SeU™ Certified Selenium Engineer (CSE)
Sample Exam – 10 Questions with Answers

Released
Version 2018 Syllabus

Selenium United

Purpose of this document

This document contains 10 sample exam questions for SeU Certified Selenium Engineer (CSE) in the English language.

The sample questions, answer sets and associated justifications in this document have been created by a team of subject matter experts and experienced question writers with the aim of assisting people who are planning to take the SeU Certified Selenium Engineer (CSE) examination.

None of these questions are used in the official SeU Certified Selenium Engineer (CSE) examination, but they are written to the same level of difficulty as the official certification exam.

Instructions

The question and answer sets are organized in the following way:

- Learning Objectives / Chapters
- Question - including any scenario followed by the question stem
- Answer Set

General Information on the sample exam paper:

- Number of Questions: 10
- Number of points: 1 per question
- Please only choose one answer per question.

List of Chapters

Main list of LOs for the SeU CSE certification:

LO1	Understand the importance of browser coverage and distinguish various options for testing UI of web applications. (K2)
LO2	Understand the relationship of Web UI with the underlying HTML and JavaScript with DOM Inspection. (K2)
LO3	Recall the History of Selenium and various tools in its suite along with their purpose. (K1)
LO4	Understand the Selenium Architecture in terms of Language Bindings, Communication Protocols and Drivers. (K2)
LO5	Recognize the purpose and API of Web UI Automation at Browser, Page, Element levels. (K2)
LO6	Understand and Explain the purpose of JUnit annotations, fixtures, assertions. (K3)
LO7	Apply different identification strategies for UI elements. (K3)
LO8	Apply different inquiry and interaction strategies for UI elements. (K3)
LO9	Apply various deeper Selenium automation constructs, which set the basis for more advanced automation. (K3)
LO10	Automate end to end user scenarios by using good coding practices and Object-Oriented principles for placing Selenium code across different classes and methods. (K3)
LO11	Troubleshoot and describe scope of improvement for automation implementation depicted in short code samples. (K3)

Question 1

(Correct answer is worth 1 point)

Which of the following styling properties can be used to determine whether an element is visible?

- ☐ (a) `view`
- ☐ (b) `visible`
- ☐ (c) `display`
- ☐ (d) `Hide`

Question 2

(Correct answer is worth 1 point)

Which of the following statements is **TRUE** about Selenium?

- ☐ (a) Selenium Grid is used to develop Selenese based browser automation code.
- ☐ (b) Selenium needs browser specific executables that support standard JSON wire protocol.
- ☐ (c) Selenium drivers use JSON wire protocol to communicate with browsers.
- ☐ (d) Selenium supports only object-oriented programming languages for developing its client bindings.

Question 3

(Correct answer is worth 1 point)

Which of the following statements is **FALSE**? (Assume that `driver` is a `WebDriver` object)

- ☐ (a) `By.className` supports compound classes.
- ☐ (b) One cannot use Xpath's `ends-with` function to identify elements because it is not supported by most of the browsers.
- ☐ (c) `driver.switchToParentFrame()` switches the search context from a child frame to the parent frame.
- ☐ (d) `HtmlUnitDriver` does not need any driver executable.

Question 4

(Correct answer is worth 1 point)

Given the following HTML form snippet:

```
<form action='abc'>
  <label>Type of vehicle</label>
  <select id = "vehicle_type">
     <option value = "2-wheeler">2 Wheeler</option>
     <option value = "3-wheeler">3 Wheeler</option>
     <option value = "4-wheeler">4 Wheeler</option>
  </select>
  <input id='B' type='submit'>
</form>
```

what would the following Java code do? (Assume that `driver` is a `WebDriver` object)

```
driver.findElement(By.id("vehicle_type")).sendKeys("3-wheeler");
```

- ☐ (a) Vehicle type is selected as “3 Wheeler” value from the drop down.
- ☐ (b) Vehicle type is unchanged as `sendKeys` doesn’t work with `select` tag.
- ☐ (c) Vehicle type is unchanged as in this case, `sendKeys` expects visible text.
- ☐ (d) Selenium throws `NoSuchElementException`.

Question 5

(Correct answer is worth 1 point)

In a given application, the home page includes quite a few `<script>` tags for large external JavaScript files. Hence, depending on network conditions, the browser spends 45-90 seconds on loading and rendering the home page. How would you implement synchronization to handle this situation?

- ☐ (a) Will implement a static wait for 90 sec.
- ☐ (b) Will use an implicit wait for 90 sec.
- ☐ (c) Will implement a systematic event based `while` loop.
- ☐ (d) Will use explicit wait for 90 sec.

Question 6

(Correct answer is worth 1 point)

Which of the following statements is **TRUE** when an alert is open in a browser? (Assume that `driver` is a `WebDriver` object)

I. `driver.findElements` throws an exception
II. `driver.close` throws an exception
III. `driver.close` closes the current window.
IV. `driver.findElements` returns an empty list.

- ☐ (a) I, II.
- ☐ (b) I, III.
- ☐ (c) II, III.
- ☐ (d) III, IV.

Question 7

(Correct answer is worth 1 point)

What would be the outcome of the following Java code snippet for Selenium? Assume that:

- `driver` is a `WebDriver` object.
- `wait` is a `WebDriverWait` object.
- `menu` is a parent menu `WebElement` object.
- `subMenu` is a sub-menu `WebElement` object which appears after hovering on the parent menu.
- `editName` is a text field with id “edit” in a page that appears when the above-mentioned sub-menu is clicked.

```
Actions actionBuilder = new Actions(driver);
actionBuilder.moveToElement(menu).click(subMenu).build();
WebElement editName = wait.until(
     ExpectedConditions.presenceOfElementLocated(By.id("edit"))
);
editName.sendKeys("Updated");
```

- ☐ (a) Enters “Updated” in `editName` field
- ☐ (b) Throws `NoSuchElementException`
- ☐ (c) Throws `TimeoutException`
- ☐ (d) Fails while clicking the `subMenu`

Question 8

(Correct answer is worth 1 point)

Out of the following, which statements represent **PREFERRED** code refactoring of the following basic `WebAutomator` class for wrapping Selenium `WebDriver`?

```
public class WebAutomator{
   WebDriver driver;
   public WebAutomator(WebDriver wd){
      driver = wd;
      WebDriverWait waiter = new WebDriverWait(driver, 15);
   }
   public void click(By by) throws Exception{
      WebDriverWait waiter = new WebDriverWait(driver, 15);
      waiter.until(
          ExpectedConditions.elementToBeClickable(by)
       ).click();
   }
}
```

I. The `waiter` variable should be declared as an instance variable.
II. The `waiter` variable should be declared as a `static` variable.
III. The `waiter` variable should be assigned value in constructor.
IV. The `waiter` variable should be assigned value in `click` method.

☐ (a) I, III.

☐ (b) II, IV.

☐ (c) I, IV.

☐ (d) II, III.

Question 9

(Correct answer is worth 1 point)

Which of the following locator strategies is used to identify parent element from child element?

- ☐ (a) Class Name.
- ☐ (b) CSS Selector.
- ☐ (c) XPath.
- ☐ (d) Nested Element finding.

Question 10

(Correct answer is worth 1 point)

John and Carter want to implement a test fixture strategy in a JUnit5 test class. Which of the following options about test fixture strategies are **CORRECT** if login is a pre-requisite and logout is a clean-up activity?

I. `BeforeAll` method launches browser; `AfterAll` method quits browser.
II. `BeforeAll` method launches browser & performs login; `AfterAll` method performs logout & quits the browser.
III. Class constructor launches the browser; `BeforeAll` method performs login; `AfterAll` method performs logout & quits the browser.
IV. `BeforeEach` launches browser & performs login; `AfterEach` performs logout and quits the browser.

- ☐ (a) I, III.
- ☐ (b) II, III.
- ☐ (c) II, IV.
- ☐ (d) I, IV.

Answer Key:

Question 1: Answer C is correct

- Web elements have multiple properties that determine their behavior. The property that determines the visibility of an element is its 'display' property.

Question 2: Answer B is correct

- Option A is wrong because Selenium Grid's primary purpose is to enable parallel execution of browser automation. Selenese is the DSL supported by Selenium IDE. Option B is right. Option C is wrong because the driver executables use browser specific native API to communicate with browsers. Option D is wrong because Selenium WebDriver does not put any constraints on the type of programming language to be used.

Question 3: Answer A is correct

- Option A is wrong because `By.className` does not support compound classes. The other options are all correct statements.

Question 4: Answer C is correct

- If a drop-down list is implemented in HTML using its standard `<select>` tag, one can select one of its options using `WebElement`'s `sendKeys` method. In doing so, one must use the visible text corresponding to the target option.

Question 5: Answer D is correct

- Explicit waits are the suggested waiting mechanism.

Question 6: Answer B is correct

- In this case `driver.findElements` method would throw an exception, however `driver.close` would close the window as expected.

Question 7: Answer C is correct

- Here the `perform()` call is missing for the compound action, because of which the compound action is not executed. Because of this the expected page would not be loaded. This would make the explicit wait not able to locate the element for the complete wait duration and throw a `TimeoutException`

Question 8: Answer A is correct

- Given the design of this class, the `waiter` variable would be needed across multiple methods when more methods are added. Hence it should be made an instance variable and initialized one time in the constructor.

Question 9: Answer C is correct

- Only one locator strategy in Selenium allows to move back in the DOM out of the provided options – XPath (using .. or /ancestor)

Question 10: Answer C is correct

- Junit fixtures are typically used in corresponding pairs for correct state setup and clean-up. In this case, statements II and IV depict the correct choice of Junit fixture combinations.