1. How many ways to create Objects in Java?
2. Using new keyword: - constructor gets called. This is the most common way to create an object in java.
3. Using class.forName():- constructor gets called. If we know the name of the class & if it has a public default constructor, we can create an object in this way.
4. Using clone (): - no constructor call. The clone () can be used to create a copy of an existing object.
5. Using object deserialization: - no constructor call. Object deserialization is nothing but creating an object from its serialized form.
6. What is constructor? Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation. generally, constructors for initializing no static members of particular class with respect to object.
7. What is static method? A static method belongs to class rather than object of class. A static method can be invoked without the need for creating an instance of a class. Static methods can access static data member and can change the value of it.
8. Can we execute a program without main () method? Yes, one of the way is static block and it is executed before main method at the time of class loading.
9. Concept of string constant pool and sting literals?

New string(“test”): - explicitly creates a new and referentially distinct instance of a sting object.

Sting s= “text” may reuse an instance from the sting constant pool if one is available. (sting literals). Total objects: - heap -2 pool -1

1. Where is constant pool memory allocated or heap memory concept? All kinds of object will be created in heap only. Heap memory again divided into 3 portions: -
2. Young generation: - stores objects which have a short life, young generation itself can be divided into two categories Eden space and survivor space.
3. Old generation: - stores object which have survived many garbage collection cycle and still being referenced.
4. Permanent generation: - stores metadata about the program e.g. runtime constant pool string constant pool belongs to permanent generation area of heap memory.
5. What is covariant return type? It is possible to override method by changing the return type if child class overrides any method whose return type is non-primitive, but it changes its return type to child class type. OR if we change the return type of override method that is called covariant return type. If overridden method returns own class reference.

|  |  |
| --- | --- |
| Class A{  A get(){ return this;}  } | Class b1 extends a{  B1 get(){ return this;}  } |

1. What is blank or uninitialized final variable and hen is initialize? A final variable that is not initialized at the time of declaration is know as blank final variable. If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed. It can be initialized only in constructor.
2. What is static blank final variable and when is initialize? A static final variable that is not initialized at the time of declaration is know as static blank final variable. it can be initialized only in static block.
3. Polymorphism and types of polymorphism? Polymorphism in which we can perform a single action /task by different ways.
4. Compile time: - if you overload static method in java, it is example of compile time polymorphism.
5. Run time: - in this process, an overridden method is called through the reference variable of a superclass.
6. What is difference between static binding and dynamic binding? In the static binding type of object is determined at compile times whereas in dynamic binding types of object is determined at runtime.
7. What is upcasting? When reference variable of parent class refers to the object of child class, it is known as upcasting.

Class A{}

Class B extends A{}

A a = new B(); // upcasting

1. What is abstraction and abstract class? Abstraction is a process of hiding the implementation details and showing only the functionality to the user. A class that is declared as abstract is known as abstract class. It cannot be instantiated. Interface is 100% example of abstraction and abstract class could be 0 to 100%.
2. What is difference between abstraction and encapsulation? Abstraction hides the implementation details whereas encapsulation binding the code and data into a single unit. Java bean is the fully encapsulated class because all data members are private here.
3. What is interface and advantage of interface? An interface in java is a blueprint of class. It has static constants and abstract methods. It is used to achieve abstraction, loose coupling and multiple inheritance in java.
4. Can you declare an interface method, and can interface be final? No, because methods of an interface are abstract by default and static and abstract keywords can’t be used together. No, because its implementation is provided by another class.
5. What is marker interface? As interface that have no data member and method is known as a marker interface. E.g. serializable, cloneable
6. What is difference between abstract class and interface?

|  |  |
| --- | --- |
| An abstract class can have method body (non-abstract methods) | Interface have only abstract methods |
| An abstract class can have instance variable | As interface cannot have instance variable |
| An abstract class can have constructor | Interface cannot have constructor |
| An abstract class can have static method | Interface cannot have static methods |
| You can extend one abstract class | You can implement multiple interfaces. |

1. What is static import, or can we call static member of class without calling it from class name? by static import, we can access the static members of the class directly, there is no to call it with the class name.
2. How to create immutable class in java?
3. Class must be declared as final.
4. Data members in the class must be declared as final
5. A parameterized constructor.
6. Getter method for all the variables in it.
7. No setters.
8. Why string objects are immutable in java? because java uses the concept of string literal. Suppose there are 5 reference variables, all refers to one object “preety”. If one reference variable changes the value of the object, it will be affected to all the reference variables. That’s way string objects are immutable in java.
9. How many ways we can create the sting object? There are 2 ways to create the string object, by string literal and by new keyword. To make java more memory efficient we used string literal concept.
10. Can a class have an interface, and can an interface have a class? Yes, it is known as nested interface. Yes, if we define a class inside the interface, java compiler creates a static nested class.
11. What is garbage collection and gc ()? Garbage collection is a process of destroy the runtime unused objects, it is performed foe memory management. Gc () is a daemon thread and it is used to send request to JVM to perform garbage collection.
12. What is the purpose of finalize () method? Finalize () method is invoked just before the object is garbage collected, it is used to perform cleanup processing.
13. What is difference between final, finally and finalize?

Final: - final is keyword, final can be variable method or class. You can’t change the value of final variable, can’t override final method, can’t inherit the final class.

Finally: - finally block is used in exception handing, finally block is always executed.

Finalize (): - finalize () method is used in garbage collection. Finalize () method is invoked just before the object is garbage collected. The finalize () method can be used to perform any cleanup processing.

1. What is serialization? Serialization is a process of writing the state of an object into a byte stream. It is mainly used to travel object’s state on the network. If you define any data member as transit, then it will not be serialized.
2. What is reflection? Reflection is the process of examining or modifying the runtime behavior of the class at runtime.
3. Can you access the private method from outside the class? Yes, by changing the runtime behavior of the class if the class is not secured.
4. Difference between final and immutable?

Final: - you cannot change the reference of the collection(object). You can modify the collection/object the reference points to. You can still add elements to the collection.

Immutable: - you cannot modify the contents of the collection/object the reference points to. You cannot add elements to the collection

If ArrayList is the final then you can modify the data of the ArrayList like remove () and add () operation will work but initialization is not possible.

1. Diamond operator in java 7?

List list = new ArrayList ();

For (object obj:list){

String name = (String) obj;

System.out.print(name);

}// compile time exception

Generic concept come in java 5:- we don’t need to mention type parameter during generic class instantiation.

List<string> l = new Array<String>();

Diamond operator come in java 7:- for automatic type inference during generic class instantiation, you must specify the diamond operator.

List<Map<String,List<String>> l = new ArrayLlist<>();

1. Java is pass by value? Java is pass by value , we can proof that with swap method in program.

Balloon red = new Balloon(“red”);

Balloon blue = new Balloon(“blue”);

Swap(red,blue);

System.out.println(“red color=”+ red.getColor());// red color = redblue

System.out.println(“blue color=”+ blue.getColor());// blue color = blue

Public static void swap(Object o1 , Object o2){

Object temp = o1;

O1=o2;

O2= temp;

}

1. How substring work internally? String is more like a utility class which works on that character sequence. This character sequence is maintained as array calked value[]. Every time you call substring() method in java.it will return a new string because String is immutable in java.

Before java 7:- when calling substring, a new string is created with the same char[] but a different offset/count, to effectively create a view on the original string.

After java 7:- Instead of sharing original character array, substring method creates a copy of it. New char[] is created every time , because there is no more offset or count field in the string class. Array.copyOfRange is used to create whole new copy of substring.

1. String hashcode caching use in hashmap? When string hashcode calculate first time then it has been store in string object.so next time if string object used it don’t calculate hashcode it’s passed cache hashcode.

Hashmap<string,integer> hm = new hashmap<>();

Hm.put(“aa”,1); //hash method called and code generated

Hm.get(“aa”); // hashcode cached return same hashcode

Hm.get(“aa”); // hashcode cached return same hashcode.

String hashcode cached because string is immutable so object value won’t change, so rather than create object every time we can cache it.

1. If constructor is private, then how we can create object of that class? Calling a static factory method.
2. What is load factor/rehashing in hashmap?
3. Now why should you keep many free buckets & what is the impact of keeping free buckets on the performance & hash collision?
4. RestFul :- when web service use HTTP methods to implement the concept of REST architecture, then it is known as RESTful web services, In this architectural style, data and functionality are served as resources and is accessed by URI(uniform resource identifiers).
5. Explain different HTTP methods supported by RESTful web services?

GET:- read only access to the resource.

PUT:- creation of new resource.

DELETE:- removal of a resource.

POST:- update of an existing resource.

OPTIONS:- get supported operations on the resource.

HEAD:- return HTTP header only, nobody.

1. What is a resources in restful web service and how it is represented? In rest architecture, each content is considered as the resource and they are identified by URIs. Resources are represented with the help of XML,JSON,text etc in restful architecture.
2. What are the core components of HTTP request and HTTP response?

|  |  |
| --- | --- |
| HTTP Requests | Meaning/Work |
| Verb | Indicate http methods like GET , PUT,POST etc. |
| URI | Identifies the resource on server |
| HTTP version | Indicates version |
| Request header | Contains metadata like client type,cache settings,message body format etc for HTTP request message. |
| Request body | Represents content of the message |

|  |  |
| --- | --- |
| HTTP response | Meaning/Work |
| Status/response code | Indicates the status of the server for the requested resource |
| HTTP version | Represents http version |
| Response header | Consists of metadata like content length,content type,server length etc for HTTP response message. |
| Response body | Represents response message content. |

1. How can we do secure rest webservices in java?
2. We can do basic authentication in web.xml file and can specify which URL you want to secure i.e. for which role the URL should be accessible along with http method type.
3. If you are using tomcat as web server. Inside server.xml file add realm.in tomcar-users.xml you can create roles,create users , assign each user – a username,password and role.
4. How many ways to implement restful services?
5. Spring rest with resttemplates api.
6. Jersey(1.9) using jax-rs JDK by default provide
7. RestEasy
8. Can we maintain user session in web services? Web services are stateless so we can’t maintain user sessions in web services.
9. PUT vs POST :- the result generated with PUT method is always same no matter how many times the operation is performed. On the other hand the result generated by POST operation is always different every time.
10. User thread vs daemon thread :- when we create a thread in java program, it’s known as user thread. A daemon thread runs in background and it’s terminated when there are no user threads running . a child thread created from daemon thread is also a daemon thread.
11. Why thread sleep() and yield() methods are static? Thread sleep() and yield() methods work on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state.
12. How can we achieve thread safety in java? synchronization using volatile keyword, using immutable classes concurrent hashmap , atomic concurrent classes , implementing concurrent lock interface.
13. Volatile keyword:- when we use volatile keyword with a variable ,all the threads read it’s value directly from the memory and don’t cache it. this makes sure that the value read is the same as in the memory.
14. How to create daemon thread in java? thread class setDaemon(true) can be used to create daemon thread in java. we need to call this method before calling start() method else it will throw IllegalThreadStateException.
15. What is deadloack? How to analyze and avoid deadlock situation? If 2 threads are waiting for each other forever(without end) such type os situation is called deadlock.

Synchronized keyword is the main cause for deadlock: consider we have two classes A and B and both have synchronized method C and if we invoked C method inside synchronized method of A and B classes.

1. What is ThreadPool and how can you implement your own ThreadPool in java? ThreadPool is a pool of threads which reuses a fixed number of threads to execute tasks. ThreadPool implementation internally used LinkedBlockingQueue for adding and removing tasks. Executor interface to create the thread pool in java.
2. What is callable and future? Callable is similar to runnable interface but it can return any object and able to throw exception. Callable tasks return future object. Using future we can find out the status of the callable task and get the returned object.
3. What is synchronization and advantage? Synchronized is the keyword applicable for methods and blocks but not for classes and variable. If a method or block declared as synchronized the at a time only one thread is allow to execute that method or block on the given object. The main advantage of synchronized keyword is we can resolve date inconsistency problems. Every object in java has unique lock. Whenever we are using synchronized keyword then only lock concept will come into the picture.
4. What is thread communication?

Wait () :- we can use wait method to enter thread in waiting state.

Notify():- we can use notify() method to give notification for only one thread.

notifyAll():- we can use notifyAll() method to give the notification for all waiting threads.

Once a thread calls wait() method on the given object 1st it release the lock of that object immediately and entered into waiting state. Once a thread calls notify() or notifyAll() methods it releases the lock of that object but may not immediately.

1. What is Executors class? Executors class provide utility methods for Executor, ExecutorService, SchecduledExecutorService, ThreadFactory and Callable classes. Executors class can be used to easily create ThreadPool in java, also this is the only class supporting execution of Callable implementations.

Executor:- used to submit a new task.

ExecutorService:- A subinterface of executor that adds method to manage lifecycle of threads used to run the submitted tasks. Passing a callable to the submit method is possible to get a future object and use it to retrieve the result of the asynchronous computation.

SchecduledExecutorService:- a subinterface of ExecutorService , to execute commands periodically or after a given delay.

1. Common method of executor class?
2. An ExecutorService with a single thread to execute commands with method newSingleThreadExecutor.
3. A SchecduledExecutorService with a single thread to execute commands with the method newSingleThread SchecduledExecutor.
4. An ExecutorService that use a fixed length pool of threads to execute commands with the method newFixedThreadPool.
5. An ExecutorService with a pool of threads that creates a new thread if no thread is available and reuse an existing thread if they are available with newCachedThreadPool.
6. A SchecduledExecutorService with fixed length pool of threads to execute scheduled commands with the method newScheduledThreadPool.
7. What is spring? It is light-weighted and loosely coupled, it has layered architecture ,which allows us to select the components to use.
8. What is dependency injection(DI) or inversion of control (IOC)? It’s design pattern based on loose coupling and In that spring container create the object and establish the dependencies. So means we don’t have to create objects but have to describe how they should be created.
9. What is spring IOC container? The container creates the object , wires them together, configures them and manages their complete life cycle.
10. Constructor vs setter injection

|  |  |
| --- | --- |
| Constructor Injection | Setter Injection |
| There is no partial injection | There can be partial injection |
| It doesn’t override the setter property | It overrides the constructor property |
| It will create a new instance if any modification is done | It will not create new instance if any modification is done |
| It works better for many properties | It works better for few properties |

1. How many types of IOC container are there in spring?
2. BeanFactory :- BeanFactory is like a factory class that contains a collection of beans,it instantiates the bean whenever asked for by clients.
3. ApplicationContext:- the ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.
4. What is spring bean? Any normal java class that is initialized by spring IoC container is called spring bean, we use spring ApplicationContext to get the spring bean instance. Spring IoC container manages the lifecycle of spring bean.
5. How configuration metadata is provided to the spring container or what are different ways to configure a class as springbean?
6. XML based configuration

<bean id = “studentbean” class=”com.StudentBean”>

<property name =”name” value = “preety” ></property>

</bean>

1. Annotation based configuration

<beans>

<context:annotation-config/>

</beans>

1. Java Based Configuration

@Configuration

Public class StudentConfig{

@Bean

Public StudentBean myStudent(){

Return new StudentBean():

}

}

1. Bean life cycle in spring bean factory container
2. The spring container instantiates the bean.
3. Spring populates all of the properties using the dependency injection.
4. The factory calls setBeanName() by passing the bean’s Id , if the bean implements the BeanNameAware interface.
5. The factory calls setBeanFactory() by passing an instance of itself, if the bean implements the BeanFactoryAware interface.
6. preProcessBeforeInitialization() methods are called if there are nay BeanPostProcessors associated with the bean.
7. If an init-method is specified for the bean, then it will be called.
8. Finally , postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean.
9. How annotation wiring can be turned on the spring? By default , annotation wiring is not turned on in spring container. <context:annotation-config/>
10. AOP :- Aspect oriented programming or AOP is a programming technique which alloes programmers to modularize crosscutting concerns or behavior that can be logging and transaction management. The core of AOP is an aspect.
11. Joinpoint :- a point during the execution of a program, such as the execution of a method or the handling of an exception.
12. Advice:- is associated with a pointcut expression and runs at any point matched by the pointcut.
13. PointCut:- It’s combination of JoinPoints and it specifies that at which JoinPoint advice will be executed.
14. Types of Advices
15. Before :- these types of advices executed before the joinpoint methods and are configured using @Before annotation.
16. After returning :- these types of advices execute after the joinpoint methods completes executing normally and are configured using @AfterReturning annotation.
17. After throwing :- these types of advice execute only if joinpoint method exits by throwing an exception and are configured using @AfterThrowing annotation.
18. After (Finally) :- these types af advices execute afte a joinpoint method, regardless of the method’s exist whether normally or exceptional return and are configured using @After annotation.
19. Around :- these types of advice execute before and after a joinpoint and are configured using @Around annotation.
20. Proxy in spring framework :- an object which is created after applying advice to a target object is known as a proxy.
21. Weaving :- the process of linking an aspect with other application types or objects to create an advised object is called Weaving. In spring AOP weaving is performed at runtime.
22. WebApplicationContext :- is an extension of the plain ApplicationContext. It has some extra features that are necessary for web applications.
23. What is controller? Controllers provide access to the application behavior. these behaviours are generally defined through a service interface. Controllers interpret the user input and transform it into a model which is represented to the user by the view.
24. How to configure spring security in our application ?
25. Update pom.xml to include required dependencies.
26. Then we need to create a Spring Security configuration file (Java or XML). In this we creates a Servlet Filter known as the springSecurityFilterChain. It is responsible for protecting the application URLs, validating submit username and password, redirecting to the login from etc.
27. Authentication: Http basic authentication and form login based authentication
28. Authorization support through ACL.
29. Logout support
30. Anonymous login support
31. Remember-me authentication
32. Concurrent session management
33. Now , we will register springSecurityFilterChain with war. To register , spring security provides a base class AbstractSecurityWebApplicationInitializer that we need to extend (for java). make filter and filter mapping tag in web.xml
34. Now, load spring security configuration file in our existing applicationInitializer file, and add into the getRootConfigClasses() method (for java). and add initialize it into web.xml
35. Spring security features
36. LDAP(light weight access protocol)
37. Single sign on
38. JAAS( java authentication and authorization service) login module
39. Basic access authentication
40. Digest access authentication
41. Remember me
42. Web form authentication
43. Authorization
44. Software localization
45. Http authorization
46. How to configure delegating filter proxy? In the web.xml we add DelegatingFilterProxy which is delegating proxy to automatically intercept a URL with a particular pattern to apply spring security.

<filter>

<filter-name> springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterproxy</filter-class>

</filter>

<filter-mapping>

<filter-name> springSecurityFilterChain</filter-name>

<url-pattern>\*</url-pattern>

</filter-mapping>

1. Hibernate :- Hibernate is an object – relational mapping(ORM) solution for java and its maps java classes to database tables and from java data types to SQL data types.
2. Session vs SessionFactory :- Sessionfactory is a factory class for session objects. It is available for the whole application while a session is only available for particular transaction. Session is dhort lived while SessionFactory objects are long lived. SessionFactory provides a second level cache and session provides a first level cache.
3. What is session and is it thread safe? It maintains a connection between hibernate application and database. It provides method to store,update,delete or fetch data from the database such as persist() , update() , delete() , load() , get (). Hibernate session object is not thread safe, every thread should get it’s own session instance and code it after it’s work is finished.
4. openSession vs getCurrentSession :- Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. Means , if any session is present in session factory then it will provide that session. We don’t need to close it. Hibernate SessionFactory openSession() method always open a new session. We should close this session object once we are done with all the database operations.
5. Get() vs load () :- get() loads the data immediate when it’s called whereas load() return a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading. Since load() throws exception when data is not found, we should use it only when we know data exists.
6. How to integrate spring and hibernate using HibernateDaoSupport? Spring and hibernate can integrate using Spring’s SessionFactory called LocalSessionFactory. The integration process is of 3 steps :-
7. Configure the hibernate SessionFactory .
8. Extend your DAO implementation from HibernateDaoSupport
9. Wire in transaction support with AOP.
10. Which object is used to create SessionFactory object in hibernate ? Configuration Object.
11. Transient vs Detached State :- Transient object does not have association with the databases and session objects. They are simple objects and not persisted to the database. Once the last reference is lost , that means the object itself is lost. Detached object have corresponding entries in the database. These are persistent and not connected to the session object.
12. What is index and how to use it? A database index is a data structure that improves the speed of search operation in a table. Indexed can be created using one or more columns, providing the basis for the both rapid random lookups and efficient ordering of access to records. The INSERT and UPDATE statements take more time on tables having indexes, whereas the SELECT statements become fast on those tables. The reason is that while doing insert or update, a database needs to insert or update the index values as well.

Create unique index index\_name ON table\_name (col1, col2….);

1. What is AJAX and how AJAX works? AJAX allows us to send and receive data asynchronously without reloading the web page. So it is fast.
2. User sends a request from the UI and javascript call goes to XMLHttpRequest object.
3. HTTP request is sent to the server by XMLHttpRequest object.
4. Server interacts with the database using JSP.
5. Data is retrieved.
6. Server sends XML data or JSON data to the XMLHttpRequest callback function.
7. HTML and CSS data is displayed on the browser.
8. What is Lambda Expression(@functionalInterface) and it’s uses? Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface. It saves a lot of code , in case of Lambda expression , we don’t need to define the method again for providing the implementation. here , we just write the implementation code. Syntax :- (argument list ) -> {body}
9. Lambda expression example :-

@FunctionalInterface

Interface Drawable{

Public void draw();

}

Public class LanbdaExpressionExp{

Public static void main(String [] args){

Int width = 10;

Drawable d2 = () -> { Sop(“drawing “ + width); }; // with lambda

// without lambda , drawable implementation using anonymous class

Drawable d = new Drawable (){

public void draw() { Sop(“drawing “ + width); };

};

D2.draw();

}

}

Thread Example :-

Public class LambdaExpessEx{

Public static void main(String args[]){

Thread t1 = new Thread(r1);

T1.start();

// with lambda

Runnable r2 = () -> {

Sop (“thread 2 running” );

};

Thread t2 = new Thread(r2):

T2.start();

}

}