized, StringBuilder is not thread-safe, faster, and efficient as compared to StringBuffer.

Also, you can check my detailed article on difference between theStringBuffer and StringBuilder.

9. Can we compare String using the == operator? What is the risk?
Yes, of course, we can compare String using the == operator. But when we are comparing string using the == operator, we are comparing their object reference, whether these string variables are pointing towards the same string object or not.

Most of the time, developers want to compare the content of the strings, but mistakenly they compare strings with == operator, instead of equals() method, which leads to an error.

Exception

1). What is catching an exception in Java?

Ans: The process of finding a handler by JVM to handle thrown exception is called catching an exception.

2). What will happen to exception object after exception handling is done?

Ans: Once exception handling is done, the exception object will be garbage collected.

3). How will you handle the checked exception?
Or What are the different ways to handle checked exceptions?

Ans: A checked exception can be handled either by using try and catch block or by using throws clause in the method declaration. If not handles properly, it will give a compile-time error.

4). Which exception class can you use in the catch block to handle both checked and unchecked exceptions?

Ans: Exception class

5). Can we throw checked exceptions from the static block?

Ans: We cannot throw because there is no specific place to catch it and it is called only once.

Abstract Class

1) Abstract class must have only abstract methods. True or false?

False. Abstract methods can also have concrete methods.

2) Is it compulsory for a class which is declared as abstract to have at least one abstract method?

Not necessarily. Abstract class may or may not have abstract methods.

3) Can we use “abstract” keyword with constructor, Instance Initialization Block and Static Initialization Block?

No. Constructor, Static Initialization Block, Instance Initialization Block and variables can not be abstract.

4) Why final and abstract can not be used at a time?

Because, final and abstract are totally opposite in nature. A final class or method can not be modified further where as abstract class or method must be modified further. “final” keyword is used to denote that a class or method does not need further improvements. “abstract” keyword is used to denote that a class or method needs further improvements.

5) Can we instantiate a class which does not have even a single abstract methods but declared as abstract?

No, We can’t instantiate a class once it is declared as abstract even though it does not have abstract methods.

6) Can we declare abstract methods as private? Justify your answer?

No. Abstract methods can not be private. If abstract methods are allowed to be private, then they will not be inherited to sub class and will not get enhanced.

Interfaces

1. What is an interface in Java?

Ans: An interface in Java is a mechanism that is used to achieve complete abstraction. It is basically a kind of class that contains only constants and abstract methods.

2. Can we define private and protected modifiers for data members (fields) in interfaces?

Ans: No, we cannot define private and protected modifiers for variables in interface because the fields (data members) declared in an interface are by default public, static, and final.

3. Which modifiers are allowed for methods in an Interface?

Ans: Only abstract and public modifiers are allowed for methods in interfaces.

4. Suppose A is an interface. Can we create an object using new A()?

Ans: No, we cannot create an object of interface using new operator. But we can create a reference of interface type and interface reference refers to objects of its implementation classes.

5. Can we define an interface with a static modifier?

Ans: Yes, from Java 8 onwards, we can define static and default methods in an interface. Prior to Java 8, it was not allowed.

6. Suppose A is an interface. Can we declare a reference variable a with type A like this: A a;

Ans: Yes.

7. Can an interface extends another interface in Java?

Ans: Yes, an interface can extend another interface.

8. Can an interface implement another interface?

Ans: No, an interface cannot implement another interface.

9. Is it possible to define a class inside an interface?

Ans: Yes, we can define a class inside an interface.

Constructor

1. What is a Constructor in Java?
Constructor is just like a method in Java that is used to initialize the state of an object and will be invoked during the time of object creation.

2. What are the Rules for defining a constructor?
Constructor name should be the same as the class name
It cannot contain any return type
It can have all Access Modifiers are allowed (private , public, protected, default)
It Cannot have any Non Access Modifiers (final ,static, abstract, synchronized)
No return statement is allowed
It can take any number of parameters
Constructor can throw exception, we can have throws clause

3. What is the use of Private Constructors in Java?
When we use private for a constructor then object for the class can only be created internally within the class, no outside class can create object for this class. Using this we can restrict the caller from creating objects.

```
class PrivateConstructorExample
{
   /**
    * Private Constructor for preventing object creation
   from outside class
   **/
   private PrivateConstructorExample(){ }

   public void disp()
   {
      System.out.println("disp() method called");
   }
}
public class Sample
{
```

```
  public static void main(String args[])
  {
    //Creating the object for the Private Constructor class
    PrivateConstructorExample pc = new PrivateConstructorExample();

    pc.disp();
  }
}
```

When we run the above code we will be getting the below exception.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The constructor PrivateConstructorExample() is not visible

        at Sample.main(Sample.java:19)
```

4. Can we have a Constructor in an Interface?

No, We cannot have a Constructor defined in an Interface.

5. What is Constructor Chaining in Java?

Constructor Chaining is nothing but calling one Constructor from another. this keyword is used to call the current class constructor and super keyword is used to call the parent class constructor.

```
class Parent
{
  public Parent()
  {
    System.out.println("Parent class no-args constructor called");
  }
  public Parent(String name)
  {
    System.out.println("Parent class Parameterized constructor called by "+name);
  }
}
public class Child extends Parent
{
  public Child()
  {
    this("JIP");
    System.out.println("Child class no-args constructor called");
  }
  public Child(String name)
  {
    super("JIP");
    System.out.println("Child class Parameterized constructor called by "+name);
  }
  public static void main(String args[])
  {
    Child c = new Child();
  }
```

```
}
```
Output :

```
Parent class Parameterized constructor called by JIP
Child class Parameterized constructor called by JIP
Child class no-args constructor called
```
6. Can we have this and super in the same constructor?
No, we cannot have have this and super in a same constructor as any one only can be in the first line of the constructor.

```
class Parent
{
   public Parent()
   {
      System.out.println("Parent class no-args constructor");
   }
}
public class Child extends Parent
{
   public Child()
   {
      this("JIP");
      super();
      System.out.println("Child class no-args constructor");
   }
   public Child(String name)
   {

      System.out.println("Child class Parameterized constructor"+name);
   }
   public static void main(String args[])
   {
      Child c = new Child();
   }
}
```
Output :

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Constructor call must be the first statement in a constructor

        at Child.(Child.java:13)
        at Child.main(Child.java:23)
```
7. Is it possible to call a sub class constructor from super class constructor?
No. You cannot call a sub class constructor from a super class constructor.

8. What is a No-arg constructor?
Constructor without arguments is called no-arg constructor. In Java Default constructor is a no-arg constructor.

```
class Demo
{
   public Demo()
   {
      //No-arg constructor
   }
}
```

9. Can we have a class with no Constructor in it ? What will happen during object creation ?
Yes, we can have a class with no constructor, When the compiler encounters a class with no constructor then it will automatically create a default constructor for you.

10. Can we have both Default Constructor and Parameterized Constructor in the same class?
Yes, we have both Default Constructor and Parameterized Constructor in the same class.

11. Can a Constructor return any value ?
A Constructor cannot return any explicit value but implicitly it will be returning the instance of the class.

12. Will compiler create the Default Constructor when we already have a Constructor defined in the class ?
No, the compiler will not create the Default Constructor when we already have a Constructor defined.

13. Can an abstract class in Java have a constructor?
Yes, an abstract class can have a constructor. The below code work perfectly fine.

Collection

1) What is the Collection framework in Java?
Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.

2) What are the main differences between array and collection?
Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data, but they differ in many ways. The main differences between the array and Collection are defined below:

Arrays are always of fixed size, i.e., a user can not increase or decrease the length of the array according to their requirement or at runtime, but In Collection, size can be changed dynamically as per need.
Arrays can only store homogeneous or similar type objects, but in Collection, heterogeneous objects can be stored.
Arrays cannot provide the ?ready-made? methods for user requirements as sorting, searching, etc. but Collection includes readymade methods to use.
3) Explain various interfaces used in Collection framework?

Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:

1. Collection interface: Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.

Syntax:

public interface Collection<E>extends Iterable
Where <E> represents that this interface is of Generic type

2. List interface: List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.

Syntax:

public interface List<E> extends Collection<E>
3. Set interface: Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface

Syntax:

public interface Set<E> extends Collection<E>
Queue interface: Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).

Syntax:

public interface Queue<E> extends Collection<E>
4. Dequeue interface: it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.

Syntax:

public interface Dequeue<E> extends Queue<E>
5. Map interface: A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.

Module Name: - Advanced JAVA

Multithreading

1)What is Thread in Java?
Threads are basically the lightweight and smallest unit of processing that can be managed independently by a scheduler. Threads are referred to as parts of a process that simply let a program execute efficiently with other parts or threads of the process at the same time. Using threads, one can perform complicated tasks in the easiest way. It is considered the simplest way to take advantage of multiple CPUs available in a machine. They share the common address space and are independent of each other.

2. What are the two ways of implementing thread in Java?
There are basically two ways of implementing thread in java as given below:

Extending the Thread class
Example:

```
class MultithreadingDemo extends Thread
{
  public void run()
 {
    System.out.println("My thread is in running state.");
 }
  public static void main(String args[])
 {
   MultithreadingDemoobj=new MultithreadingDemo();
      obj.start();
  }
}
```

Output:
My thread is in running state.

Implementing Runnable interface in Java
Example:

```
class MultithreadingDemo implements Runnable
{
   public void run()
 {
     System.out.println("My thread is in running state.");
  }
    public static void main(String args[])
 {
     MultithreadingDemo obj=new MultithreadingDemo();
     Threadtobj =new Thread(obj);       tobj.start();
 }
}
```

Output:

My thread is in running state.

3. What's the difference between thread and process?
Thread: It simply refers to the smallest units of the particular process. It has the ability to execute different parts (referred to as thread) of the program at the same time.

Process: It simply refers to a program that is in execution i.e., an active program. A process can be handled using PCB (Process Control Block).

4) What's the difference between User thread and Daemon thread?
User and Daemon are basically two types of thread used in Java by using a 'Thread Class'.

User Thread (Non-Daemon Thread): In Java, user threads have a specific life cycle and its life is independent of any other thread. JVM (Java Virtual Machine) waits for any of the user threads to complete its tasks before terminating it. When user threads are finished, JVM terminates the whole program along with associated daemon threads.

Daemon Thread: In Java, daemon threads are basically referred to as a service provider that provides services and support to user threads. There are basically two methods available in thread class for daemon thread: setDaemon() and isDaemon().

5.What are the wait() and sleep() methods?
wait(): As the name suggests, it is a non-static method that causes the current thread to wait and go to sleep until some other threads call the notify () or notifyAll() method for the object's monitor (lock). It simply releases the lock and is mostly used for inter-thread communication. It is defined in the object class, and should only be called from a synchronized context.

Example:

```
synchronized(monitor)
{
monitor.wait();      Here Lock Is Released by Current Thread
}
```

sleep(): As the name suggests, it is a static method that pauses or stops the execution of the current thread for some specified period. It doesn't release the lock while waiting and is mostly used to introduce pause on execution. It is defined in thread class, and no need to call from a synchronized context.

Example:

```
synchronized(monitor)
{
Thread.sleep(1000);     Here Lock Is Held by The Current Thread
//after 1000 milliseconds, the current thread will wake up, or after we call that is interrupt() method
}
```

9. What's the difference between notify() and notifyAll()?

notify(): It sends a notification and wakes up only a single thread instead of multiple threads that are waiting on the object's monitor.

notifyAll(): It sends notifications and wakes up all threads and allows them to compete for the object's monitor instead of a single thread.

6. Why wait(), notify(), and notifyAll() methods are present in Object class?
We know that every object has a monitor that allows the thread to hold a lock on the object. But the thread class doesn't contain any monitors. Thread usually waits for the object's monitor (lock) by calling the wait() method on an object, and notify other threads that are waiting for the same lock using notify() or notifyAll() method. Therefore, these three methods are called on objects only and allow all threads to communicate with each that are created on that object.

7. What is Runnable and Callable Interface? Write the difference between them.
Both the interfaces are generally used to encapsulate tasks that are needed to be executed by another thread. But there are some differences between them as given below:

Running Interface: This interface is basically available in Java right from the beginning. It is simply used to execute code on a concurrent thread.
Callable Interface: This interface is basically a new one that was introduced as a part of the concurrency package. It addresses the limitation of runnable interfaces along with some major changes like generics, enum, static imports, variable argument method, etc. It uses generics to define the return type of object.

```
public interface Runnable
{
  public abstract void run();
}
public interface Callable<V>
{
 call() throws Exception;
}
```

Servlet

1) How many objects of a servlet is created?
Only one object at the time of first request by servlet or web container.
2) What is the life-cycle of a servlet?
Servlet is loaded
servlet is instantiated
servlet is initialized
service the request
servlet is destroyed

3. What is difference between GenericServlet and HttpServlet?
The GenericServlet is protocol independent whereas HttpServlet is HTTP protocol specific. HttpServlet provides additional functionalities such as state management etc.

4What is the purpose of RequestDispatcher Interface?
The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of antoher resource.

5) Can you call a jsp from the servlet?
Yes, one of the way is RequestDispatcher interface for example:

```
RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");
rd.forward(request,response);
```

Hibernate

1)What does HQL stand for?
Hibernate Query Language

2) How is HQL query created?
The HQL query is created with the help of the following syntax:

```
Session.createQuery
```

3) How can we add criteria to a SQL query?
A criterion is added to a SQL query by using the Session.createCriteria.

4) Define persistent classes.
Classes whose objects are stored in a database table are called as persistent classes.

5) What is SessionFactory?
SessionFactory provides the instance of Session. It is a factory of Session. It holds the data of second level cache that is not enabled by default.

6) Is SessionFactory a thread-safe object?
Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

7) What is Session?
It maintains a connection between the hibernate application and database.

It provides methods to store, update, delete or fetch data from the database such as persist(), update(), delete(), load(), get() etc.

It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.

8) Is Session a thread-safe object?
No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

Beans

1.What components does a Spring application have?

Answer: A typical Spring application can be subdivided into the following components:

Bean Class – Contains properties, functions, setter and getter methods, et cetera

Bean Configuration File – Contains information on classes as well as how to configure the same

Interface – Defines the functions

Spring Aspect Oriented Programming – Provides the functionality of cross-cutting concerns

User Program – Uses the function

2: What do you understand by the Spring IoC Container? Explain their types.

Answer: The Spring IoC container lies at the core of the Spring Framework. The container uses Dependency Injection for managing various Spring application components.

The IoC container is responsible for creating the objects, configuring them, wiring them together, and managing their lifecycle. The container receives instructions about the same from the provided configuration metadata.

Means for providing the configuration metadata can include Java annotations, Java code, or XML. There are two types of IoC containers in Spring:

ApplicationContext – Provides additional functionality. It is built on top of the BeanFactory interface.

BeanFactory – A prepackaged class containing a collection of beans. Instantiates the bean whenever as required by the clients

3: Please explain the Dependency Injection in Spring. In how many ways can the same be used?

Answer: Instead of creating objects directly, Dependency Injection allows defining how objects should be created. As such, the code doesn't directly contain connecting components and services together.

The configuration file has the information on which services are needed by which components. The IoC container is responsible for connecting components with the appropriate services. Dependency Injection can be used in the following forms:

Construction Injection

Setter Injection

4: Can you differentiate between ApplicationContext and BeanFactory in Spring?

Answer:

Annotation Based Dependency – BeanFactory doesn't support annotation-based dependency while ApplicationContext does

Interface Definition – BeanFactory interface is defined in org.springframework.beans.factory.BeanFactory while the ApplicationContext interface is defined in org.springframework.context.ApplicationContext

Internationalization Support – While ApplicationContext supports internationalization, BeanFactory doesn't

Object Management – BeanFactory uses syntax for providing a resource object. Contrarily, ApplicationContext creates as well as manages resource objects on its own

Type of Initialization – ApplicationContext makes use of eager or aggressive initialization. On the other hand, BeanFactory uses lazy initialization

5: What do you understand by Spring Beans? How many bean scopes are supported by Spring Framework?

Answer: Configured, instantiated, managed, and wired by the Spring IoC container, Spring Beans are the objects that form the core of a Spring application. Spring Beans are created with the configuration metadata supplied to the Spring IoC container.

Spring Framework provides support for a total of 5 scopes:

Global-session* – Provides scope for a bean definition to a Global HTTP-session

Prototype – Provides scope for a single bean definition for having any number of object instances

Request* – Provides scope for a bean definition to an HTTP-request

Session* – Provides scope for the bean definition to a single instance per Spring IoC container

Singleton – Provides scope for the bean definition to a single instance per Spring IoC container * ? Available only when using a web-aware ApplicationContext.

Spring Boot

1.What are the Spring Boot Annotations?

The @RestController is a stereotype annotation. It adds @Controller and @ResponseBody annotations to the class. We need to import org.springframework.web.bind.annotation package in our file, in order to implement it.

2) What is Spring Boot dependency management?

Spring Boot manages dependencies and configuration automatically. You don't need to specify version for any of that dependencies.

Spring Boot upgrades all dependencies automatically when you upgrade Spring Boot.

3) What are the Spring Boot properties?

Spring Boot provides various properties which can be specified inside our project's application.properties file. These properties have default values and you can set that inside the properties file. Properties are used to set values like: server-port number, database connection configuration etc.

4) What are the Spring Boot Starters?

Starters are a set of convenient dependency descriptors which we can include in our application.

Spring Boot provides built-in starters which makes development easier and rapid. For example, if we want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in your project.

5) What is Spring Boot Actuator?

Spring Boot provides actuator to monitor and manage our application. Actuator is a tool which has HTTP endpoints. when application is pushed to production, you can choose to manage and monitor your application using HTTP endpoints.

6) What is thymeleaf?

It is a server side Java template engine for web application. It's main goal is to bring elegant natural templates to your web application.

It can be integrate with Spring Framework and ideal for HTML5 Java web applications.

7) How to use thymeleaf?

In order to use Thymeleaf we must add it into our pom.xml file like:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

8) How to connect Spring Boot to the database using JPA?

Spring Boot provides spring-boot-starter-data-jpa starter to connect Spring application with relational database efficiently. You can use it into project POM (Project Object Model) file.

9) How to connect Spring Boot application to database using JDBC?

Spring Boot provides starter and libraries for connecting to our application with JDBC. Here, we are creating an application which connects with Mysql database. It includes the following steps to create and setup JDBC with Spring Boot.

9) What is @RestController annotation in Spring Boot?

The @RestController is a stereotype annotation. It adds @Controller and @ResponseBody annotations to the class. We need to import org.springframework.web.bind.annotation package in our file, in order to implement it.

10) What is @RequestMapping annotation in Spring Boot?

The @RequestMapping annotation is used to provide routing information. It tells to the Spring that any HTTP request should map to the corresponding method. We need to import org.springframework.web.annotation package in our file.

Spring MVC

1) What is the Spring Framework and why is it so popular?

Answer: Spring Framework is a lightweight framework that is used to develop loosely coupled Java web applications.

It provides an inbuilt container, dependency injection, and MVC architecture. It is very popular as it allows decoupling, reusability, and provides design patterns like singleton, factory to reduce the code length, etc and also removes weak connections and has powerful integration with the third party.

2) What is Spring MVC?

Answer: Spring MVC follows the concept of Model, View, and Controller.

It is an instance of a controller that is shared alone and also helps to handle several requests that execute in Inversion of Control containers like interceptors and controllers. The Spring MVC pattern helps to separate the aspects of the application that includes input, business and presentation logic.

3) What are the core features of the Spring Framework?

Answer:

Core features of Spring MVC are:

It is capable of effectively configuring the framework and classes as beans. It also divides the functional roles and responsibilities separately.

It allows the definition of an unlimited controller method which makes the application highly adjustable and flexible.

It provides good customization for handler mapping, binding, view resolution, and validations.

It helps to transfer the model by using a map. It also provides supports for velocity, JSTL, JSP, and the user can customize locale and theme resolution.

Spring has its own tag library which makes it more flexible and supports data binding, themes, beans having life cycle up to HTTP request.

4) Explain the concept of the Dispatcher Servlet.

Answer: Dispatcher Servlet is the main central servlet that handles all the incoming HTTP Request and Responses. It has integration with Spring IOC, and thus it allows to use all the features of Spring.

Once the dispatcher servlet receives a request, it forwards it to handler mapping for getting an appropriate controller, and now the controller will call the correct service method and again it will send it back to the dispatcher servlet.

Again, the servlet sends the request to the view resolver for getting the required view and then it sends the response to the client browser.

6) Explain the front controller class of Spring MVC.

Answer: Front Controller is responsible to handle the entire incoming request of an application. In Spring MVC, dispatcher servlet acts as a front controller and handles the entire incoming requests.

7) Explain the function of @Autowired Annotation.

Answer: The @Autowired annotation is responsible for injecting a bean by its type along with the fields and methods. Thereby helping the Spring framework in resolving and injection of collaborating beans into our bean.

Restful Web Services

1.What are HTTP Status codes?

These are the standard codes that refer to the predefined status of the task at the server. Following are the status codes formats available:

1xx - represents informational responses

2xx - represents successful responses

3xx - represents redirects

4xx - represents client errors

5xx - represents server errors

Most commonly used status codes are:

200 - success/OK

201 - CREATED - used in POST or PUT methods.

304 - NOT MODIFIED - used in conditional GET requests to reduce the bandwidth use of the network. Here, the body of the response sent should be empty.

400 - BAD REQUEST - This can be due to validation errors or missing input data.

401- UNAUTHORIZED - This is returned when there is no valid authentication credentials sent along with the request.

403 - FORBIDDEN - sent when the user does not have access (or is forbidden) to the resource.

404 - NOT FOUND - Resource method is not available.

500 - INTERNAL SERVER ERROR - server threw some exceptions while running the method.

502 - BAD GATEWAY - Server was not able to get the response from another upstream server.

2. What are the HTTP Methods?

HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

GET: This is used for fetching details from the server and is basically a read-only operation.

POST: This method is used for the creation of new resources on the server.

PUT: This method is used to update the old/existing resource on the server or to replace the resource.

DELETE: This method is used to delete the resource on the server.

PATCH: This is used for modifying the resource on the server.

OPTIONS: This fetches the list of supported options of resources present on the server.

The POST, GET, PUT, DELETE corresponds to the create, read, update, delete operations which are most commonly called CRUD Operations.

GET, HEAD, OPTIONS are safe and idempotent methods whereas PUT and DELETE methods are only idempotent. POST and PATCH methods are neither safe nor idempotent.

3. Can you tell the disadvantages of RESTful web services?

The disadvantages are:

As the services follow the idea of statelessness, it is not possible to maintain sessions. (Session simulation responsibility lies on the client-side to pass the session id)

REST does not impose security restrictions inherently. It inherits the security measures of the protocols implementing it. Hence, care must be chosen to implement security measures like integrating SSL/TLS based authentications, etc.

4. Define Messaging in terms of RESTful web services.

The technique of sending a message from the REST client to the REST server in the form of an HTTP request and the server responding back with the response as HTTP Response is called Messaging. The messages contained constitute the data and the metadata about the message.

5) Name the protocol which is used by RESTful web services.

Answer: RESTful web services use a famous web protocol i.e. HTTP protocol. This serves as a medium of data communication between client and server. HTTP standard methods are used to access resources in RESTful web service architecture.

6) Explain the term ‘Addressing’ with respect to RESTful WEB service.

Answer: Just like we require an address with postal code to reach any person, in the same way, ‘Addressing’ locates resources that are present on the server for the purpose of hosting web services. This is usually done with URI i.e. Unified Resource Identifier.

7) Enlist features of RESTful web services.

Answer: Every RESTful web services should have the following features and characteristics that are enlisted below:

Based on the Client-Server representation.

Use of HTTP protocol for performing functions like fetching data from the web service, retrieving resources, execution of any query, etc.

The communication between the server and client is performed through the medium known as 'messaging'.

Addressing of resources available on the server through URIs.

Based on the concept of statelessness where every client request and the response is independent of the other with complete assurance of providing required information.

Uses the concept of caching.

Works on the Uniform interface.

8) Explain messaging technique.

Answer: Messages are the mode of exchanging data for any type of communication to take place. In the same way, HTTP protocol plays the role of message communication between the client and server through HTTP Request and Response methods. HTTP request is sent by the client who contains information about the data and in turn, receives HTTP Response from the server.

Messages are the collection of information about the data i.e. Metadata.

9) What are the core components of the HTTP request and HTTP response?

Answer: The core components under HTTP Request are:

Verb: Includes methods like GET, PUT, POST, etc.

Uniform Resource Identifier for identifying the resources available on the server.

HTTP Version for specifying the HTTP version.

HTTP Request header for containing the information about the data.

HTTP Request body that contains the representation of the resources in use.

The core components under HTTP Response are:

Request Code: This contains various codes that determine the status of the server response.

HTTP Version for specifying the HTTP version.

HTTP Response header for containing the information about the data.

HTTP Response body that contains the representation of the resources in use.

10) Explain the term ‘Statelessness’ with respect to RESTful WEB service.

Answer: In REST, ST itself defines State Transfer and Statelessness means complete isolation. This means, the state of the client’s application is never stored on the server and is passed on.

In this process, the clients send all the information that is required for the server to fulfill the HTTP request that has been sent. Thus every client requests and the response is independent of the other with complete assurance of providing the required information.

Every client passes a ‘session identifier’ which also acts as an identifier for each session.

JDBC

1)What are the JDBC API components?

The java.sql package contains following interfaces and classes for JDBC API.

Interfaces:

- **Connection:** The Connection object is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.

- **Statement:** The Statement object is created by using createStatement() method of Connection class. The Connection interface is the factory for Statement.

- **PreparedStatement:** The PrepareStatement object is created by using prepareStatement() method of Connection class. It is used to execute the parameterized query.

- **ResultSet:** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points before the first row. The executeQuery() method of Statement interface returns the ResultSet object.

- **ResultSetMetaData:** The object of ResultSetMetaData interface cotains the information about the data (table) such as numer of columns, column name, column type, etc. The getMetaData() method of ResultSet returns the object of ResultSetMetaData.

- **DatabaseMetaData:** DatabaseMetaData interface provides methods to get metadata of a database such as the database product name, database product version, driver name, name of the total number of tables, the name of the total number of views, etc. The getMetaData() method of Connection interface returns the object of DatabaseMetaData.

- **CallableStatement:** CallableStatement interface is used to call the stored procedures and functions. We can have business logic on the database through the use of stored procedures and functions that will make the performance better because these are precompiled. The prepareCall() method of Connection interface returns the instance of CallableStatement.

Classes:

- **DriverManager:** The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.

- **Blob:** Blob stands for the binary large object. It represents a collection of binary data stored as a single entity in the database management system.

- **Clob:** Clob stands for Character large object. It is a data type that is used by various database management systems to store character files. It is similar to Blob except for the difference that BLOB represent binary data such as images, audio and video files, etc. whereas Clob represents character stream data such as character files, etc.

- **SQLException** It is an Exception class which provides information on database access errors.

---

2) What are the JDBC statements?

In JDBC, Statements are used to send SQL commands to the database and receive data from the database. There are various methods provided by JDBC statements such as execute(), executeUpdate(), executeQuery, etc. which helps you to interact with the database.

There is three type of JDBC statements given in the following table.

Statements	Explanation
Statement	Statement is the factory for resultset. It is used for general purpose access to the database. It executes a static SQL query at runtime.
PreparedStatement	The PreparedStatement is used when we need to provide input parameters to the query at runtime.
CallableStatement	CallableStatement is used when we need to access the database stored procedures. It can also accept runtime parameters.

---

3) What is the return type of Class.forName() method?

The Class.forName() method returns the object of java.lang.Class object.

4) What are the differences between Statement and PreparedStatement interface?

Statement	PreparedStatement
The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet; i.e., it provides the factory method to get the object of ResultSet.	The PreparedStatement interface is a subinterface of Statement. It is used to execute the parameterized query.
In the case of Statement, the query is compiled each time we run the program.	In the case of PreparedStatement, the query is compiled only once.
The Statement is mainly used in the case when we need to run the static query at runtime.	PreparedStatement is used when we need to provide input parameters to the query at runtime.

---

5) How can we set null value in JDBC PreparedStatement?

By using setNull() method of PreparedStatement interface, we can set the null value to an index. The syntax of the method is given below.

6) What are the benefits of PreparedStatement over Statement?

The benefits of using PreparedStatement over Statement interface is given below.

- The PreparedStatement performs faster as compare to Statement because the Statement needs to be compiled everytime we run the code whereas the PreparedStatement compiled once and then execute only on runtime.
- PreparedStatement can execute Parameterized query whereas Statement can only run static queries.
- The query used in PreparedStatement is appeared to be similar every time. Therefore, the database can reuse the previous access plan whereas, Statement inline the parameters into the String, therefore, the query doesn't appear to be same everytime which prevents cache reusage.

---

7) What are the differences between execute, executeQuery, and executeUpdate?

execute	executeQuery	executeUpdate
The execute method can be used for any SQL statements(Select and Update both).	The executeQuery method can be used only with the select statement.	The executeUpdate method can be used to update/delete/insert operations in the database.
The execute method returns a boolean type value where true indicates that the ResultSet s returned which can later be extracted and false indicates that the integer or void value is returned.	The executeQuery() method returns a ResultSet object which contains the data retrieved by the select statement.	The executeUpdate() method returns an integer value representing the number of records affected where 0 indicates that query returns nothing.

---

8) What are the different types of ResultSet?

ResultSet is categorized by the direction of the reading head and sensitivity or insensitivity of the result provided by it. There are three general types of ResultSet.

Type	Description
ResultSet.TYPE_Forward_ONLY	The cursor can move in the forward direction only.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can move in both the direction (forward and backward). The ResultSet is not sensitive to the changes made by the others to the database.
ResultSet.TYPE_SCROLL_SENSITIVE	The cursor can move in both the direction. The ResultSet is sensitive to the changes made by the others to the database.

---

9) What are the differences between ResultSet and RowSet?

ResultSet	RowSet
ResultSet cannot be serialized as it maintains the connection with the database.	RowSet is disconnected from the database and can be serialized.
ResultSet object is not a JavaBean object	ResultSet Object is a JavaBean object.
ResultSet is returned by the executeQuery() method of Statement Interface.	Rowset Interface extends ResultSet Interface and returned by calling the RowSetProvider.newFactory().createJdbcRowSet() method.
ResultSet object is non-scrollable and non-updatable by default.	

Module Name: - Data Structure

1) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.

The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

2) Differentiate linear from a nonlinear data structure.

The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

3) Are linked lists considered linear or non-linear data structures?

It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

4) Differentiate NULL and VOID

Null is a value, whereas Void is a data type identifier. A variable that is given a Null value indicates an empty value. The void is used to identify pointers as having no initial size.

5) What is the advantage of the heap over a stack?

The heap is more flexible than the stack. That’s because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

Module Name: - SE

1) Describe the software development process in brief:

The software development is a life cycle is composed of the following stages:

Requirement analysis

Specification

Software architecture

Implementation

Testing

Documentation

Training and support

Maintenance

2) What is feasibility study?

It is a measure to find out how practical and beneficial the software project development will prove to the organization. The software analyzer conducts a study to know the economic, technical and operational feasibility of the project.

Economic: It includes the cost of training, cost of additional and tools and overall estimation of costs and benefits of the project.

Technical: It evaluate technical aspect. Is it possible to develop this system? Assessing the suitability of machine(s) and OS on which software will execute, knowledge of the software development and tools available for this project.

Operational: Here the analyst need to assess that the organization will able to adjust smoothly to the changes done as per the demand for the project. Is the problem worth solving at the estimated cost?

After, studying all this the final feasibility report is created.

3) What is a Git repository?

Git repository refers to a place where all the Git files are stored. These files can either be stored on the local repository or on the remote repository.

4) Name a few Git commands with their function.

Git config - Configure the username and email address

Git add - Add one or more files to the staging area

Git diff - View the changes made to the file

Git init - Initialize an empty Git repository

Git commit - Commit changes to head but not to the remote repository

5) What is Cohesion and Coupling?

Cohesion indicates the relative functional capacity of the module. Aggregation modules need to interact less with other sections of other parts of the program to perform a single task. It can be said that only one coagulation module (ideally) needs to be run. Cohesion is a measurement of the functional strength of a module. A module with high cohesion and low coupling is functionally independent of other modules. Here, functional independence means that a cohesive module performs a single operation or function. The coupling means the overall association between the modules.

Coupling relies on the information delivered through the interface with the complexity of the interface between the modules in which the reference to the section or module was created. High coupling support Low coupling modules assume that there are virtually no other modules. It is exceptionally relevant when both modules exchange a lot of information. The level of coupling between two modules depends on the complexity of the interface.

Module Name: - ASP.Net

1) Explain how HTTP protocol works?

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It handles communication between web browsers and web servers. HTTP follows a classical client-server model. A client, such as a web browser, opens a connection to make a request, then waits until it receives a response from the server.

HTTP is a protocol that allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web, and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.

2) What is NuGet package manager?

Software developers don't write all code from scratch. They rely on libraries of code written by other developers. Any modern development platform must provide a mechanism where developers can download and use existing libraries, often called packages. For example, the JavaScript ecosystem has NPM (Node Package Manager), where developers can find and use libraries written by other JavaScript developers.

NuGet is a package manager for the .NET ecosystem. Microsoft developed it to provide access to thousands of packages written by .NET developers. You can also use it to share the code you wrote with others.

A typical web application developed using ASP.NET relies on many open source NuGet packages to function. For example, Newtonsoft.Json is a very popular package (with 91,528,205 downloads at the time of writing) used to work with JSON data in .NET.

3) What is IIS?

IIS stands for Internet Information Services. It is a powerful web server developed by Microsoft. IIS can also act as a load balancer to distribute incoming HTTP requests to different application servers to allow high reliability and scalability.

It can also act as a reverse proxy, i.e. accept a client's request, forward it to an application server, and return the client's response. A reverse proxy improves the security, reliability, and performance of your application.

A limitation of IIS is that it only runs on Windows. However, it is very configurable. You can configure it to suit your application's specific needs.

4) What is an Action Method?

An action method is a method in a controller class with the following restrictions:

It must be public. Private or protected methods are not allowed.
It cannot be overloaded.
It cannot be a static method.
An action method executes an action in response to an HTTP request.

For example, here is an example of an Index() action method on the PostController. It takes an id as an input and returns an IActionResult, which can be implemented by any result classes (see the following question).

```
public class PostController : Controller
{
  public IActionResult Index(int id)
  {

  }
}
```

5) What is dependency injection?

Dependency injection is a design pattern that helps to develop loosely coupled code. This pattern is used extensively in ASP.NET.

Dependency injection means providing the objects that an object needs (its dependencies) in that object's constructor instead of requiring the object to construct them.

Dependency injection reduces and often eliminates unnecessary dependencies between objects that don't need to know each other. It also helps in testing by mocking or stubbing out the dependencies at runtime.

Module Name: - MongoDB

1) What do you understand by NoSQL databases? Is MongoDB a NoSQL database? Explain.

At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems, It is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

Yes. MongoDB is a NoSQL database.

2) Can you achieve primary key - foreign key relationships in MongoDB?

We can achieve primary key-foreign key relationship by embedding one document inside another. For example: An address document can be embedded inside customer document.

3) What is sharding in MongoDB?

In MongoDB, Sharding is a procedure of storing data records across multiple machines. It is a MongoDB approach to meet the demands of data growth. It creates horizontal partition of data in a database or search engine. Each partition is referred as shard or database shard.

4)What will happen when you remove a document from database in MongoDB? Does MongoDB remove it from disk?

Yes. If you remove a document from database, MongoDB will remove it from disk too.

5) What are some of the advantages of MongoDB?

Some advantages of MongoDB are as follows:

MongoDB supports field, range-based, string pattern matching type queries. for searching the data in the database

MongoDB support primary and secondary index on any fields

MongoDB basically uses JavaScript objects in place of procedures

MongoDB uses a dynamic database schema

MongoDB is very easy to scale up or down

MongoDB has inbuilt support for data partitioning (Sharding).

Module Name: - Node JS

1) What is package.json?

The package.json file in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

2) What is an Event loop in Node.js and how does it work?

An event loop in Node.js handles all the asynchronous callbacks in an application. It is one of the most important aspects of Node.js and the reason behind Node.js have non-blocking I/O. Since Node.js is an event-driven language, you can easily attach a listener to an event

and then when the event occurs the callback will be executed by the specific listener. Whenever functions like setTimeout, http.get, and fs.readFile are called, Node.js executed the event loop and then proceeds with the further code without waiting for the output. Once the entire operation is finished, Node.js receives the output and then executes the callback function. This is why all the callback functions are placed in a queue in a loop. Once the response is received, they are executed one by one.

3)Explain the purpose of module.exports?

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file.

4) Explain the concept of URL module.

The URL module of Node.js provides various utilities for URL resolution and parsing. It is a built-in module that helps in splitting up the web address into a readable format:

```
var url = require('url');
```

5) What do you understand by global objects in Node.js?

In Node.js, Globals are the objects which are global in nature and are available in all the modules of the application. You can use these objects directly in your application, rather than having to include them explicitly. The global objects can be modules, functions, strings, object, etc. Moreover, some of these objects can be in the module scope instead of global scope.

Module Name: - Express JS

1) What is the difference between Express.js and Node.js?

Node.js is an open-source, cross-platform run-time environment used for executing JavaScript code outside of a browser. Node.js is not a framework or a programming language; it is a platform that acts as a web server. Many big companies such as Paypal, Uber, Netflix, Wallmart, etc., are using this. On the other hand, Express is a small framework based on the functionality of Node.js.

2) What do you understand by Scaffolding in Express.js?

Scaffolding is a technique used for creating the skeleton structure of an application. It facilitates users to easily create their public directories, routes, views, etc., or a web application skeleton. Generally, users manually create their public directory, add middleware, create separate route files, etc. Using a scaffolding tool, they can set up all these things to directly get started with building their application.

There are two ways to install Scaffolding and use it in your application.

Express application generator

Yeoman

3) Which are the arguments available to an Express JS route handler function?

Following are the arguments that are available to an Express.js route handler-function:

Req: the request object

Res: the response object

Next (optional): It is a function employed to pass management to one of the above route handlers.

4) Which template engines do Express support?

Express.js supports any template engine that follows the (path, locals, callback) signature.

5) What is the use of next in Express JS?

Answer:

Next -It passes management to a consecutive matching route. OR a operate to pass management to 1 of the following route handlers.

The argument could also be omitted, however, is beneficial in cases wherever you have got a series of handlers and you'd wish to pass management to 1 of the following route handlers, and skip this one.

app.get('/user details/:id?', function(req, res, next));

Req and Res: It represents the request and response objects

Next: It passes management to a consecutive matching route.

Module Name: - React JS

1) What is useState() in React?

The useState() is a built-in React Hook that allows you for having state variables in functional components. It should be used when the DOM has something that is dynamically manipulating/controlling.

In the below-given example code, The useState(0) will return a tuple where the count is the first parameter that represents the counter's current state and the second parameter setCounter method will allow us to update the state of the counter.

```
...
const [count, setCounter] = useState(0);
const [otherStuffs, setOtherStuffs] = useState(...);
...
const setCount = () => {
  setCounter(count + 1);
  setOtherStuffs(...);
  ...
};
```

We can make use of setCounter() method for updating the state of count anywhere. In this example, we are using setCounter() inside the setCount function where various other things can also be done. The idea with the usage of hooks is that we will be able to keep our code more functional and avoid class-based components if they are not required.

2) What is JSX?

JSX stands for JavaScript XML. It allows us to write HTML inside JavaScript and place them in the DOM without using functions like appendChild( ) or createElement( ).

As stated in the official docs of React, JSX provides syntactic sugar for React.createElement( ) function.

Note- We can create react applications without using JSX as well.

Let's understand how JSX works:

Without using JSX, we would have to create an element by the following process:

```
const text = React.createElement('p', {}, 'This is a text');

const container = React.createElement('div','{}',text );

ReactDOM.render(container,rootElement);
```

Using JSX, the above code can be simplified:

```
const container = (

<div>

  <p>This is a text</p>

</div>

);

ReactDOM.render(container,rootElement);
```

As one can see in the code above, we are directly using HTML inside JavaScript.

3) What are the differences between functional and class components?

Before the introduction of Hooks in React, functional components were called stateless components and were behind class components on a feature basis. After the introduction of Hooks, functional components are equivalent to class components.

Although functional components are the new trend, the react team insists on keeping class components in React. Therefore, it is important to know how these components differ.

On the following basis let's compare functional and class components:

Declaration

Functional components are nothing but JavaScript functions and therefore can be declared using an arrow function or the function keyword:

```
 function card(props){

  return(

    <div className="main-container">

      <h2>Title of the card</h2>
```

```
    </div>
  )
 }
 const card = (props) =>{
  return(
    <div className="main-container">
      <h2>Title of the card</h2>
    </div>
  )
 }
```

Class components, on the other hand, are declared using the ES6 class:

```
class Card extends React.Component{
 constructor(props){
   super(props);
 }
  render(){
    return(
      <div className="main-container">
        <h2>Title of the card</h2>
      </div>
    )
  }
 }
```

4) What are props in React?

The props in React are the inputs to a component of React. They can be single-valued or objects having a set of values that will be passed to components of React during creation by using a naming convention that almost looks similar to HTML-tag attributes. We can say that props are the data passed from a parent component into a child component.

The main purpose of props is to provide different component functionalities such as:

Passing custom data to the React component.

Using through this.props.reactProp inside render() method of the component.

Triggering state changes.

For example, consider we are creating an element with reactProp property as given below:
<Element reactProp = "1" />

This reactProp name will be considered as a property attached to the native props object of React which already exists on each component created with the help of React library:
props.reactProp;

5) What is React Hooks?

React Hooks are the built-in functions that permit developers for using the state and lifecycle methods within React components. These are newly added features made available in React 16.8 version. Each lifecycle of a component is having 3 phases which include mount, unmount, and update. Along with that, components have properties and states. Hooks will allow using these methods by developers for improving the reuse of code with higher flexibility navigating the component tree.

Using Hook, all features of React can be used without writing class components. For example, before React version 16.8, it required a class component for managing the state of a component. But now using the useState hook, we can keep the state in a functional component.

IMPORTANT------JAVA 8----------

1)What New Features Were Added in Java 8?

Java 8 ships with several new features, but the most significant are the following:

Lambda Expressions − a new language feature allowing us to treat actions as objects

Method References − enable us to define Lambda Expressions by referring to methods directly using their names

Optional − special wrapper class used for expressing optionality

Functional Interface – an interface with maximum one abstract method; implementation can be provided using a Lambda Expression

Default methods − give us the ability to add full implementations in interfaces besides abstract methods

Nashorn, JavaScript Engine − Java-based engine for executing and evaluating JavaScript code

Stream API − a special iterator class that allows us to process collections of objects in a functional manner

Date API − an improved, immutable JodaTime-inspired Date API

2)What Is Optional? How Can It Be Used?

Optional is a new class in Java 8 that encapsulates an optional value, i.e. a value that is either there or not. It's a wrapper around an object, and we can think of it as a container of zero or one element.

Optional has a special Optional.empty() value instead of wrapped null. Thus it can be used instead of a nullable value to get rid of NullPointerException in many cases.

3)What Is a Functional Interface? What Are the Rules of Defining a Functional Interface?

A functional interface is an interface with one single abstract method (*default* methods do not count), no more, no less.

Where an instance of such an interface is required, a Lambda Expression can be used instead. More formally put: *Functional interfaces* provide target types for lambda expressions and method references.

The arguments and return type of such an expression directly match those of the single abstract method.

4) What Is a Lambda Expression and What Is It Used For?

In very simple terms, a lambda expression is a function that we can reference and pass around as an object.

Moreover, lambda expressions introduce functional style processing in Java, and facilitate the writing of compact and easy-to-read code.

As a result, lambda expressions are a natural replacement for anonymous classes such as method arguments. One of their main uses is to define inline implementations of functional interfaces.

5) What Is a Stream? How Does It Differ From a Collection?

In simple terms, a stream is an iterator whose role is to accept a set of actions to apply on each of the elements it contains.

The stream represents a sequence of objects from a source such as a collection, which supports aggregate operations. They were designed to make collection processing simple and concise.

Contrary to the collections, the logic of iteration is implemented inside the stream, so we can use methods like map and flatMap for performing a declarative processing.

Additionally, the Stream API is fluent and allows pipelining:

Git

What is a git repository?

A repository is a file structure where git stores all the project-based files. Git can either stores the files on the local or the remote repository.

2. What does git clone do?

The command creates a copy (or clone) of an existing git repository. Generally, it is used to get a copy of the remote repository to the local repository.

4. What does the command git config do?

The git config command is a convenient way to set configuration options for defining the behavior of the repository, user information and preferences, git installation-based configurations, and many such things.

For example:

To set up your name and email address before using git commands, we can run the below commands:

git config --global

user.name

“<<your_name>>”

git config --global user.email “<<your_email>>”

5. What is a conflict?

Git usually handles feature merges automatically but sometimes while working in a team environment, there might be cases of conflicts such as:

1. When two separate branches have changes to the same line in a file

2. A file is deleted in one branch but has been modified in the other.

These conflicts have to be solved manually after discussion with the team as git will not be able to predict what and whose changes have to be given precedence.

How will you create a git repository?

Have git installed in your system.

Then in order to create a git repository, create a folder for the project and then run git init.

Doing this will create a .git file in the project folder which indicates that the repository has been created.

6. Tell me something about git stash?

Git stash can be used in cases where we need to switch in between branches and at the same time not wanting to lose edits in the current branch. Running the git stash command basically pushes the current working directory state and index to the stack for future use and thereby providing a clean working directory for other tasks.

7. What is the command used to delete a branch?

To delete a branch we can simply use the command git branch –d [head].

To delete a branch locally, we can simply run the command: git branch -d <local_branch_name>

To delete a branch remotely, run the command: git push origin --delete <remote_branch_name>

Deleting a branching scenario occurs for multiple reasons. One such reason is to get rid of the feature branches once it has been merged into the development branch.

8. What differentiates between the commands git remote and git clone?

git remote command creates an entry in git config that specifies a name for a particular URL. Whereas git clone creates a new git repository by copying an existing one located at the URL.

9. What does git stash apply command do?

git stash apply command is used for bringing the works back to the working directory from the stack where the changes were stashed using git stash command.

This helps the developers to resume their work where they had last left their work before switching to other branches.