**General**

**1. Describe in detail your process for unit testing. Give an example of how you tested and what**

**frameworks you used. What level of code coverage do you maintain and why?**

**(Explain in detail and use pseudo code where appropriate)**

**The idea is to develop a unit test by following these 3 simple steps: Arrange – setup the testing objects and prepare the prerequisites for your test. Act – perform the actual work of the test. Assert – verify the result.**

**Write the program logic**

**Let's write the logic to find the maximum number for an array.**

1. **package com.javatpoint.logic;**
2. **public class Calculation {**
4. **public static int findMax(int arr[]){**
5. **int max=0;**
6. **for(int i=1;i<arr.length;i++){**
7. **if(max<arr[i])**
8. **max=arr[i];**
9. **}**
10. **return max;**
11. **}**
12. **}**

**Write the test case**

**Here, we are using JUnit 4, so there is no need to inherit TestCase class. The main testing code is written in the testFindMax() method. But we can also perform some task before and after each test, as you can see in the given program.**

1. **package com.javatpoint.testcase;**
3. **import static org.junit.Assert.\*;**
4. **import com.javatpoint.logic.\*;**
5. **import org.junit.Test;**
7. **public class TestLogic {**
9. **@Test**
10. **public void testFindMax(){**
11. **assertEquals(4,Calculation.findMax(new int[]{1,3,4,2}));**
12. **assertEquals(-1,Calculation.findMax(new int[]{-12,-1,-3,-4,-2}));**
13. **}**
14. **}**

**Framework: Mockito –Junit4**

**With that being said it is generally accepted that 80% coverage is a good goal to aim for. Trying to reach a higher coverage might turn out to be costly, while not necessary producing enough benefit. The first time you run your coverage tool you might find that you have a fairly low percentage of coverage.**

**2. Describe in detail your CI/CD process, what steps are involved for building, testing, security,**

**and release.  What experience do you have building CI/CD pipelines?**

**A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD.**

**There are four stages of a CI/CD pipeline 1) Source Stage, 2) Build Stage, 3) Test Stage, 4) Deploy Stage. Important CI/CD tools are Jenkins, Bambo, and Circle CI. CI/CD pipeline can improve reliability**

**I have almost 4 years of experience in building CI/CD pipelines.**

**3. Explain in detail how you would make a REST API for the following data model:**

**- Account that contains basic account details (such as Name, office address, office phone**

**number, etc)**

**- Contacts, that belong to accounts, that contain basic contact details (Name, title, personal**

**address, personal phone number, etc)**

**Focus on the REST API design, security considerations, and techniques you would use. Focus**

**first on the REST structure and then go into any specifics for your framework.**

**(Explain in detail and use pseudo code to show the various parts of your API )**

**This call returns an example of data model structure that will be used in Data-In operations.**

**URL**

**/ Account/title**

**Method**

**GET**

**Headers**

**None**

**Request type**

**None**

**Response type**

**JSON**

**URL Parameters**

**Required: None**

**Optional: None**

**Payload**

**None**

**Success response**

**Retrieve All Entity Details in an Environment**

**Parameters:**

**projectIdSpecifies the Account ID.**

**projectVersionIdSpecifies the project version.**

**includeRelationships(Optional) Includes entity relationships for all entities.**

**Values: true or false (default)**

**includeRelatedEntities(Optional) Includes all related entities.**

**Values: true or false (default)**

**includeAttributes(Optional) Includes all entity attributes.**

**Values: true or false (default)**

**includeHierarchy(Optional) Includes the fully qualified path of an entity in a hierarchical format.**

**Values: true or false (default)**

**q(Optional) Queries the entities to identify a specific entity name, attribute name, schema name, data source name, or database name. If no key field is specified, all entities, attributes, schema names, data sources, and databases are queried. The string value in this parameter is case insensitive.**

**This parameter supports the basic wild card characters such as \* (used to match one or more characters) and ? (used to match a single character). All supplied search terms are 'ANDed' together. For example, when you specify "q=attribute=address+database=travel" the response includes all entities that have either a travel database and an address attribute.**

**You can perform a search on the relevant data sources based on one or more of the following key fields:**

**entity=**

**Matches an entity name. For example, entity=cust\* matches all entities starting with "cust" search term, such as CUSTOMER, custs, customers, and so on.**

**attribute=**

**Matches an attribute name.**

**REST Components**

**Resource Path (request target)**

**HTTP Verb.**

**Body.**

**Header.**

**Framework:- Spring Boot 2.7.0**

**4. Given the REST API you created in the previous question (if you did not answer the previous**

**question assume a service to save/load accounts and contacts), describe in detail how you**

**would implement a user experience that consists of the following:**

**- A Table view that display details of the Multiple Accounts that has a link to each that allows**

**you to click into a detail view**

**- A Detail view that shows more details of the Account and allows the user to edit and save**

**the account.**

**You can assume any API structure and you may use any framework / library as long as you**

**state what it is and its use.**

**(Please provide in depth details and use pseudo code to show the various parts of your UX.**

**What approaches are you using and why)**

**Account API**

**The Account API provides endpoints that allow you to perform create, read, update, and delete (CRUD) operations on existing tables.**

**The calling user must have sufficient roles to access the data in the table specified in the request.**

**DELETE /now/Account/{Accountid}/{sys\_id}**

**Deletes the specified record from the specified table.**

**URL Format**

|  |  |
| --- | --- |
| **Name** | **Value** |
| **Versioned URL** | **/api/now/{api\_version}/table/{tableName}/{sys\_id}** |
| **Default URL** | **/api/now/table/{tableName}/{sys\_id}** |

**Path parameters**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **api\_version** | **Optional. Version of the endpoint to access. For example, v1 or v2. Only specify this value to use an endpoint version other than the latest.**  **Data type: String** |
| **sys\_id** | **Sys\_id of the record to delete.**  **Data type: String** |
| **tableName** | **Name of the table from which to delete the specified record, such as "incident" or "asset".**  **Data type: String** |

**Query parameters**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **sysparm\_query\_no\_domain** | **Flag that indicates whether to restrict the record search to only the domains for which the logged in user is configured.**  **Valid values:**   * **false: Exclude the record if it is in a domain that the currently logged in user is not configured to access.** * **true: Include the record even if it is in a domain that the currently logged in user is not configured to access.**   **Data type: Boolean**  **Default: false**  **Note: The sysparm\_query\_no\_domain parameter is available only to system administrators or users who have the query\_no\_domain\_table\_api role.** |

**Request body parameters (XML or JSON)**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **None** |  |

**Headers**

**The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST APIRequest headers**

|  |  |
| --- | --- |
| **Header** | **Description** |
| **Accept** | **Data format of the response body. Supported types: application/json or application/xml.**  **Default: application/json** |

**Response headers**

|  |  |
| --- | --- |
| **Header** | **Description** |
| **None** |  |

**Status codes**

**The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API,**

**Status codes**

|  |  |
| --- | --- |
| **Status code** | **Description** |
| **204** | **Indicates that the request completed successfully.** |

**Response body parameters (JSON or XML)**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **None** |  |

* **curl**

**Delete a record from the Incident table.**

**5.  What use cases have you used Docker in the past. Explain a high level of how dockerfiles work**

**and how docker works. What are some of the advantages / disadvantages of docker?**

**Docker use cases for businesses**

* **1) Adoption of DevOps. ...**
* **2) App infrastructure isolation. ...**
* **3) Multi-tenancy support. ...**
* **4) Improvement in software testing. ...**
* **5) Smart Disaster Recovery (DR) ...**
* **6) Continuous rapid deployment. ...**
* **7) Creation of microservices architecture.**

**Dockerfile is a text document containing all the commands the user requires to call on the command line to assemble an image. With the help of a Dockerfile, users can create an automated build that executes several command-line instructions in succession.**

**What are the advantages of Docker?**

**The benefits of Docker in building and deploying applications are many:**

* **Caching a cluster of containers.**
* **Flexible resource sharing.**
* **Scalability - many containers can be placed in a single host.**
* **Running your service on hardware that is much cheaper than standard servers.**

**Disadvantage of Dockers**

* **Docker is not good for application that requires rich GUI.**
* **It is difficult to manage large amount of containers.**
* **Docker does not provide cross-platform compatibility means if an application is designed to run in a Docker container on windows, then it cannot run on Linux Docker container.**

**Javascript / Typescript / Browser / Node**

**• What are some ways memory leaks occur in the browser with Javascript? What are some**

**ways to resolve or prevent these memory leaks?**

**The main cause of memory leaks in an application is due to unwanted references. The garbage collector finds the memory that is no longer in use by the program and releases it back to the operating system for further allocation**

**Forgotten timers or callbacks**

**Having a setTimeout or a setInterval referencing some object in the callback is the most common way of preventing the object from being garbage collected. If we set the recurring timer in our code the reference to the object from the timer's callback will stay active for as long as the callback is invocable.**

**How to prevent it: Especially if the callback’s lifespan is undefined or indefinite:**

* **being aware of the objects referenced from the timer’s callback,**
* **using the handle returned from the timer to cancel it when necessary.**
* **returned from the timer to cancel it when necessary.**

**(Use pseudo code to show how some of the leaks are created / fixed)**

**• What is web accessibility. What experience do you have with accessibility in the past? Give**

**some examples of Accessibility in the browser.**

**Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them. More specifically, people can: perceive, understand, navigate, and interact with the Web.**

**Common examples of important accessibility features include: Image alt text. Keyboard accessibility. Sequential heading structure**

**(Explain in detail and use pseudo code where appropriate)**

**• What is responsive web design? What experience do you have building responsive websites?**

**What ways to build responsive websites using HTML/CSS/JS?**

**(Explain in detail and use pseudo code where appropriate)**

**a responsive web design is an approach where a web designer develops a single web page that automatically resizes itself according to the screen's size. In this approach, the web page's design and development respond to the user's behavior and environment, ensuring a smoother user experience**

**The best way to achieve a responsive design with CSS and HTML is through media queries. You can place a media query within a CSS file or the HTML**link**tag.**

**• Describe in detail the Flux/Redux pattern. You may use React Redux, Angular NGRX, or Vue**

**Vuex/Pinia as examples.**

**(Use pseudo code to give various examples of framework you use)**

**The primary difference of Flux vs Redux is that Flux includes multiple Stores per app, but Redux includes a single Store per app. Rather than placing state information in multiple Stores across the application, Redux keeps everything in one region of the app.**

**Redux is actually view-layer agnostic, so it can easily be used with Vue via simple bindings. Vuex is different in that it knows it's in a Vue app. This allows it to better integrate with Vue, offering a more intuitive API and improved development experience.**

**• Describe in detail, a Node application, and your role within that project. How did you break up**

**the file/folder structure? What libraries/frameworks did you use and why? How did you**

**implement middleware?**

**(Explain in detail and use pseudo code where appropriate)**

**Angular**

**1. What are Lifecycle hooks in Angular? Explain some life cycles hooks and their uses.**

**(Explain in detail and use pseudo code where appropriate)**

**Lifecycle hooks are a special functionality in Angular that allow us to “hook into” and run code at a specific lifecycle event of a component or directive. Angular manages components and directives for us when it creates them, updates them, or destroys them.**

|  |  |
| --- | --- |
| **COMPONENT** | **DETAILS** |
| [**Peek-a-boo**](https://angular.io/guide/lifecycle-hooks#peek-a-boo) | **Demonstrates every lifecycle hook. Each hook method writes to the on-screen log.** |
| [**Spy**](https://angular.io/guide/lifecycle-hooks#spy) | **Shows how to use lifecycle hooks with a custom directive. The SpyDirective implements the ngOnInit() and ngOnDestroy() hooks, and uses them to watch and report when an element goes in or out of the current view.** |
| [**OnChanges**](https://angular.io/guide/lifecycle-hooks#onchanges) | **Demonstrates how Angular calls the ngOnChanges() hook every time one of the component input properties changes, and shows how to interpret the changes object passed to the hook method.** |
| [**DoCheck**](https://angular.io/guide/lifecycle-hooks#docheck) | **Implements the ngDoCheck() method with custom change detection. Watch the hook post changes to a log to see how often Angular calls this hook.** |
| [**AfterView**](https://angular.io/guide/lifecycle-hooks#afterview) | **Shows what Angular means by a** [**view**](https://angular.io/guide/glossary#view)**. Demonstrates the ngAfterViewInit() and ngAfterViewChecked() hooks.** |
| [**AfterContent**](https://angular.io/guide/lifecycle-hooks#aftercontent) | **Shows how to project external content into a component and how to distinguish projected content from a component's view children. Demonstrates the ngAfterContentInit() and ngAfterContentChecked() hooks.** |

**2. What are different ways of data binding within templates?**

**(Explain in detail and use pseudo code where appropriate)**

* **String Interpolation: Interpolation refers to embedding expressions into marked up text. By default, interpolation uses the double curly braces {{ and }} as delimiters.**
* **Property Binding: Property binding moves a value in one direction, from a component's property into a target element property.**
* **Event Binding: Event binding lets you listen for and respond to user actions such as keystrokes, mouse movements, clicks, and touches.**
* **Two-way binding: What is 2 way data binding in Angular?**
* **The two-way data binding in Angular is used to display information to the end user and allows the end user to make changes to the underlying data using the UI.**

**3. Describe the differences AOT vs JIT compiling and what their uses are.**

**(Explain in detail and use pseudo code where appropriate)**

**There are two ways of compiling a Java application: using Just in Time Compilation (JIT) or Ahead of Time Compilation (AOT). The first is the default mode, and it is used by the Java Hotspot Virtual Machine to translate bytecode into machine code at runtime.**

**4. Describe in detail how does Lazy Loading work within Angular?**

**(Explain in detail and use pseudo code where appropriate)**

**Lazy loading is the process of loading components, modules, or other assets of a website as they're required. Since Angular creates a SPA (Single Page Application), all of its components are loaded at once. This means that a lot of unnecessary libraries or modules might be loaded as well.**

**5. What challenges have you experienced and what solutions have you had for testing various**

**aspects of Angular?**

**(Explain in detail and use pseudo code where appropriate)   .**

**In deciding the experience of the end user, app efficiency is a significant factor. Do you want to know how to figure out whether or not your app is doing well?**

**Here are some determiners:**

**A sharp reduction in the traffic of your app visits.**

**A decrease in the engagement rate.**

**An increase in the bounce rate.**

**Once you begin to note one or more of the above-mentioned determiners, it is time you take your performance of the Angular app seriously. Now, it is important that you concentrate on why things are going wrong instead of looking at how wrong things are going in the backend.**

**How to fix the most common performance problems**

**Add onPush change detection to components (make sure inputs are immutable)**

**Use pure pipes or async pipe (subscribe to observables) instead of using template methods.**

**Use trackBy in ngFor for dynamic lists.**

**Java**

**1. Explain differences between Collection, List, and Set and also classes vs interface?**

**(Explain in detail and use pseudo code where appropriate)**

**List preserves the insertion order, it allows positional access and insertion of elements. The set interface in the java. util package and extends Collection interface is an unordered collection of objects in which duplicate values cannot be stored. It is an interface that implements the maths set.The main difference between List and Set is that Set is unordered and contains different elements, whereas the list is ordered and can contain the same elements in it.**

**2. Explain how to achieve best performance in java?**

**(Explain in detail and use pseudo code where appropriate)**

**Java Optimization: Avoid Writing Long Methods. ...**

**Avoid Using the BigDecimal Class. ...**

**Use Primitive Types Wherever Possible. ...**

**Avoid Using Regular Expressions in Your Java Code. ...**

**Perform Profiling and Load Testing. ...**

**Use Stored Procedures Instead of Queries.**

**3. Explain few java exceptions that you came across in your experience?**

**(Explain in detail and use pseudo code where appropriate)**

**Checked and unchecked and error too**

**Checked Expection ClassNotFoundException, IOException, SQLException**

**Uncheked expection: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException,**

**Error: like a disk failure or a RAM failure.**

**4. What design patterns did you use in your experience? What is a singleton design pattern?**

**(Explain in detail and use pseudo code where appropriate)**

**We used Singleton design pattern**

**Implementation of Singleton class**

**An implementation of singleton class should have following properties:**

1. **It should have only one instance : This is done by providing an instance of the class from within the class. Outer classes or subclasses should be prevented to create the instance. This is done by making the constructor private in java so that no class can access the constructor and hence cannot instantiate it.**
2. **Instance should be globally accessible : Instance of singleton class should be globally accessible so that each class can use it. In Java, it is done by making the access-specifier of instance public.**

**//A singleton class should have public visibility**

**//so that complete application can use**

**public class GFG {**

**//static instance of class globally accessible**

**public static GFG instance = new GFG();**

**private GFG() {**

**// private constructor so that class**

**//cannot be instantiated from outside**

**//this class**

**}**

**}**

**5. What is the difference between Java 8 and Java 11 versions?**

**(Explain in detail and use pseudo code where appropriate)**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Java 8** | **Java 11** |
| **1.** | **The appletviewer tool is available in Java 8.** | **The appletviewer tool is not available in Java 11.** |
| **2.** | **Java 8 has the AWTUtilities class, which we didn't recommend you to use because it can break any program depending on it.** | **In Java 11, the AWTUtilities class is not available.** |
| **3.** | **It has fewer string methods.** | **Several new methods of String such as isBlank(), lines(),repeat(n), stripLeading(), stripTrailing(), and strip() are introduce.** |
| **4.** | **No special variables are used for lambda parameters.** | **Java 11 allows us to use var variables to be used in lambda expressions.** |
| **5.** | **Java Deployment Technologies are available in Java 8.** | **Java Deployment Technologies are removed in Java 11.** |
| **6.** | **JMC and JavaFX are available in the Oracle JDK.** | **JMC and JavaFX are removed from the Oracle JDK in Java 11.** |

**REST / Spring Boot**

**1. What are the differences between Restful, SOAP and RPC. Which one would you prefer and**

**why?**

**(Explain in detail and use pseudo code where appropriate)**

**Rest is better**

**REST allows a greater variety of data formats, whereas SOAP only allows XML. Coupled with JSON (which typically works better with data and offers faster parsing), REST is generally considered easier to work with. Thanks to JSON, REST offers better support for browser clients.**

**The most fundamental difference between RPC and REST is that RPC was designed for actions, while REST is resource-centric. RPC executes procedures and commands with ease. Alternatively, REST is ideal for domain modeling and handling large quantities of data.**

**2. How to implement parallel processing while consuming multiple API&#39;s in your API?**

**(Explain in detail and use pseudo code where appropriate)**

**Calling APIs in parallel with Java code**

**public static void updateUser(String userId) {**

**String url = BASE\_URL + userId + API\_KEY;**

**HttpResponse outGetReq = sendHttpReqAndCheckThresholds(url, "GET", null);**

**JSONObject jsonObject = new JSONObject(outGetReq. ...**

**// PUT to change middle\_name.**

**JsonObject.**

**Using javascript: PromiseAll method**

**3. What framework(s) do you use for Unit testing?**

**(Explain in detail and use pseudo code where appropriate)**

**Mockito** **framework**

**Mockito is a java based mocking framework, used in conjunction with other testing frameworks such as JUnit and TestNG. It internally uses Java Reflection API and allows to create objects of a service. A mock object returns a dummy data and avoids external dependencies**

**4. What security protocols do you use while designing a REST API?**

**(Explain in detail and use pseudo code where appropriate)**

**Use HTTPS/TLS for REST APIs**

**HTTPS and Transport Layer Security (TLS) offer a secured protocol to transfer encrypted data between web browsers and servers. Apart from other forms of information, HTTPS also helps to protect authentication credentials in transit**

**5. Have you built apps using Maven?**

**(Explain in detail and use pseudo code where appropriate)**

**Yes, To build a Maven project via the command line, you use the mvn command from the command line. The command must be executed in the directory which contains the relevant pom file. You pass the build life cycle, phase or goal as parameter to this command.**