+91 8318815041
chaitanyaa292@gmail.com

Exceptions in Java

What are Exceptions in Java?

An **exception** in Java is an event that disrupts the normal flow of the program during runtime.

Types of Exceptions

1. **Checked Exceptions:**
   - Checked at **compile-time**.
   - The compiler forces you to handle them using `try-catch` or `throws`.
   - **Examples:** `IOException`, `SQLException`.
2. **Unchecked Exceptions:**
   - Checked at **runtime**.
   - The compiler doesn't force you to handle them.
   - **Examples:** `NullPointerException`, `ArrayIndexOutOfBoundsException`.
3. **Errors:**
   - Serious problems that applications should not try to handle.
   - **Examples:** `OutOfMemoryError`, `StackOverflowError`.

Handling Exceptions in Selenium

1. try-catch in Selenium

Use Case: Trying to click a button that might not be present.

```
try {
    WebElement loginBtn = driver.findElement(By.id("login"));
    loginBtn.click();
} catch (NoSuchElementException e) {
    System.out.println("Login button not found on the page.");
}
```

📝 **Explanation:** If the element is not found, `NoSuchElementException` is caught and handled.

---

2. finally in Selenium

Use Case: Closing the browser regardless of test result.

```
WebDriver driver = new ChromeDriver();

try {
    driver.get("https://example.com");
    System.out.println("Title: " + driver.getTitle());
} catch (Exception e) {
    System.out.println("Error: " + e.getMessage());
} finally {
    driver.quit();  // Ensures browser is closed
}
```

📝 **Explanation:** `finally` ensures the browser closes whether an exception occurs or not.

---

3. throw in Selenium

Use Case: Manually throwing an exception when title validation fails.

```
String expectedTitle = "Dashboard";
String actualTitle = driver.getTitle();

if (!actualTitle.equals(expectedTitle)) {
    throw new RuntimeException("Page title mismatch! Expected:
" + expectedTitle);    }
```

📝 **Explanation:** If the condition fails, we throw an exception manually to stop test execution.

---

4. throws in Selenium

Use Case: Declaring that a method may throw `InterruptedException` during wait.

```
public void waitBeforeClick() throws InterruptedException {
    Thread.sleep(3000);  // may throw InterruptedException
    driver.findElement(By.id("continue")).click();
}
```

📝 **Explanation:** `throws` is used to declare that the method may throw `InterruptedException`.

Java Exception Hierarchy

```
Object
└── Throwable
    ├── Error                          ← Serious problems (not
meant to handle)
    │   ├── OutOfMemoryError
    │   └── StackOverflowError
    │
    └── Exception                      ← Handleable exceptions
        ├── Checked Exceptions        ← Must be handled using
try-catch or throws
        │   ├── IOException
        │   ├── SQLException
        │   ├── FileNotFoundException
        │   └── InterruptedException
        │
        └── Unchecked Exceptions      ← Runtime exceptions
            ├── RuntimeException
            │   ├── NullPointerException
            │   ├── ArithmeticException
            │   ├── ArrayIndexOutOfBoundsException
            │   ├── IllegalArgumentException
            │   └── NumberFormatException
```

NullPointerException = when accessing null variable
ArithmeticException = when dividing a number by 0
ArrayIndexOutOfBoundsException = when accessing element out of the size of array

Selenium Exception Hierarchy (Graph)

```
RuntimeException
└── WebDriverException
    ├── NoSuchElementException
    ├── TimeoutException
    ├── ElementNotVisibleException (Deprecated → use
ElementNotInteractableException)
    ├── ElementNotInteractableException
    ├── NoSuchWindowException
    ├── NoSuchFrameException
    ├── StaleElementReferenceException
    ├── InvalidSelectorException
    ├── MoveTargetOutOfBoundsException
    └── SessionNotCreatedException
```

Selenium Exceptions in Detail

1. RuntimeException

- **Occurs:** General-purpose exception for runtime logic issues.
- **Example:** Invalid data or unexpected conditions.

Handling:

```
try {
    // Code that could throw a runtime exception
} catch (RuntimeException e) {
    System.out.println("Runtime Exception: " +
e.getMessage());
}
```

2. WebDriverException

- **Occurs:** WebDriver-related issues.
- **Example:** Browser connection problems.

Handling:

```
try {
    // WebDriver code
} catch (WebDriverException e) {
    System.out.println("WebDriver Exception: " +
e.getMessage());
}
```

3. NoSuchElementException

- **Occurs:** When WebDriver can't find an element using the specified locator.
- **Example:** Trying to interact with a non-existing element.

Handling:

```
try {
    WebElement element =
driver.findElement(By.id("loginBtn"));
} catch (NoSuchElementException e) {
    System.out.println("Element not found: " +
e.getMessage());
}
```

4. TimeoutException

- **Occurs:** When an action exceeds the timeout period.
- **Example:** Waiting for an element that never loads.

Handling:

```
WebDriverWait wait = new WebDriverWait(driver, 10);
try {
    WebElement element =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id
("login")));
} catch (TimeoutException e) {
    System.out.println("Timeout occurred: " + e.getMessage());
}
```

5. ElementNotVisibleException (Deprecated → ElementNotInteractableException)

- **Occurs:** When an element is in the DOM but not visible.
- **Example:** Trying to click an element that is hidden.

Handling:

```
try {
    WebElement element =
driver.findElement(By.id("hiddenElement"));
    element.click();
} catch (ElementNotVisibleException e) {
    System.out.println("Element is not visible: " +
e.getMessage());
}
```

6. ElementNotInteractableException

- **Occurs:** When an element is visible but can't be interacted with (disabled).
- **Example:** Trying to click a disabled button.

Handling:

```
try {
    WebElement element =
driver.findElement(By.id("submitButton"));
    element.click();
} catch (ElementNotInteractableException e) {
    System.out.println("Element not interactable: " +
e.getMessage());
}
```

7. NoSuchWindowException

- **Occurs:** When WebDriver tries to switch to a non-existent window.
- **Example:** Switching to a closed window.

Handling:

```
try {
    String windowHandle = driver.getWindowHandle();
    driver.switchTo().window(windowHandle);
} catch (NoSuchWindowException e) {
    System.out.println("No such window: " + e.getMessage());
}
```

8. NoSuchFrameException

- **Occurs:** When WebDriver tries to switch to a non-existent frame.
- **Example:** Switching to an incorrect iframe.

Handling:

```
try {
    driver.switchTo().frame("frameName");
} catch (NoSuchFrameException e) {
    System.out.println("No such frame: " + e.getMessage());
}
```

9. StaleElementReferenceException

- **Occurs:** When an element is no longer attached to the DOM.
- **Example:** Interacting with a removed or refreshed element.

Handling:

```
try {
    WebElement element = driver.findElement(By.id("item"));
    element.click();
} catch (StaleElementReferenceException e) {
    System.out.println("Element is no longer available: " +
e.getMessage());
}
```

10. InvalidSelectorException

- **Occurs:** When a CSS or XPath selector is invalid.
- **Example:** Using incorrect selector syntax.

Handling:

```
try {
    WebElement element =
driver.findElement(By.xpath("//div[@class='invalid-class']"));
} catch (InvalidSelectorException e) {
    System.out.println("Invalid selector: " + e.getMessage());
}
```

11. MoveTargetOutOfBoundsException

- **Occurs:** When trying to move the mouse pointer outside the viewport.
- **Example:** Using `Actions.moveToElement()` when the element is out of bounds.

Handling:

```
Actions action = new Actions(driver);
try {
    WebElement element =
driver.findElement(By.id("outOfBoundsElement"));
    action.moveToElement(element).perform();
} catch (MoveTargetOutOfBoundsException e) {
    System.out.println("Target element is out of bounds: " +
e.getMessage());
}
```

12. SessionNotCreatedException

- **Occurs:** When WebDriver fails to create a new session.
- **Example:** Driver binary incompatibility.

Handling:

```
try {
    WebDriver driver = new ChromeDriver();
} catch (SessionNotCreatedException e) {
    System.out.println("Session could not be created: " +
e.getMessage());
}
```

General Handling Strategy for Selenium Exceptions:

- **Explicit Waits:** Use `WebDriverWait` to ensure elements are visible or interactable.
- **Try-Catch:** Always handle exceptions and log meaningful messages, especially for flaky tests.
- **Element Location:** Use robust locators and validate elements before interacting with them.
- **Browser Configuration:** Ensure that browser drivers match the browser version.
- **Logging:** Log exceptions for easier debugging and analysis.
