**Corejava**

1.Reverse a String

String builder,charAt()

**Strings**

1. **How to compare two Strings in java program?**

**operator“==”**

"==" or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. "==" compare two objects based on memory reference. so "==" operator will return true only if two object reference it is comparing represent exactly same object otherwise "==" will return false.

**operator“equals”**

equals()method compare two Strings for content equality. So if two string contains same letters, in same order and in same case they will be equals by equals() method. equals() method is defined in Object class and String class overrides that for character based comparison.

**publicstaticvoid** main(String[] args) {

String s1="swapna";

String s2="swapna";

String s3=**new** String("swapna");//it will create new object

System.***out***.println("s1==s2::: "+(s1==s2));

System.***out***.println("s1==s3:::"+(s1==s3));

System.***out***.println("s1.equals(s2)::: "+s1.equals(s2));

System.***out***.println("s1.equals(s3):::"+s1.equals(s3));

}

}

**Output::**

**s1==s2::: true**

**s1==s3:::false**

**s1.equals(s2)::: true**

**s1.equals(s3):::true**

1. **Why String is immutable or final in Java?**

The string is Immutable in Java because String objects are cached in String pool. Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client.

1. **Does String is thread-safe in Java?**

Strings are immutable, so we can’t change it’s value in program. Hence it’s thread-safe and can be safely used in multi-threaded environment.

1. **What is String Pool?**

String Pool is a pool of Strings stored in Java heap memory. We know that String is special class in java and we can create String object using new operator as well as providing values in double quotes.

**Oopsconcepts**

1. Why Java doesn’t support multiple inheritance?

Java doesn’t support multiple inheritance in classes because it can lead to **diamond problem** and rather than providing some complex way to solve it, This leads to the ambiguity as compiler doesn’t know which superclass method to execute.

Other way::  multiple inheritance is not supported in classes but it’s supported in interfaces. A single interface can extend multiple interfaces,This is perfectly fine because the interfaces are only declaring the methods and the actual implementation will be done by concrete classes implementing the interfaces. So there is no possibility of any kind of ambiguity in multiple inheritance in java interfaces.

1. On which version you worked on java?

1.7 or 1.8

1. Tell me some of the features of java 8?

**1.Interface changes with default and static methods**

interfaces are enhanced to have method with implementation. We can use default and static keyword to create interfaces with method implementation.

**2.Functional interfaces and Lambda Expressions**

**3.Java Stream API for collection classes**

A new java.util.stream has been added in Java 8 to perform filter/map/reduce like operations with the collection. Stream API will allow sequential as well as parallel execution. This is one of the best feature for me because I work a lot with Collections and usually with Big Data.

**4.Java Date Time API**

1. What is the difference between stack and heap in Java?

Stack and heap are different memory areas in the JVM and they are used for different purposes. The stack is used to hold method frames and local variables while objects are always allocated memory from the heap. The stack is usually much smaller than heap memory and also didn't shared between multiple threads, but heap is shared among all threads in JVM.

1. What is a.hashCode() used for? How is it related to a.equals(b)?

hashCode() method returns an int hash value corresponding to an object. It's used in hash based collection classes e.g Hashtable, HashMap, LinkedHashMap and so on. It's very tightly related to equals() method. According to Java specification, two objects which are equal to each other using equals() method must have same hash code.

1. What is the difference between Hashtable and HashMap?

Hashtable is a legacy class and present from JDK 1, HashMap was added later.

Hashtable is synchronized and slower but HashMap is not synchronized and faster.

Hashtable doesn't allow null keys but HashMap allows one null key.

For synchronization we can use ConcurrentHashMap.

1. The difference between checked and unchecked Exception in Java?

checked exception is checked by the compiler at compile time. It's mandatory for a method to either handle the checked exception or declare them in their throws clause. These are the ones which are a sub class of Exception but doesn't descend from RuntimeException. The unchecked exception is the descendant of RuntimeException and not checked by the compiler at compile time.

Checked Exceptions-

FileNotFoundException ,

IOException

ClassNotFoundException

UncheckedExceptions-

NullPointerException

ArrayIndexOutOfBoundsException

ArithmeticException

IllegalArgumentException

NumberFormatException

1. What is the difference between Comparator and Comparable in Java?r

**Comparable-java.lang.package**

Comparable interface for default natural Sorting order .it contains one method compareTo().

//Tree set-Sorting order provide by jvm, it take default natural sorting order

**Comparator-java.util**

Comparator interface for Customised sorting order.it contains two methods compare and equals().

Comparator have more preference than Comparable.

**Comparator>Comparable.**

1. how to connect java code with mysql database?

Jdbc

Hibernate.xml

Spring.xml

1. what is singleton class in java?

singleton class is a class that can have only one object (an instance of the class) at a time.  
After 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, it affects the variable of the single instance created.

**Implementation**

1.Make constructor as private.

2.Write a static method that has return type object of this singleton class

**Spring**

1.what are the features of Spring Framework?

Dependency Injection or Inversion of Control to write components that are independent of each other, spring container takes care of wiring them together to achieve our work.

Spring IoC container manages Spring Bean life cycle and project specific configurations

2.What is Deependency Injection?

Java Dependency Injection design pattern allows us to remove the hard-coded dependencies and make our application loosely coupled, extendable and maintainable. We can implement dependency injection in java to move the dependency resolution from compile-time to runtime.

1.Setter and Getter methods

2.Constructor

3.What is Inversion of Control?

Inversion of Control which the control of objects or portions of a program is transferred to a container or framework.

Inversion of Control (IoC) is the mechanism to achieve loose-coupling between Objects dependencies

4.What is spring IOC Container?

To achieve loose coupling and dynamic binding of the objects at runtime, the objects define their dependencies that are being injected by other assembler objects. Spring IoC container is the program that injects dependencies into an object and make it ready for our use.

The Spring container is responsible for instantiating, configuring and assembling objects known as beans, as well as managing their lifecycle.

5. What is the importance of Spring bean configuration file?

We use Spring Bean configuration file to define all the beans that will be initialized by Spring Context. When we create the instance of Spring ApplicationContext, it reads the spring bean xml file and initialize all of them. Once the context is initialized, we can use it to get different bean instances.

6. What are different ways to configure a class as Spring Bean?

There are three different ways to configure Spring Bean.

**XML Configuration**: This is the most popular configuration and we can use bean element in context file to configure a Spring Bean. For example:

<bean name="myBean" class="com.journaldev.spring.beans.MyBean"></bean>

**Java Based Configuration**: If you are using only annotations, you can configure a Spring bean using @Bean annotation. This annotation is used with @Configuration classes to configure a spring bean. Sample configuration is:

@Configuration

@ComponentScan(value="com.journaldev.spring.main")

public class MyConfiguration {

@Bean

public MyService getService(){

return new MyService();

}

}

To get this bean from spring context, we need to use following code snippet:

AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(

MyConfiguration.class);

MyService service = ctx.getBean(MyService.class);

**Annotation Based Configuration:** We can also use @Component, @Service, @Repository and @Controller annotations with classes to configure them to be as spring bean. For these, we would need to provide base package location to scan for these classes. For example:

Copy

<context:component-scan base-package="com.journaldev.spring" />

7. What are different scopes of Spring Bean?

There are five scopes defined for Spring Beans.

singleton: Only one instance of the bean will be created for each container. This is the default scope for the spring beans..

prototype: A new instance will be created every time the bean is requested.

request: This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.

session: A new bean will be created for each HTTP session by the container.

global-session: This is used to create global session beans for Portlet applications

To set spring bean scopes we can use “scope” attribute in bean element or @Scope annotation for annotation based configurations.

**Hibernate**

1. What is hibernate?

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate, and retrieve data from the database.

2. What is SessionFactory?

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

SessionFactory is an immutable thread-safe cache of compiled mappings for a single .

3. What is Session?

Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.

Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object.

4. What is difference between Hibernate Session get() and load() method?

|  |  |
| --- | --- |
| get() | Load() |
| Returns null if an object is not found. | Throws ObjectNotFoundException if an object is not found. |
| get() method always hit the database | load() method doesn't hit the database. |
| It returns the real object | It returns proxy object. |
| It should be used if you are not sure about the existence of instance. | It should be used if you are sure that instance exists. |

5.