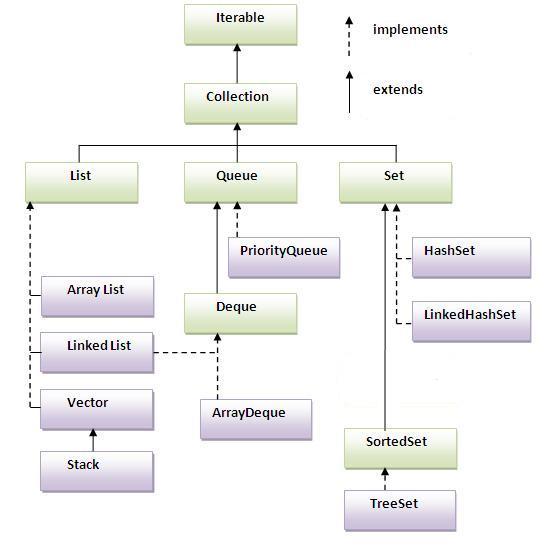
1. What is Abstraction and Encapsulation?

**Abstraction** is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

**Encapsulation** is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.

1. What all the collection you have used?



1. Given an Employee class with id, name what will happen if store Employee objects in tree set?

Exception in thread "main" java.lang.ClassCastException: Employee cannot be cast to java.lang.Comparable

at java.util.TreeMap.compare(TreeMap.java:1290)

at java.util.TreeMap.put(TreeMap.java:538)

at java.util.TreeSet.add(TreeSet.java:255)

at TreeSetExample.main(TreeSetExample.java:11)

Need to Implement Comparable

@Override

**public** **int** compareTo(Employee o)

{

**if** (o.id > id)

**return** -1;

**else**

**return** 1;

}

1. What if I don’t want to implement comparable in Employee class can I still store in Tree set?

**TreeSet<Employee> eSet = new TreeSet<>(new EmployeeComparator());**

class EmployeeComparator implements Comparator<Employee>

{

@Override

public int compare(Employee o1, Employee o2)

{

if (o1.getId() > o2.getId())

return 1;

else

return -1;

}

}

1. For a given scenario, there are n number of books in a library that I want to store in a collection. Which collection I will use and why?

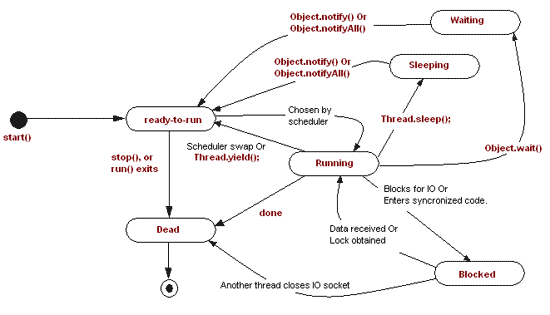
Every collection type is suitable for a particular scenario. There is no *fastest* or *best* collection.

* If you need fast access to elements using *index*, ArrayList is your answer.
* If you need fast access to elements using a *key*, use HashMap.
* If you need fast add and removal of elements, use LinkedList (but it has a *very* poor index access performance).

1. Have you used thread? What are threads? How do you run them?

A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution.

Threads are independent, if there occurs exception in one thread, it doesn't affect other threads. It shares a common memory area



1. what happens when you do thread.run and what happens when you do thread.start?

Method call & Thread Start

1. what happens when you do thread.join?

**The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.**

class TestJoinMethod1 extends Thread{

public void run(){

for(int i=1;i<=5;i++){

try{

Thread.sleep(500);

}catch(Exception e){System.out.println(e);}

System.out.println(i);

}

}

public static void main(String args[]){

TestJoinMethod1 t1=new TestJoinMethod1();

TestJoinMethod1 t2=new TestJoinMethod1();

TestJoinMethod1 t3=new TestJoinMethod1();

t1.start();

try{

t1.join();

}catch(Exception e){System.out.println(e);}

t2.start();

t3.start();

}

}

1. what happens when you do thread.start twice?

Exception in thread "main" java.lang.IllegalThreadStateException

at java.lang.Thread.start(Thread.java:705)

at TestJoinMethod1.main(TestJoinMethod1.java:32)

1. what is synchronization? How you do it?

The synchronization is mainly used to

1. To prevent thread interference.
2. To prevent consistency problem.

Mutual Exclusive

Synchronized method.

**synchronized void printTable(int n){//synchronized method**

Synchronized block.

**void printTable(int n){**

**synchronized(this){//synchronized block**

static synchronization.

**synchronized static void printTable(int n){**

Cooperation (Inter-thread communication in java)

1. what do you mean by synchronization on  object and synchronization on class ?

The snippet synchronized(X.class) uses the class instance as a monitor. As there is only one class instance (the object representing the class metadata at runtime) one thread can be in this block.

With synchronized(this) the block is guarded by the instance. For every instance only one thread may enter the block.

synchronized(X.class) is used to make sure that there is exactly one Thread in the block. synchronized(this) ensures that there is exactly one thread per instance. If this makes the actual code in the block thread-safe depends on the implementation. If mutate only state of the instance synchronized(this) is enough.

Spring:

1. -What is IOC?

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. **BeanFactory**
2. **ApplicationContext**

**Difference between BeanFactory and the ApplicationContext**

The org.springframework.beans.factory.BeanFactory and the org.springframework.context.ApplicationContext **interfaces** acts as the IoC container. The **ApplicationContext** interface is built on top of the **BeanFactory** interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application. So it is better to use ApplicationContext than BeanFactory.

**Using BeanFactory**

The XmlBeanFactory is the implementation class for the BeanFactory interface. To use the BeanFactory, we need to create the instance of XmlBeanFactory class as given below:

1. Resource resource=new ClassPathResource("applicationContext.xml");
2. BeanFactory factory=new XmlBeanFactory(resource);

The constructor of XmlBeanFactory class receives the Resource object so we need to pass the resource object to create the object of BeanFactory.

**Using ApplicationContext**

The ClassPathXmlApplicationContext class is the implementation class of ApplicationContext interface. We need to instantiate the ClassPathXmlApplicationContext class to use the ApplicationContext as given below:

1. ApplicationContext context =
2. new ClassPathXmlApplicationContext("applicationContext.xml");

The constructor of ClassPathXmlApplicationContext class receives string, so we can pass the name of the xml file to create the instance of ApplicationContext.

1. -What is auto wiring?

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

|  |  |  |
| --- | --- | --- |
| **No.** | **Mode** | **Description** |
| 1) | no | It is the default autowiring mode. It means no autowiring bydefault. |
| 2) | byName | The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method. |
| 3) | byType | The byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method. |
| 4) | constructor | The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters. |
| 5) | autodetect | It is deprecated since Spring 3. |

1. -what is setter injection and constructor injection?

Setter :

<bean id="obj" class="com.javatpoint.Employee">

<property name="id">

<value>20</value>

</property>

<property name="name">

<value>Arun</value>

</property>

<property name="city">

<value>ghaziabad</value>

</property>

Constructor

<beam id=”” class=””>

<constructor-arg ref="spellChecker"/>

</bean>

1. -Given a bean A which is dependent of bean B and Bean B is dependent A what will happen?

You would see the following output when creating a context using this configuration:

Creating instance of A

Creating instance of B

Setting property a of B instance

Setting property b of A instance

Note that when a is injected into b, a is not yet fully initialized.

Restful web service:

1. -How do you configure Restful web service?

**@Path("/UserService")**

public class UserService {

UserDao userDao = new UserDao();

**@GET**

@Path("/users")

**@Produces(MediaType.APPLICATION\_XML)**

public List<User> getUsers(){

return userDao.getAllUsers();

}

}

1. -What are different types of annotations you have used?

@Path("/users")

public class UserService {

@GET

@Path("/query")

**public Response getUsers(**

**@QueryParam("from") int from,**

**@QueryParam("to") int to,**

**@QueryParam("orderBy") List<String> orderBy) {**

return Response

.status(200)

.entity("getUsers is called, from : " + from + ", to : " + to

+ ", orderBy" + orderBy.toString()).build();

}

}

@Path("/users")

public class UserRestService {

@GET

**@Path("{id}")**

**public Response getUserById(@PathParam("id") String id) {**

return Response.status(200).entity("getUserById is called, id : " + id).build();

}

}

1. -what are security implementation you did?

OAuth, Secirity Key, Authorization

1. -What you will do if you want return response in xml format

**@Produces("application/xml")**

**Use @XmlRootElement on bean**

**return Response.status(ex.getStatus())**

**.entity(new ErrorMessage(ex))**

**.type(MediaType.APPLICATION\_JSON).**

**build();**

1. -what is load-on-startup mean? What are its values?

Container will load that class on define priority

1. -How did you handle exception handling in Rest web services

**Alternatively, you can implement and register instances of javax.ws.rs.ext.ExceptionMapper. These objects know how to map a thrown application exception to a Response object:**

**public interface ExceptionMapper<E extends Throwable> {**

**{**

**Response toResponse(E exception);**

**}**

For example, one exception that is commonly thrown in Java Persistence API (JPA)–based database applications is **javax.persistence.EntityNotFoundException**. It is thrown when JPA cannot find a particular object in the database. Instead of writing code to handle this exception explicitly, you could write an **ExceptionMapper** to handle this exception for you. Let’s do that:

@Provider

public class EntityNotFoundMapper

implements ExceptionMapper<EntityNotFoundException> {

public Response toResponse(EntityNotFoundException e) {

return Response.status(Response.Status.NOT\_FOUND).build();

}

}

Our **ExceptionMapper** implementation must be annotated with the **@Provider** annotation. This tells the JAX-RS runtime that it is a component. The class implementing the **ExceptionMapper** interface must provide the parameterized type of the **ExceptionMapper**. JAX-RS uses this generic type information to match up thrown exceptions to **ExceptionMappers**. Finally, the **toResponse()** method receives the thrown exception and creates a **Response** object that will be used to build the HTTP response.

JAX-RS supports exception inheritance as well. When an exception is thrown, JAX-RS will first try to find an **ExceptionMapper** for that exception’s type. If it cannot find one, it will look for a mapper that can handle the exception’s superclass. It will continue this process until there are no more superclasses to match against.

1. -What all are the response code you have used while returning response

|  |  |  |
| --- | --- | --- |
| BadRequestException | 400 | Malformed message |
| NotAuthorizedException | 401 | Authentication failure |
| ForbiddenException | 403 | Not permitted to access |
| NotFoundException | 404 | Couldn’t find resource |
| NotAllowedException | 405 | HTTP method not supported |
| NotAcceptableException | 406 | Client media type requested not supported |
| NotSupportedException | 415 | Client posted media type not supported |
| InternalServerErrorException | 500 | General server error |
| ServiceUnavailableException | 503 | Server is temporarily unavailable or busy |

Hibernate:

1. What is Hibernate?

**Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities.**

1. Why should one use hibernate?

|  |  |
| --- | --- |
| **JDBC** | **Hibernate** |
| With JDBC, developer has to write code to map an object model's data representation to a relational data model and its corresponding database schema. | Hibernate is flexible and powerful ORM solution to map Java classes to database tables. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this. |
| With JDBC, the automatic mapping of Java objects with database tables and vice versa conversion is to be taken care of by the developer manually with lines of code. | Hibernate provides transparent persistence and developer does not need to write code explicitly to map database tables tuples to application objects during interaction with RDBMS. |
| JDBC supports only native Structured Query Language (SQL). Developer has to find out the efficient way to access database, i.e. to select effective query from a number of queries to perform same task. | Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries. Hibernate also supports native SQL statements. It also selects an effective way to perform a database manipulation task for an application. |
| Application using JDBC to handle persistent data (database tables) having database specific code in large amount. The code written to map table data to application objects and vice versa is actually to map table fields to object properties. As table changed or database changed then it’s essential to change object structure as well as to change code written to map table-to-object/object-to-table. | Hibernate provides this mapping itself. The actual mapping between tables and application objects is done in XML files. If there is change in Database or in any table then the only need to change XML file properties. |
| With JDBC, it is developer’s responsibility to handle JDBC result set and convert it to Java objects through code to use this persistent data in application. So with JDBC, mapping between Java objects and database tables is done manually. | Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost. |
| With JDBC, caching is maintained by hand-coding. | Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance if client application reads same data many times for same write. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code. |
| In JDBC there is no check that always every user has updated data. This check has to be added by the developer. | Hibernate enables developer to define version type field to application, due to this defined field Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table. So if two users retrieve same tuple and then modify it and one user save this modified tuple to database, version is automatically updated for this tuple by Hibernate. When other user tries to save updated tuple to database then it does not allow saving it because this user does not have updated data. |

1. What is first level caching and second level caching?

#### First Level Cache

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.

#### Second Level Cache

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

1. **read-only:** caching will work for read only operation.
2. **nonstrict-read-write:** caching will work for read and write but one at a time.
3. **read-write:** caching will work for read and write, can be used simultaneously.
4. **transactional:** caching will work for transaction.

<cache usage="read-only" />

1. What is cascading?

Cascading is about deletion of one object from the database causing deletion of other (dependent) objects .... though of course it is typically database rows that are actually getting deleted.

Cascading can also apply to saves / updates as well.

1. What is lazy loading in hibernate?

**Lazy fetching** decides whether to **load** child objects while **loading** the Parent Object. You need to do this setting respective **hibernate** mapping file of the parent class. **Lazy** = true (means not to **load** child) By default the **lazy loading** of the child objects is true

1. How do you implement transaction management in hibernate

Session session = null;

Transaction tx = null;

try {

session = sessionFactory.openSession();

tx = session.beginTransaction();

//some action

tx.commit();

}catch (Exception ex) {

ex.printStackTrace();

tx.rollback();

}

finally {session.close();}

Given a scenario you have two glass balls and a 50 floor building what is the effective way to identify the max floor where the glass ball doesn’t  break?

==================================================================================

Core Java:

1. Explain Overlaoding?

Ans: method name same with different method signature.

1. Can we do overloading in same class?

Ans: Yes

1. Explain Overriding?

Ans: method name same with same method signature.

1. Can we do overriding in same class?

Ans: No

1. What comes in method signature?

Ans: method name with input parameters +return type and access specifiers

1. Difference between encapsulation and abstraction?

Ans: Encapsulation= Binding state and behavior of class into single enity. Abstraction= Hiding the functionality of method and just providing interface.

1. Checked exception and unchecked exception?

Ans: Checked exception = Exception under Throwable class except RunTime and error. Unchecked= Exception exte4nding Runtime

1. If there are three method calls and each one is throwing different exceptions like first one is throwing custom exception, second as FilenotFoundException and third is throwing IOException then how you will handle it?

Ans : will  put the code inside try catch block and also declare method with keyword ‘throws’ particular exception at method level and for catch will use multiple catch block with hierarchy.

1. Explain Set and List in collection?

Ans: Set= allows unique entity and don’t allow duplicates. List= Allows duplicate

1. Insertion order in list?

Ans: Every class which is implementing List then it will follow insertion order.

1. If there are two employee object with same value and we putting into set then how it will work?

Ans: it will override

1. If there are two employee object with different  value and we putting into set then how it will work?

Ans: it will check at backend with hashcode and equals method because internally it is creating Hashmap

1. Comparable vs comparator?

Ans: Comaparator= compare() Comaparabel=compareTo()

Spring:

1. What is IOC and DI?

Ans: Process where object itself deciding it’s dependency and those are created by container and injected so its inverse process hence the name is IOC. DI is also the same as its just another word to IOC.

1. How you will Autowire beans?

Ans: By @Autowired annotation or adding attribute in xml file of bean definition as autowired=”byname/byType”

1. What are types of @Autowired?

Ans: byname,

byType,

constructor

1. Which is by default mode of autowiring?

Asn: byType

1. If there are two classes B and C  implementing the interface A and there is class D which is having all these three in it . But @Autowired is written above only Interface A field and others are just declared then by default which will be injected?

Interface A because we are using @autowired on that field only.

1. Explain Jersey?

Ans: We can write rest services using Jersey with annotaitons @Path/@Produces/@consumes/@post

1. Flow of project using Jersey?

Ans: Told the actual flow of project.

1. How to configure Jersey?

Ans: Inside class which is extending ResourceConfig. There you will register the controller class.

<servlet>

<servlet-name>jersey-serlvet</servlet-name>

<servlet-class>

com.sun.jersey.spi.container.servlet.ServletContainer

</servlet-class>

<init-param>

<param-name>com.sun.jersey.config.property.packages</param-name>

<param-value>com.mkyong.rest</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

1. What are annotation used in Spring and jersey?

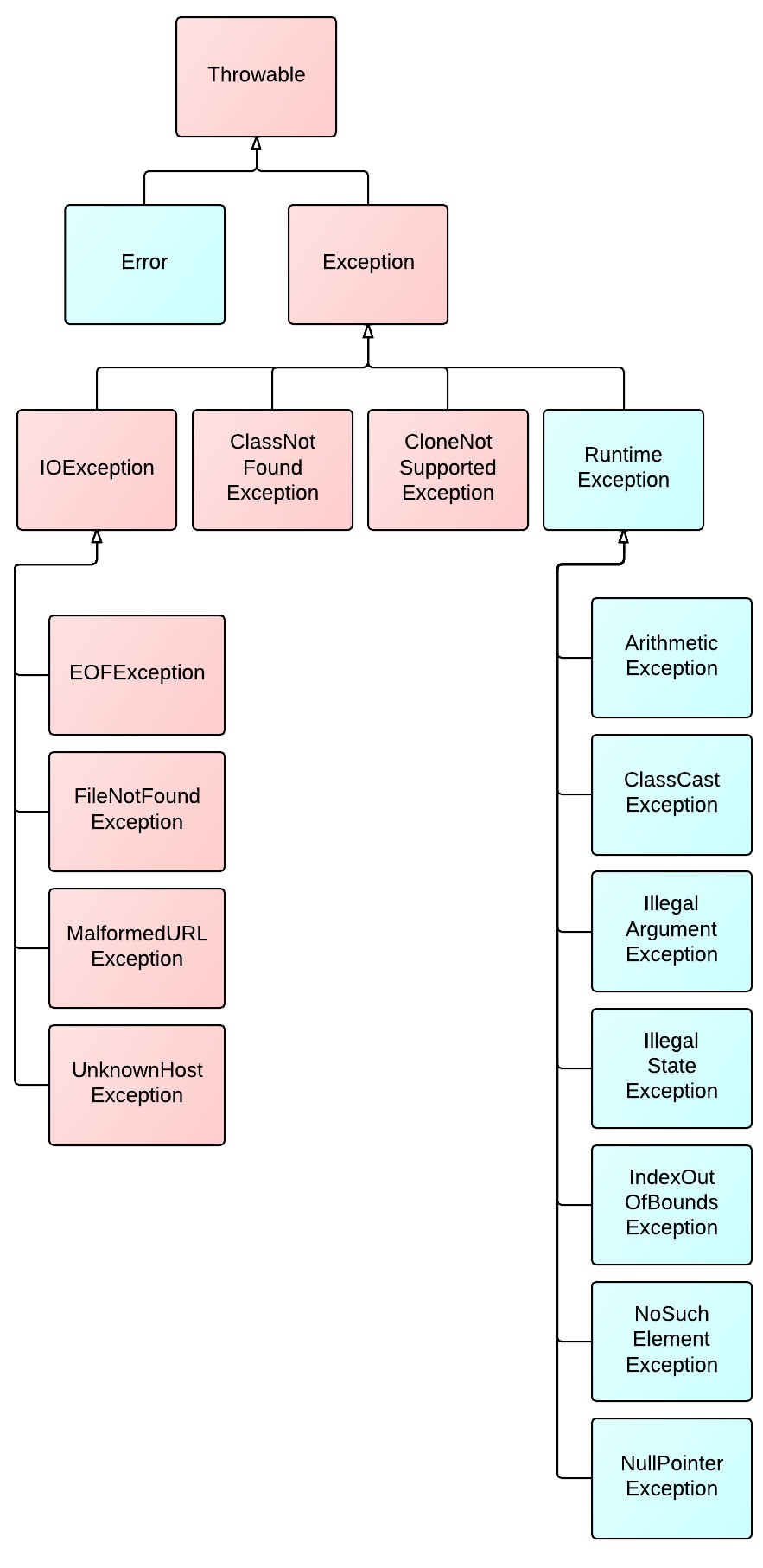
Ans: @Path/@Produces/@consumes/@post

1. How the request will come to particular controller?

Ans: through URI wich is written in @path

==================================================================================

1. Explain about your previous project and experience with any GE project
2. Rate yourself on scale of 5 for Java, Web services, Spring, ORM
3. What is Inheritance? What are its rules?
4. What is aggregation, and Composition?
5. Explain Exception hierarchy



1. What is the disadvantage of Finalize method?
2. Which collections have you used? Which one is good for faster access?
3. Why String is preferred for HashMap key?

One of the reason why we generally use **String** as a **key in HashMap** is that since **String** is immutable in **Java** that allows **String** to cache its hashcode , being immutable **String** in **Java** caches its hash code and do not calculate every time we call hashcode method of **String**, which makes it very fast as **HashMap key**.

1. Have you worked on threading?
2. Have you used any design patterns? Please explain with use cases?
3. What is the difference between Strategy and State pattern?
4. What is Singleton pattern? How you create it? Is it full proof? How to avoid instance creation?

public class SingleObject {

//create an object of SingleObject

private static SingleObject instance = new SingleObject();

//make the constructor private so that this class cannot be

//instantiated

private SingleObject(){}

//Get the only object available

public static SingleObject getInstance(){

return instance;

}

public void showMessage(){

System.out.println("Hello World!");

}

}

1. If I have a file with 20 K lines of data and need to cache it using Spring how will you design it ? Each request may ask for different set of data.

Not worked on Spring Cache

1. Have you used any ORM? ORM concepts

Hibernate

1. What is SOAP and REST?

**SOAP** (Simple Object Access Protocol) is a protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to induce extensibility, neutrality and independence.

**SAML :** Security Assertion Markup Language is an XML-based, open-standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. **SAML** is a product of the OASIS Security Services Technical Committee.

**REST :** While **REST** stands for **Representational State Transfer**, which is an architectural style for networked hypermedia applications, it is primarily used to build **Web services** that are lightweight, maintainable, and scalable. A **service** based on **REST** is called a **RESTful service**

1. Have you used any REST web services? How have you created a service using Jersey? Explain Jersey annotations
2. What is PUT and POST request?

**Do you name your URL objects you create explicitly, or let the server decide? If you name them then use PUT. If you let the server decide then use POST.**

**PUT is idempotent, so if you PUT an object twice, it has no effect. This is a nice property, so I would use PUT when possible.**

**You can update or create a resource with PUT with the same object URL**

**With POST you can have 2 requests coming in at the same time making modifications to a URL, and they may update different parts of the object.**

1. What is TDD and BDD?

TDD or Test-Driven Development is a process for when you write and run your tests. Following it makes it possible to have a very high test-coverage. Test-coverage refers to the percentage of your code that is tested automatically, so a higher number is better. TDD also reduces the likelihood of having bugs in your tests, which can otherwise be difficult to track down.

The TDD process consists of the following steps:

1. Start by writing a test
2. Run the test and any other tests. At this point, your newly added test should fail. If it doesn’t fail here, it might not be testing the right thing and thus has a bug in it.
3. Write the minimum amount of code required to make the test pass
4. Run the tests to check the new test passes
5. Optionally refactor your code
6. Repeat from 1

It can take some effort to learn well, but spending the time can pay off big. TDD projects often get a code-coverage of 90-100%, which means maintaining the code and adding new features is easy. This is because you have a large set of tests, so you can trust your code and changes work, and didn’t break any other code either.

Some think you must use the “xUnit style” testing tools to use the TDD process. This is not the case – TDD works great with unit tests, but you can apply it to other testing methods as well. It also does not require any specific tool or syntax.

BDD : Behavior-Driven Developmen

The important part here is thinking of the scenario, rather than the implementation, can lead you to design a better test.

1. What is the difference between IOC and DI?

**IoC is a generic term meaning rather than having the application call the methods in a framework, the framework calls implementations provided by the application.**

**DI is a form of IoC, where implementations are passed into an object through constructors/setters/service look-ups, which the object will 'depend' on in order to behave correctly.**

1. Java 8 features? What is Lambda?

There are dozens of features added to Java 8, the most significant ones are mentioned below −

• Lambda expression − Adds functional processing capability to Java.

• Method references − Referencing functions by their names instead of invoking them directly. Using functions as parameter.

• Default method − Interface to have default method implementation.

• New tools − New compiler tools and utilities are added like ‘jdeps’ to figure out dependencies.

• Stream API − New stream API to facilitate pipeline processing.

• Date Time API − Improved date time API.

• Optional − Emphasis on best practices to handle null values properly.

• Nashorn, JavaScript Engine − A Java-based engine to execute JavaScript code.

Along with these new featuers, lots of feature enhancements are done under-the-hood, at both compiler and JVM level.

1. What is Autowiring?
2. How will you sort a collection? How do I sort a collection with Employee object?

Comparable => comapareTo() or comparable => compare

1. Explain Spring Bean Lifecycle



1. Any questions for us?

==================================================================================

1. Which collection you will use if you want a faster access?
2. Consider a book of library and you want to design an application then how would you approach?
3. What would be abstraction layer for the library problem and how would you implement encapsulation?
4. What is singleton pattern? How you avoid refection and cloning while implementing singleton?

**private BookingFactory() {**

**if (instance != null)**

**throw new IllegalStateException("Only one instance may be created");**

**System.out.println("Object is created.");**

**}**

1. How can you create a thread?
2. What will happen if you call Thread.run()?
3. What if two thread accessing same variable in class and you want to implement concurrency?
4. What is atomic?

Here's an example, because an example is often clearer than a long explanation. Suppose foo is a variable of type long. The following operation is not an atomic operation:

foo = 65465498L;

Indeed, the variable is written using two separate operations: one that writes the first 32 bits, and a second one which writes the last 32 bits. That means that another thread might read the value of foo, and see the intermediate state. Making the operation atomic consists in using synchronization mechanisms in order to make sure that the operation is seen, from any other thread, as a single, atomic (i.e. not splittable in parts), operation. That means that any other thread, once the operation is made atomic, will either see the value of foo before the assignment, or after the assignment. But never the intermediate value.

A simple way of doing this is to make the variable volatile:

private volatile long foo;

Or to synchronize every access to the variable:

public synchronized void setFoo(long value) {

this.foo = value;

}

public synchronized long getFoo() {

return this.foo;

}

// no other use of foo outside of these two methods, unless also synchronized

Or to replace it with an AtomicLong:

private AtomicLong foo;

1. Do you know about concurrency package?

**java.util.concurrent**

1. What is difference between wait and sleep?

**The fundamental difference is wait() is from Object and sleep() is static method of Thread . The major difference is that wait() releases the lock while sleep() doesn't releas any lock while waiting. The wait() is used for inter-thread communication while sleep() is used to introduce pause on execution, generally.**

1. What is IOC? Difference between IOC and DI?
2. What are different scopes in Spring?
3. What are the advantages of Spring boot and when you would consider the spring boot as a development technology?
4. What if class A and Class B are singleton and C is prototype . If A and B both are using C then how many instance of C are created
5. Have you work on any web services ?
6. What provider you used in your application for web services?
7. How would you expose your rest web service in your application?
8. Have you face any problem or difficulty while implementing rest framework in your project?
9. Suppose there is a date format you want to send to rest webservice then how Jersey will handle the date format for conversion into date type?
10. Have you face any problem in your previous project and how you resolved it?
11. You know any ORM framework?
12. How much you are confident if you want to implement (gave idea of the project and technology stack) rate out of 5?
13. What would be your expectation from us (client) when you are working on the development project?
14. What is TDD and BDD?

==================================================================================

1. Difference between Comparable and comparator?

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List,Comparator)** method. |

2. What is composition and aggregation. Explain with example?  
3. SOLID principles?

* **S** – Single-responsiblity principle
* **O** – Open-closed principle
* **L** – Liskov substitution principle
* **I** – Interface segregation principle
* **D** – Dependency Inversion Principle

4. How will we implement concurrency without java concurrency package?

**Synchronization & Lock**Difference between IOC and DI?   
In Spring xml config file, two bean definitions implement the same interface. Which bean definition will container pick?  
What is composite key in relational db?

Difference between Junit and BDD. Which tool you use for BDD. Advantages of using BDD?  
How Spring @autowire by type works?  
How does TreeSet work internally?

==================================================================================

1.  Brief about the previous experience?

2.  What is overriding?

3. What does method signature contains?

4. Can a overridden method declare exception if it is not declared in the parent class?

5. What is checked and unchecked exception?

6.  Name some of the checked exceptions?

**Unchecked Exception List**  
ArrayIndexOutOfBoundsException  
ClassCastException  
IllegalArgumentException  
IllegalStateException  
NullPointerException  
NumberFormatException  
AssertionError  
ExceptionInInitializerError  
StackOverflowError  
NoClassDefFoundError

**Checked Exception List**  
Exception  
IOException  
FileNotFoundException  
ParseException  
ClassNotFoundException  
CloneNotSupportedException  
InstantiationException  
InterruptedException  
NoSuchMethodException  
NoSuchFieldException

7. How will you make any class to get a thread object?

8. How will you start a thread?

9. [When would you call thread.run()](http://stackoverflow.com/questions/262816/when-would-you-call-javas-thread-run-instead-of-thread-start) ?

10. Have you worked on webservices?

11. What is SOAP webservice?

12. Do you have experience on RESTful webservice?

13. Why will you go for RESTful webservice, mean advantages over SOAP webservice?

14. What are the major difference between SOAP and RESTful webservice?

15. What is Spring MVC?

16. What is Spring IOC?

17. What is Autowiring?

18. How @Autowired is detected by compiler?

19. What are the different ways of autowiring?

20. Mention the different core java design patterns and explain?

21. What is TDD?

1. Brief about your experience and technology you have worked on ?
2. Which company you have worked earlier ?
3. Which project you have worked on, brief about your project ?
4. Have you worked on Web services ?
5. What is REST and SOAP web service ?
6. Difference between SOAP and REST web service ?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **No.** | **SOAP** | **REST** | | 1) | SOAP is a **protocol**. | REST is an **architectural style**. | | 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. | | 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. | | 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. | | 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. | | 6) | SOAP **defines standards** to be strictly followed. | REST does not define too much standards like SOAP. | | 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. | | 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. | | 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. | | 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. | |

1. When did you last used any web service ?
2. Difference between XML and JSON ?

|  |  |
| --- | --- |
| JSON | XML |
| **Stands For** | **JSON : “JavaScript Object Notation”.** | **XML: “Extensible Markup Language”.** |
| **Extended From** | JSON is extended from **JavaScript.** | XML is extended from **SGML** : “**Standard Generalized Markup Language**“. |
| **Purpose** | JSON is one type of text-based format or standard for interchanging  data i.e. **human readable**.  JSON is developed by “**Douglas Crockford**”. | XML is a **Markup Language** having format that contains set of rules for the encoding the documents which is readable for both **human & machine**. XML is developed by **W3C** :“**World Wide Web Consortium** “. |
| **Syntax** | JSON syntax is lighter than XML as JSON has serialized format of data having  less redundancy. JSON does not contain start and end tags. | XML is not so lighter as JSON as having start and end tags and it takes more character than JSON to represent same data. |
| **Speed** | JSON is light – weighted in compare to XML as it has serialized format and so faster also. | XML is not so light weighted as JSON. |
| **Support of Data Type & Array** | JSON supports **datatype** including integer and strings, JSON also supports array. | XML does not provide any data type so needs to be **parsed** into particular datatype. No direct support for array also. |
| **Object Support** | JSON has support of native **objects**. | XML can get support of objects through mixed use of attributes & elements. |
| **Comments** | JSON does not support **Comments** | XML supports comments. |
| **Namespace** | JSON does not have support for Namespaces. | XML supports **Namespaces**. |
| **Mapping** | JSON is **data oriented** and can be mapped more easily. | XML is **document oriented** and needs more effort for **mapping**. |
| **Parsing** | JSON uses only **evel()** for **parsing** i.e. for interpreting the JavaScript code & returns the result. It does not need any additional code for parsing. | XML needs XML **Document Object Model** (DOM) implementation & with that additional code for mapping text back to the JavaScript objects. |
| **Application** | For W**eb services**, JSON is better. | For **configuration**, XML is better. |
| **Changing Format** | You can not change JSON data to other **format**. | In XML, using **XSLT** you can change XML data into another format like **comma –delimited, plain text**, JSON, etc. |
| **Access** | In JSON no such is interface for getting direct access to a part in JSON data-structure. | In XML, Using **XPath**, it is possible to get the direct access to a particular part of multiple parts of XML da |

1. Have you worked on SPRING ?
2. What is IOC ?
3. Do you have healthcare domain experience ?
4. List out the design patterns you have came across ?