1. Write java selenium code for moving a cursor to hover an element.

Ans:

WebElement aravindh = driver.findElement(By.id("hoverElementId"));

Actions actions = new Actions(driver);

actions.moveToElement(aravindh).perform();

1. Write a java program to find second largest number in an array.

Ans:

**int**[] numbers = {34, 34, 54, 12, 67, 89, 11, 32, 67};

**for** (**int** number : array) {

**if** (number > firstLargest) {

secondLargest = firstLargest;

firstLargest = number;

} **else** **if** (number > secondLargest && number != firstLargest) {

secondLargest = number;

}

}

1. Why do we use BDD framework instead of other automation framework and explain?

Ans:

 **Step Definitions:** BDD frameworks allow for the reuse of step definitions across different scenarios. This modularity reduces code duplication and makes the test suite easier to maintain.

 **Reusable Components:** Common actions (like logging in, navigating to a page) can be defined once and reused in multiple scenarios, promoting code reuse and consistency.

1. Static method

Ans:

 **Static Variables:** Shared among all instances of a class.

static int counter = 0;

System.out.println("Static counter: " + Example.counter);

 **Static Methods:** Belong to the class, not instances, and can be called without creating an instance.

public static int add(int a, int b) { return a + b; }

int sum = Example.add(5, 10);

 **Static Blocks:** Used for static initialization tasks and executed when the class is loaded.

static int counter;

// Static block static

{

counter = 10; System.out.println("Static block executed.");

}

public static void main(String[] args) { System.out.println("Counter: " + Example.counter); // Output: 10 }

 **Static Nested Classes:** Nested classes that can be instantiated without an instance of the outer class.

public class OuterClass

{

static class NestedStaticClass

{

void display()

{ System.out.println("Inside static nested class."); }

}

public static void main(String[] args)

{ // Instantiating static nested class

OuterClass.NestedStaticClass nestedObject = new OuterClass.NestedStaticClass(); nestedObject.display(); // Output: Inside static nested class. }

}

5.Javascriptexexutor

Ans:

* Sometimes elements are hidden or not accessible through normal WebDriver interactions. JavaScript can interact with such elements directly
* You can scroll to a specific element or position on the page.
* Custom handling of browser alerts, pop-ups, or custom dialogs.

// Example 1: Click on an element

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].click();", element);

Action class

Ans:

* Hovering, Drag and drop, click and hold ,

Actions actions = new Actions(driver);

//Hover

actions.movetoelement(Aravind).perform();

//Drag and drop

actions.dragAndDrop(sourceElement, targetElement).perform();

//click and hold

Actions.clickandhold(source).movetoelement(target).release().perform();

1. Jenkins pipeline integration

 **Checkout**: Checks out the code from the Git repository.

 **Build**: Builds the project using Maven.

 **Test**: Runs the tests using Maven.

 **Deploy**: (Optional) Deploys the build artifact to a server.

Jenkins is an open-source automation server that helps automate parts of the software development process, including building, testing, and deploying applications. It's a popular tool used in Continuous Integration (CI) and Continuous Deployment (CD) pipelines.

**Install and Configure Jenkins**:

**Create Jobs**:

**Define Pipelines**:

**Execute Builds**: trigger builds based on various events, such as code commits, scheduled intervals, or manual triggers. Builds can include compiling code, running tests, and deploying applications.

**Monitor and Manage**:

Example:

 **Developer Pushes Code**: A developer pushes code changes to a version control repository (e.g., GitHub, GitLab).

 **Jenkins Triggers Build**: Jenkins, configured to monitor the repository, detects the change and triggers a build.

 **Build and Test**: Jenkins executes the build script, compiles the code, and runs automated tests.

 **Report Results**: Jenkins provides feedback on the build and test results. If the build fails, notifications are sent to the development team.

 **Deploy**: If the build and tests are successful, Jenkins can automate the deployment of the application to a staging or production environment.

11. How do you reduce the execution time by enhancing framework – what steps you do to achieve it in testing

**Prioritize Tests**: Focus on the most critical tests that have the highest impact on the system.

**Data-Driven Testing**: Use data-driven approaches to minimize repetitive test code by reusing test scripts with different data sets.

**Parallel Execution**: Use parallel test execution to run multiple tests simultaneously. Tools like Selenium Grid, TestNG, and JUnit support parallel execution.

**Stable Environment**: Ensure that the test environment is stable and consistent to avoid intermittent failures and reruns.

**Select Lightweight Frameworks**: Choose testing frameworks that are lightweight and optimized for speed.

**Optimize Test Scripts**: Ensure that test scripts are efficient and avoid unnecessary waits or delays.

**Use Proper Assertions**: Use efficient assertion mechanisms to avoid performance overhead.

**CI/CD Pipelines**: Integrate testing into CI/CD pipelines to ensure tests are run automatically with each build.

**Lock Dependencies**: Use version control to lock dependencies and avoid changes that could impact test performance.

**Update Dependencies**: Regularly update dependencies to benefit from performance improvements and bug fixes.

12. Types of variables

 **Local Variables**: Declared within methods, constructors, or blocks; scope limited to the block.

 **Instance Variables**: Declared within a class (non-static); each object has its own copy.

 **Class Variables**: Declared with static keyword; shared among all instances of the class.

 **Parameters**: Variables in method or constructor definitions.

 **Constants**: Declared with final keyword; value cannot be changed once initialized.

**Functional Questions:**

1.Entry and exit criteria:

**Entry:**

Collect Requirments

test plan(designs and scripts)

test environment

test data

tools and resources

defect management system

**Exit:**

Test case execution

Pass/fail rate

Defect resolution

Performance testing

User acceptance testing

regression testing

documentation : reports

stakeholder approval

2.Types of testing

**Integeration testing**

To verify the interactions between different components or systems.

Focuses on the interfaces and flow of data between modules.

**System testing: (controlled minimal production environment)**

“Verify system functionality and technical specs”

The primary goal is to validate the end-to-end system specifications. It verifies that the application works as expected in a complete, integrated environment.

System testing covers the entire application, including software, hardware, network, and other interconnected components

**Functional Testing**: Ensures that the application functions according to the requirements.

**Performance Testing**: Evaluates the application's performance under specific conditions, such as load testing, stress testing, and scalability testing.

**Compatibility Testing**: Ensures that the application works across different browsers, operating systems, and devices.

**User acceptance testing(UAT): (Staging environment)**

**“**Validate business requirements and user needs**”**

conducted to ensure that the system meets the business requirements and is ready for deployment. It focuses on validating that the software can handle real-world scenarios and user needs.

It involves testing the system from the end-users' perspective, ensuring that the software meets their expectations and requirements.

3.Defect leakage and example:

Defect leakage is defect not identified at testing phase ,it detected at production environment.

Example: Multiple coupons not able to apply at checkout.

It happens because

* Not covered that scenario while designing
* Human error
* Environment differences
* Time constarints

It can be overcomed by:

* Good test planning and designs
* Similar environment replication with production environment
* Automated testing with complex data and scenarios
* Peer reviews

4. Types of methodology and explain.

* Waterfall Methodology
  + The Waterfall methodology is a linear and sequential approach where each phase of the development process must be completed before the next phase begins. It typically includes the following stages: requirements, design, implementation, testing, deployment, and maintenance.
  + Example:
    - Development of a Payroll Management System for a mid-sized company
      * This will be created and will be maintained or supported later and no new functions will be added.
* Agile Methodology
  + Agile methodology is an iterative and incremental approach that focuses on flexibility, collaboration, and customer feedback. It emphasizes short development cycles (sprints) and frequent releases.
  + Example
    - Development of a Mobile Shopping App for a retail company
      * The app will be developed and will reviewed frequently by owner and functions will be added according to needs.
* Scrum
  + Scrum is a specific Agile methodology that uses fixed-length iterations called sprints, usually lasting two to four weeks. It involves roles like Scrum Master, Product Owner, and Development Team and ceremonies such as sprint planning, daily stand-ups, sprint reviews, and retrospectives.

5.Why Retrospective meeting

A retrospective meeting is a critical component of Agile methodologies, such as Scrum and Kanban. It is a regular meeting where the team reflects on their recent work, discusses what went well, what didn't, and how processes can be improved. This meeting typically takes place at the end of each iteration or sprint.

6. Deferred state and explain defect life cycle.

Deferred State:

In the context of software testing and defect management, a "Deferred" state refers to a status assigned to a defect that indicates it has been acknowledged but will not be addressed or fixed in the current release cycle. This may be due to various reasons such as low priority, limited resources, or planned future enhancements.

Defect Life Cycle:

1. **New**: The tester logs the defect with detailed steps to reproduce, severity, priority, and relevant screenshots.
2. **Assigned**: The project manager reviews the defect and assigns it to a developer for investigation.
3. **Open**: The developer starts analyzing the issue and identifies that a missing database index is causing the problem.
4. **Fixed**: The developer adds the missing index and updates the code to resolve the issue.
5. **Retest**: The tester verifies the fix by retesting the search functionality with the specified keyword.
6. **Verified**: The tester confirms that the search functionality works correctly and the defect is fixed.
7. **Closed**: The defect is marked as closed, indicating it is resolved and no further action is required.

Alternatively, if the defect is deemed low priority and scheduled for a future release, it may enter the **Deferred** state:

1. **Deferred**: The project manager decides to defer the defect due to its low priority and plans to address it in the next release cycle.

7. Risk based testing.

Risk-based testing is a strategic approach to software testing that prioritizes testing efforts based on the level of risk associated with various features, functionalities, or components of the software. The primary goal is to identify and focus on areas that pose the highest risk to the project's success, ensuring that critical issues are discovered and addressed early in the development cycle.

**Example of Risk-Based Testing**

**Project**: Development of an online banking application

**Risk Identification**:

* Functional Risk: Incorrect balance calculations for transactions
* Performance Risk: Slow response times during peak usage
* Security Risk: Vulnerabilities in user authentication
* Business Risk: Non-compliance with financial regulations

**Risk Assessment**:

* **Likelihood** and **Impact**:
  + Incorrect balance calculations: High likelihood, high impact
  + Slow response times: Medium likelihood, high impact
  + Vulnerabilities in authentication: Low likelihood, very high impact
  + Non-compliance with regulations: Low likelihood, very high impact

8. One principle which will not fall under agile methodology.

**Big Design Up Front (BDUF)**

**Definition**:

* BDUF refers to the practice of extensively designing and detailing the entire software architecture and system requirements before any actual development begins. This approach relies on creating a comprehensive and detailed plan at the beginning of the project.
* The project team spends several months in the initial phases, creating detailed design documents, specifications, and project plans. Only after all design documents are approved does the development phase begin. Any changes to the design later in the project are difficult and costly to implement.

Example:

* + - **Extensive Documentation**: A 100-page document outlining every functional and non-functional requirement of the system.
    - **Linear Process**: The entire architecture is locked in after the initial design phase, with limited scope for modifications.
    - **Rigid Planning**: A detailed project plan outlining specific milestones, deliverables, and timelines for each phase of development.
    - **Delayed Feedback**: User feedback is gathered only during the testing phase, after most of the development work is completed.

9.When to do we confirm execution is completed

* Review test plan and objectives
* Check test case execution status
* Defect management
* Test coverage (all functionality covered) analysis
* Final test summary report(overview completion, defects found, resolution).
* Stakeholder review and signoff

10. Difference between SIT and UAT

| **Aspect** | **System Integration Testing (SIT)** | **User Acceptance Testing (UAT)** |
| --- | --- | --- |
| **Objective** | **SIT** ensures that  integrated systems work  together as intended by  focusing on technical  interfaces and data exchange. | **UAT** validates that the software  meets business requirements  and is ready for production  by involving end-users to test real-world scenarios. |
| **Focus** | Technical aspects of integration | Business processes and user requirements |
| **Participants** | Developers, testers, integration team | End-users, business analysts, clients |
| **Environment** | Integration testing environment | UAT environment similar to production |
| **Test Cases** | Based on integration points and interfaces | Based on real-world user scenarios |

11. Scenario Based: Suppose if you find a bug on day before the release, and you have informed your manager regarding this and your manager asked you to not to disclose the information with anyone else let the release happen. In this case what will you do? And manager is aware of risk which is involved

* Assess the bug(severiy)- Consider how it affects functionality, user experience, and business operations.
* Document the bug
* **Written Confirmation**: Request written confirmation from your manager about the decision to proceed with the release despite the known bug. This can be in the form of an email or a meeting note.
* **Escalation**: If the bug is severe and you believe the decision could cause significant harm, consider escalating the issue to higher management or relevant stakeholders.
* **Monitoring**: Closely monitor the system after the release for any issues related to the bug.
* **Plan for Fix**: Prepare a plan to address the bug in the next patch or release cycle.

12.Static Testing:

Static testing is a type of software testing that involves examining the code, requirements, and design documents without actually executing the code. The main goal of static testing is to identify potential defects early in the development lifecycle, which can help reduce the cost and effort required to fix these defects later on.

* Requirments Review – it is co related with requirments in documentation
* Design Review - Analyze the design documents, including architecture diagrams and data flow diagrams
* Code Review - Conduct code reviews using both automated tools and manual inspections to identify potential defects, adherence to coding standards, and opportunities for optimization.
* Peer review
* Inspection- Organize formal inspections for critical components of the application, such as security modules, to ensure that they meet the highest standards of quality and reliability.

13.Who will do end to end testing and how to do you categorize to complete end to end testing with all scenarios how do you handle

E2E Testing will be done by QA testing team

* **Core Functional Scenarios**: User registration and login.
* **User Interface Scenarios**: Navigation through different product categories.
* **Data Flow and Integrity Scenarios**: Correct order details in the database.
* **Security and Access Control Scenarios**: Secure user authentication and authorization.
* **Performance and Load Scenarios**: Load testing during peak shopping hours.
* **Integration Scenarios**: Interaction with payment gateway.

14. White Box and blackbox testing

 **White Box Testing**: Focuses on internal code and logic, ensuring thorough coverage and early detection of defects. It requires detailed knowledge of the internal workings of the application.

Example : Factorial number – need to understand the code and write and execute the test cases

 **Black Box Testing**: Focuses on the external functionality and user perspective, ensuring the application behaves as expected based on requirements. It does not require knowledge of the internal code, making it more accessible to non-developers.

Example : Testing login features –

Create test cases for valid login, invalid login, empty input, and boundary values and check the behaviour and output.

1. Write a program using Java and Selenium to switch to new window and capture multiple texts and validate it.

// Get the handle of the original window

String originalWindow = driver.getWindowHandle();

// Get all window handles

Set<String> allWindows = driver.getWindowHandles();

// Switch to the new window

for (String windowHandle : allWindows)

{

if (!windowHandle.equals(originalWindow))

{

driver.switchTo().window(windowHandle); break;

}

}

 **String Representation**: Window handles are represented as strings because they are unique identifiers managed by the browser.

 **Set Collection**: Using a Set ensures that all window handles are unique, which matches the requirement for unique browser window identifiers.

2.Print even digit words in a string