CS608
Software Testing

Dr. Stephen Brown

Room Eolas 116

stephen.brown@mu.ie

Tutorial: Lab 7

- OO/state-based testing for Eshopping
- All-transitions strategy:
  - Task 1: unique signature table
  - Task 2: state table
  - Task 3: initial TCI
  - Task 4: checkState()
  - Task 5: Test Cases
  - Task 6: complete TCI table
  - Implementation

CS608

Application Testing

(Testing Web Applications with User Stories)

(Essentials of Software Testing, Chapter 10, Sections 10.1-10.3)

(Web) Application Testing

- Today, we look at testing a web application
- The testing of desktop and mobile applications is very similar:
  - the key difference is the test automation tools

(Web) Application Testing

- Today, we look at testing a web application
- The testing of desktop and mobile applications is very similar:
  - the key difference is the test automation tools
- Three additional complexities compared to unit testing:
  1. How to locate the inputs on the screen
  2. How to locate the outputs on the screen
  3. How to automate the tests running over the user interface

Testing Web Applications with User Stories

- Many different elements of an application's specification can be used as the test basis for application testing
- Modern, Agile development processes focus on the user requirements expressed in the form of *user stories*
- Larger systems may contain a number of stories which can be grouped into collections called *epics*
- An alternative is UML Use-Case Diagrams
- Or *ad-hoc* specifications – often sketches of each screen and a brief description of required behaviour

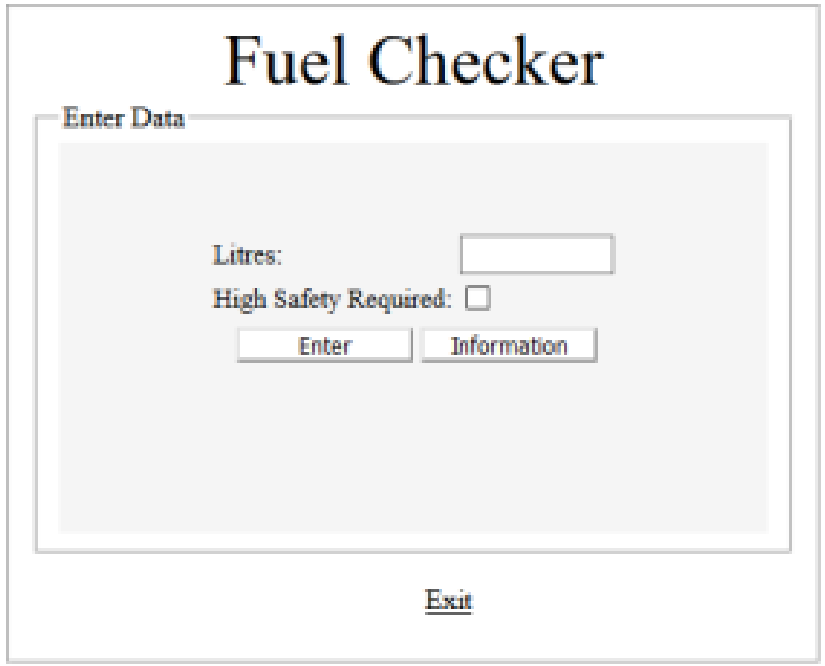
Definition: User Story

- User stories usually take the form:
  - **As a <*role*>, I want <*to do something*> so that I can <*achieve a goal*>**
- A stakeholder in a project may be an end user, the sponsor of the project, a representative from sales or marketing, etc.
- The *role* refers to the part that the stakeholder is playing for that story

Acceptance Criteria or *Confirmations*

- Each user story is detailed with acceptance criteria (also called confirmations) which the stakeholders will use to verify that the system has been designed and implemented correctly
- These acceptance criteria provide the basis for test cases for automated tests for the stories
- We will first consider a small example, and then discuss some of the underlying principles and issues in more detail

Example: Fuel Checker



- A fuel depot wants a web-based system to enable the dispatcher to determine whether a load of fuel to be received at the depot will fit in a single tank
- Low volatility fuels can fill a tank completely
- High-volatility fuels require expansion space to be left for safety reasons
- Tank capacity is:
  - 1200 litres without the expansion space
  - 800 litres with the expansion space
- All loads are considered to the nearest litre (no decimal points)

User Story S1

- Following conversations with the user, the following user story has been developed

- Story S1

  As a fuel depot dispatcher, I want to check if a fuel load fits in a tank so I can decide whether to accept it or not

User Story S1

- Following conversations with the user, the following user story has been developed

- Story S1

  <role>

  As a fuel depot dispatcher, I want to check if a fuel load fits in a tank so I can decide whether to accept it or not

User Story S1

- Following conversations with the user, the following user story has been developed

- Story S1

<role> *<to do something>*

As a fuel depot dispatcher, I want to check if a fuel load fits in a tank so I can decide whether to accept it or not

User Story S1

- Following conversations with the user, the following user story has been developed

- Story S1

<role> *<to do something>*

As a fuel depot dispatcher, I want to check if a fuel load fits in a tank so I can decide whether to accept it or not

<to achieve a goal>

User Story S1 Acceptance Criteria

- Acceptance criteria agreed with the customer
  - S1A1 Check a low volatility fuel load that fits in a tank
  - S1A2 Check a high volatility fuel load that fits in a tank
  - S1A3 Check a low volatility fuel load that does not fit in a tank
  - S1A4 Check a high volatility fuel load that does not fit in a tank
  - S1A5 List tank capacities
  - S1A6 Exit when done (a general requirement for their software)
  - S1A7 Identify a user input data error (suggested to the customer by the development team, based on their experience in web application design)
- One user story, and seven acceptance criteria

Steps

- Analysis
- Test Coverage Items
- Test Cases
- Verifying the Test Cases
- Implementation
- Reviewing the results

App Analysis

- Identify three things:
  - The different **screens** the application presents
  - The **user interface elements** on each screen we need to interact with
  - How input and output **data is represented** on the screen

- Note: the interface is quite different from the programming interfaces we have considered in unit testing and OO testing – we will discuss this later!

Trial Runs

- In most cases, the user interface can be most easily investigated by using **trial runs** of the software to determine how each story is achieved
- An example of this follows for the Fuelchecker application described above
- Each screen in the trial is shown below along with a brief explanation
- Note: we are not following a user story, but trying to explore every aspect of the interface
- Note: a well specified system will allow you to do this via the specification (of the app screens & their functionality)

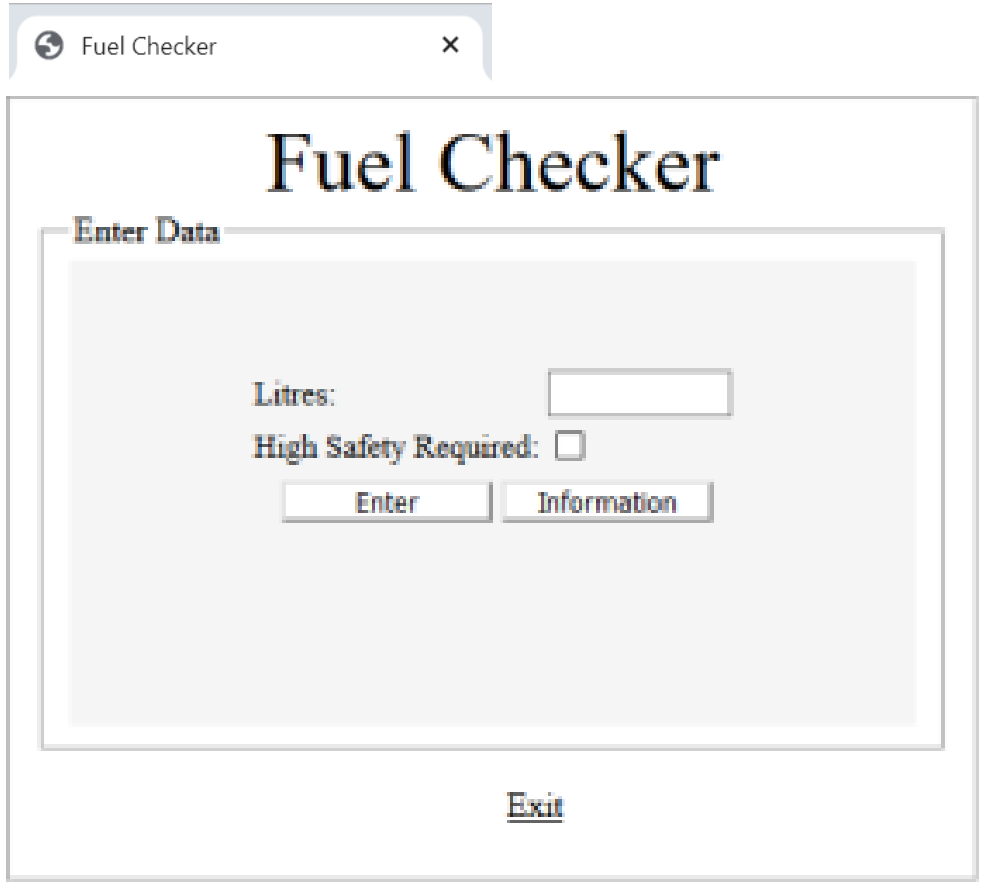
RUN FUELCHECKER

(a) Enter Data Screen

- At startup, the Enter Data screen is displayed
- We need to keep track of the HTML page titles
- Next: the user clicks on the

  Information

  button



(a) Title: Fuel Checker

(b) Information Screen

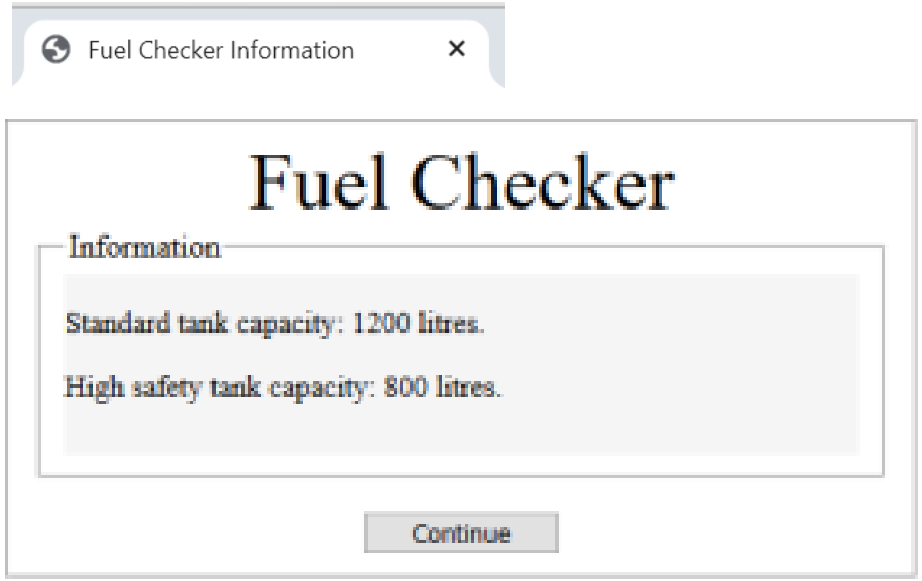
- After clicking on the

  Information

  button the Information screen is displayed
- Next: the user clicks on the

  Continue

  button



(b) Title: Fuel Checker Information

(c) Return to Enter Data Screen

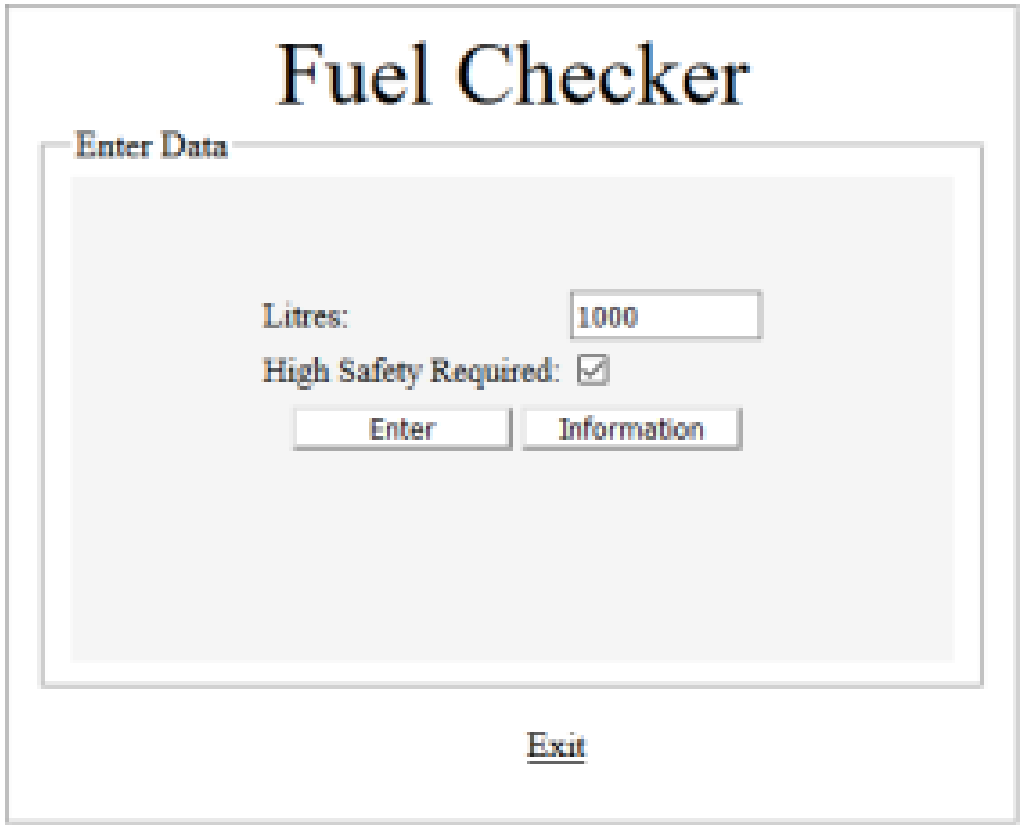
- After clicking on the

  Continue

  button the Enter Data screen is redisplayed
- Next the user:
  - enters 1000 for Litres
  - selects High Safety Required
  - Clicks on the

    Enter

    button



(c) Title: Fuel Checker

(d) Results Screen

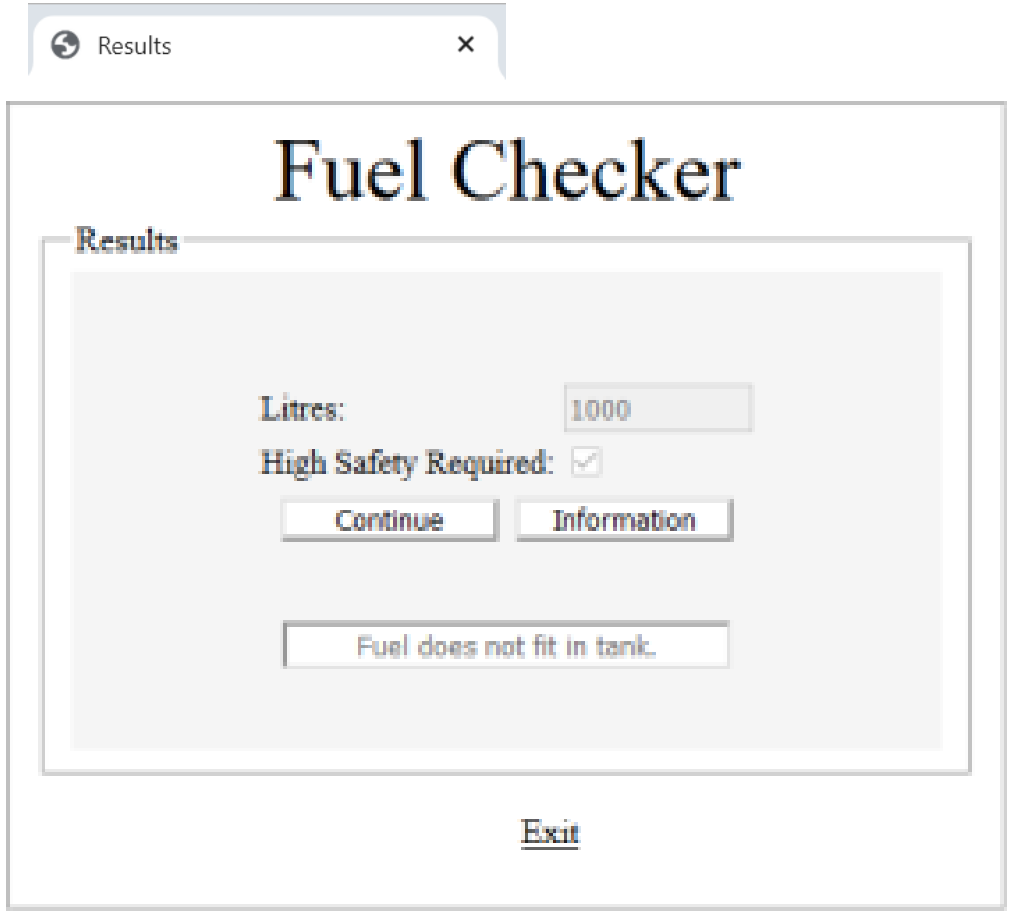
- After clicking on the

  Enter

  button the Results screen is displayed with the message "Fuel does not fit in tank."
- Next: the user clicks on the

  Continue

  button



(d) Title: Results

(e) Return to Enter Data Screen

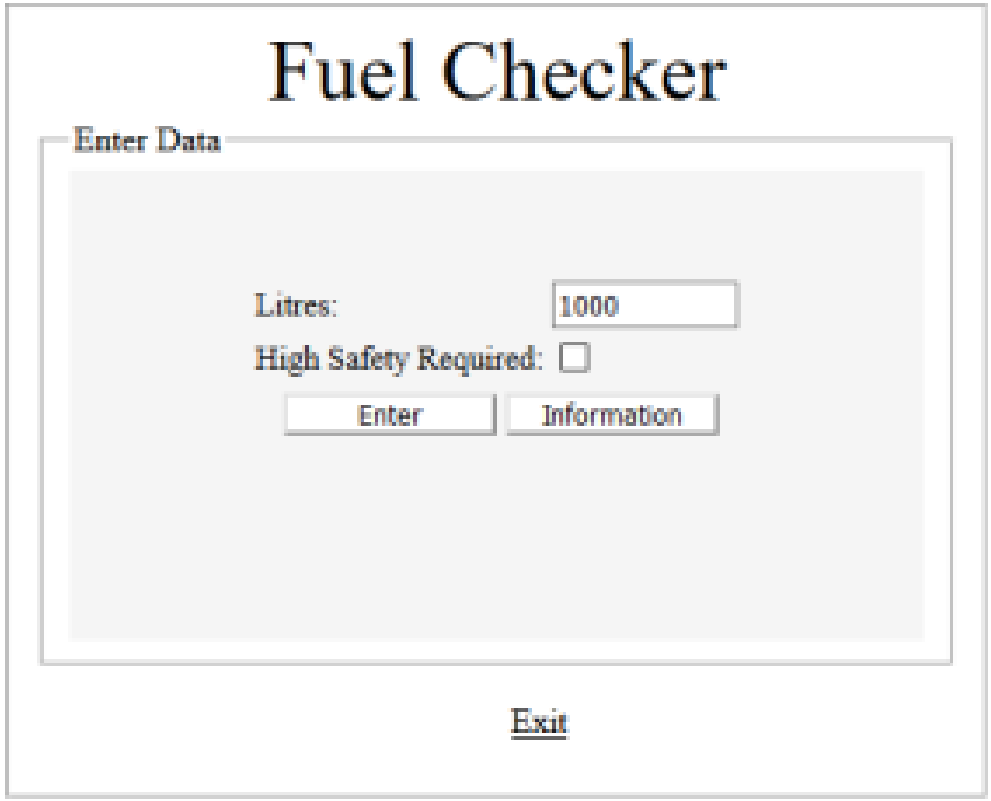
- After clicking on the

  Continue

  button the Enter Data screen is redisplayed with Litres and High Safety cleared
- Next the user:
  - Re-enters 1000 for Litres
  - Clicks on the

    Enter

    button



(e) Title: Fuel Checker

(f) Results Screen

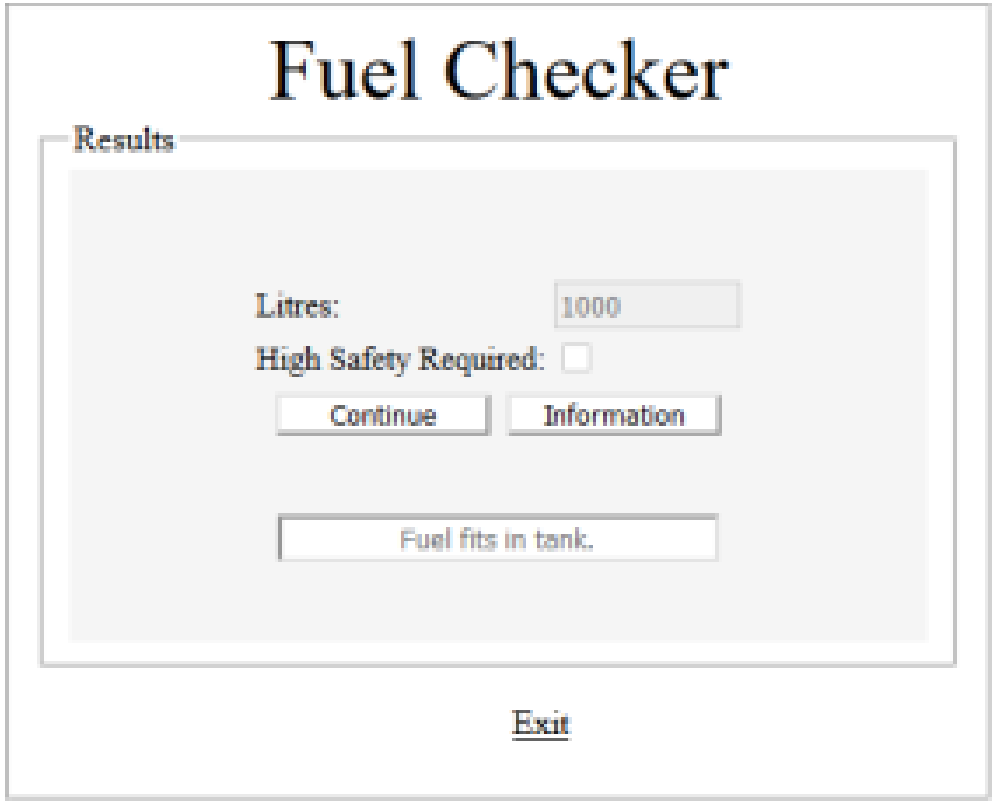
- After clicking on the

  Enter

  button the Results screen is displayed with the message "Fuel fits in tank."
- Next: the user clicks on the

  Continue

  button



(f) Title: Results

(g) Return to Enter Data Screen

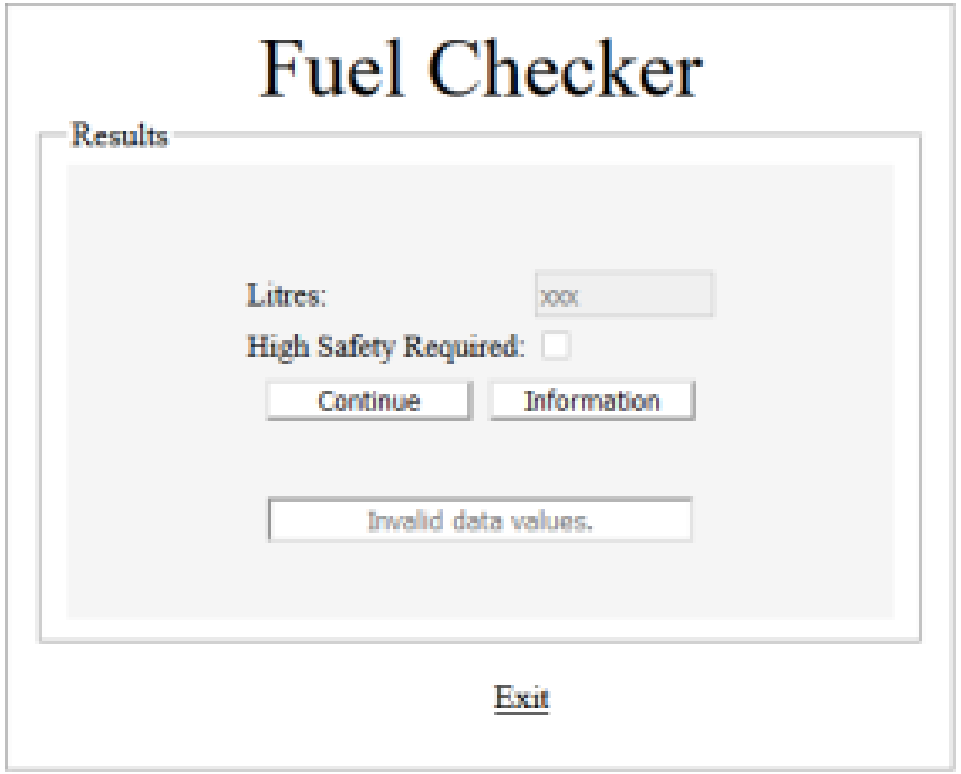
- After clicking on the

  Continue

  button the Enter Data screen is redisplayed
- The user:
  - Enters xxx for Litres
  - Clicks on the

    Enter

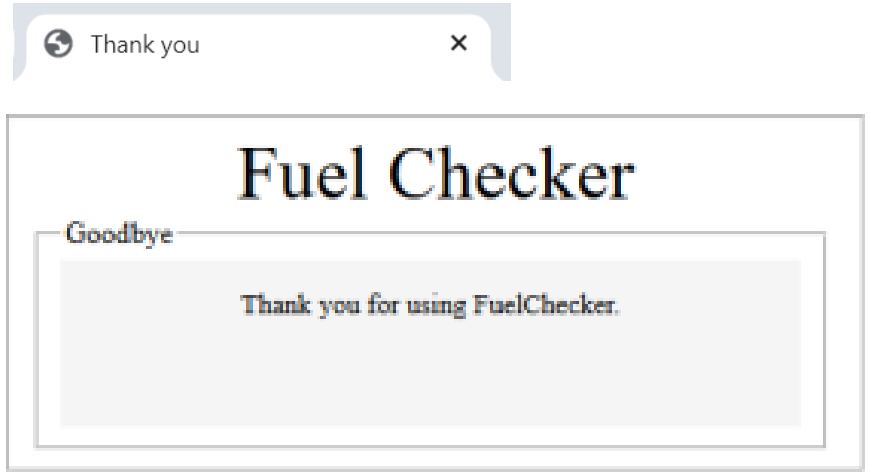
    button
- The Results screen shows “Invalid data values.”



(g) Title: Fuel Checker

(h) Thank You Screen

- After clicking on Continue to return to the Enter Data screen, the user clicks on the Exit link at the bottom of the screen
- The Goodbye screen is now displayed, with the message "Thank you for using FuelChecker."



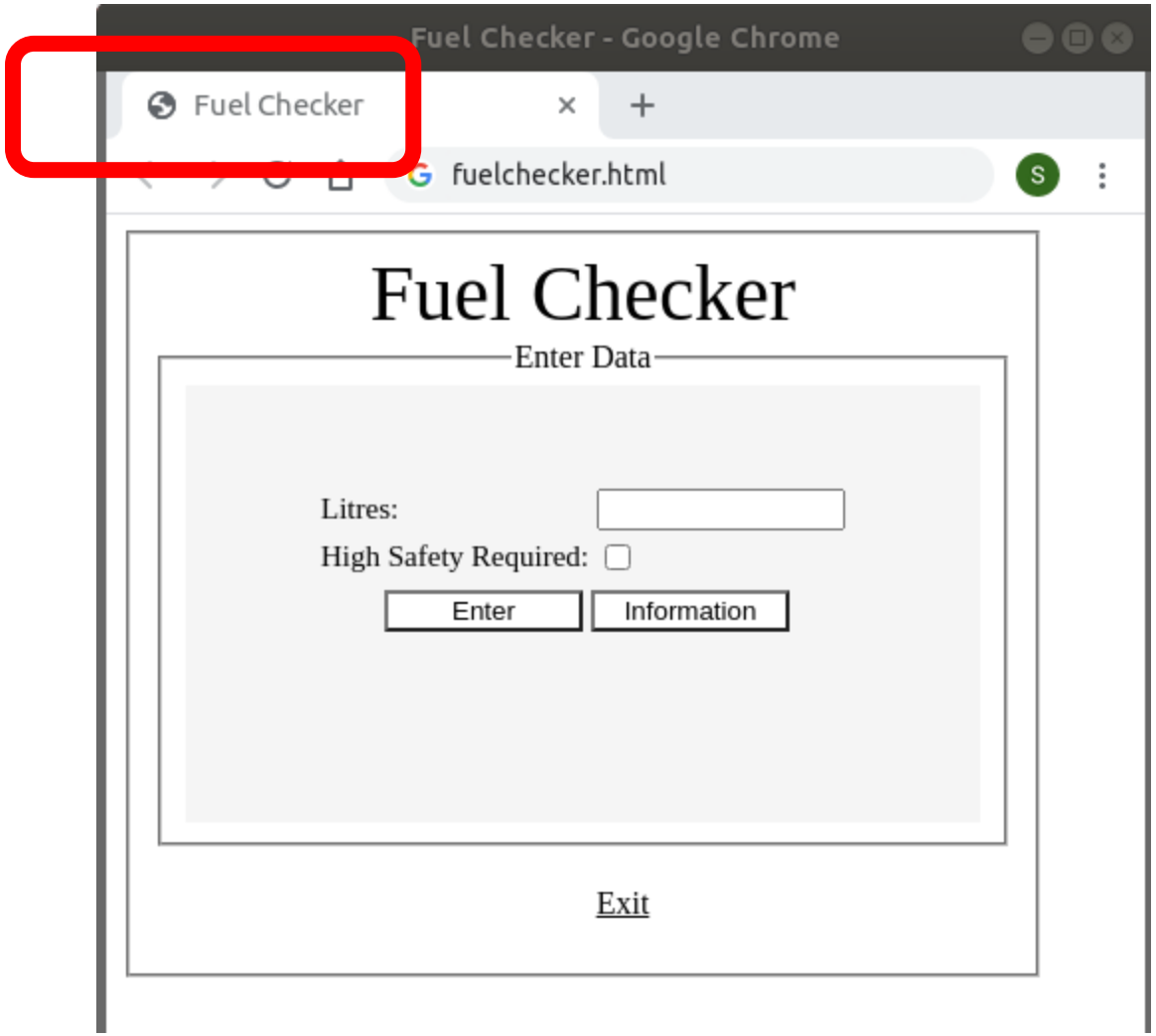
(h) Title: Thank you

Information from the Trial Runs

- From these trial runs we can now identify:
  - Each **page** (or screen) displayed by the application
  - The interface **components** required for testing
  - The data **representation** being used
- Use trial runs when these are not specified in detail (often the case):
  - The navigation between screens
  - The behaviour of each screen
- Risks:
  - The app may not be correctly designed or implemented
  - So the trial runs may not be the correct behaviour…

Page Titles

- The page titles can be seen in the web browser
- They are usually displayed in the tab name
- And can be easily read from the screen

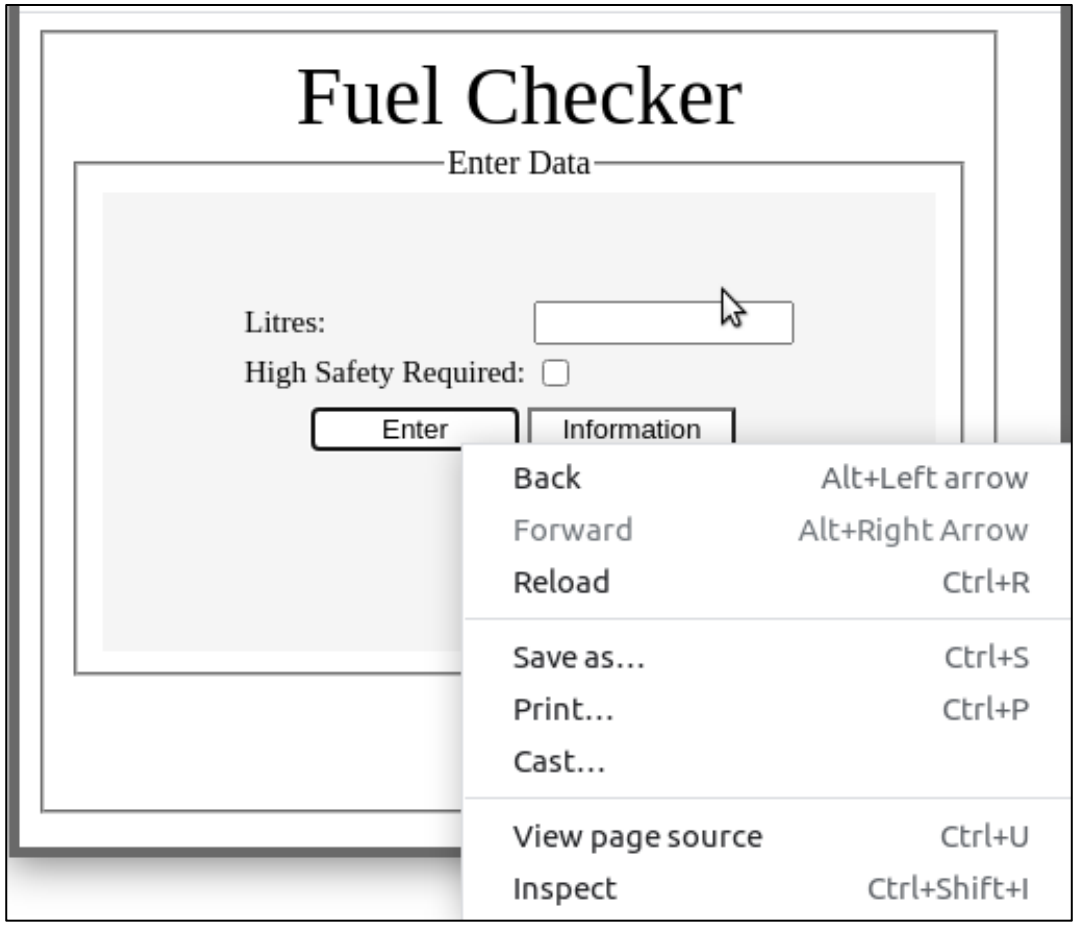


Web Element IDs

- A well designed web page uses unique HTML id attributes to identify each element
- These trial runs allow the tester to discover the id's required to interact with the input and output elements on each web page
- Without these, automated testing is much more difficult
- In a Test Driven Design (TDD) environment, tests may be developed as soon as the screen layout has been designed, using id's selected by the Graphical User Interface (GUI) designer or the tester. The code would then use these ID's in order to pass the tests

Browser Inspector

- Most web browsers include an inspector that allows the element id (and other information) to be examined in the web browser
- Example shown: right-click in the box to right of "**Litres:**"
- Or, the page source may be viewed in the browser



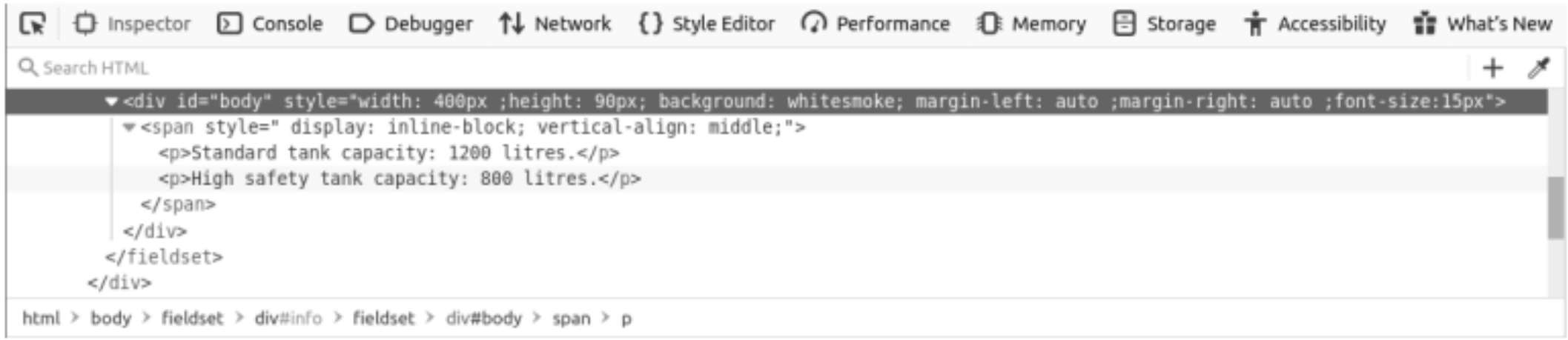
Example 1: Litres Input Textbox

- The important details of this element for the tester are:
- The html element type:
  - <input type="text">
- The ID:
  - id="litres"

Example 2: Enter Button

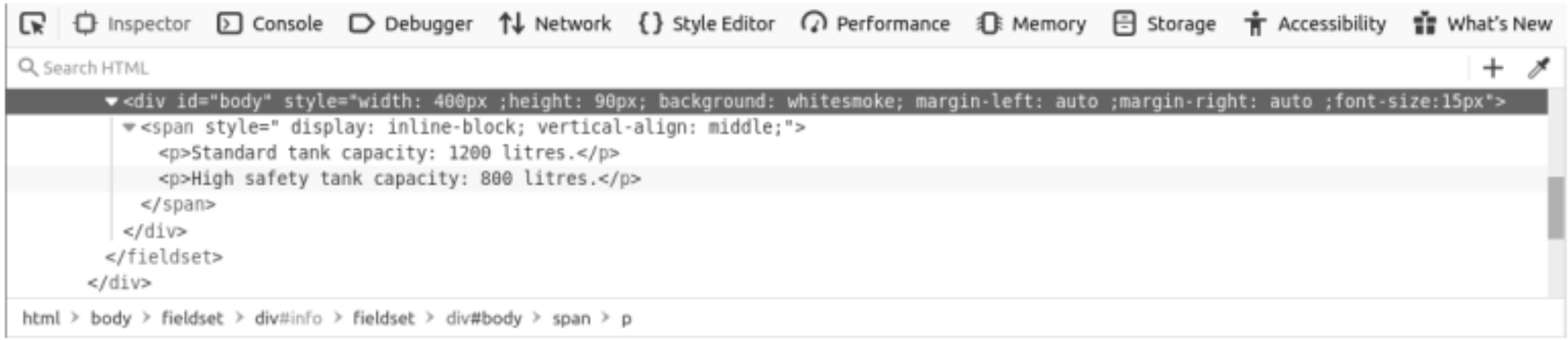
- The important details of this element for the tester are:
- The html element type:
  - <button type="button">
- The ID:
  - id="Enter"

Example 3: Hierarchical Elements



- In cases where displayed text is not within a container with an id, then a higher level container may be used (nested HTML elements)
- For example, the lines of displayed text for the Information screen are contained within <p> elements, which are in turn contained within a element with no id
- The element is in turn contained with a element which has an id "body"

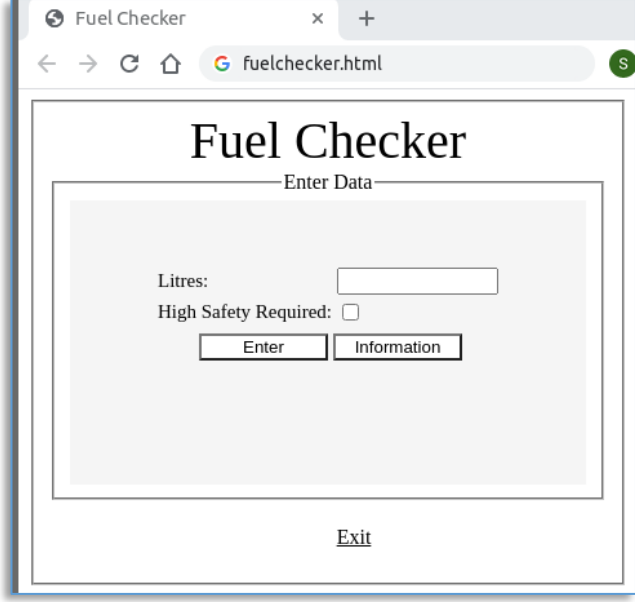
Example 3: Body Div



- The important details of this element for the tester are:
- The html element type:
  -
- The ID:
  - id=“body"

HTML Element Information

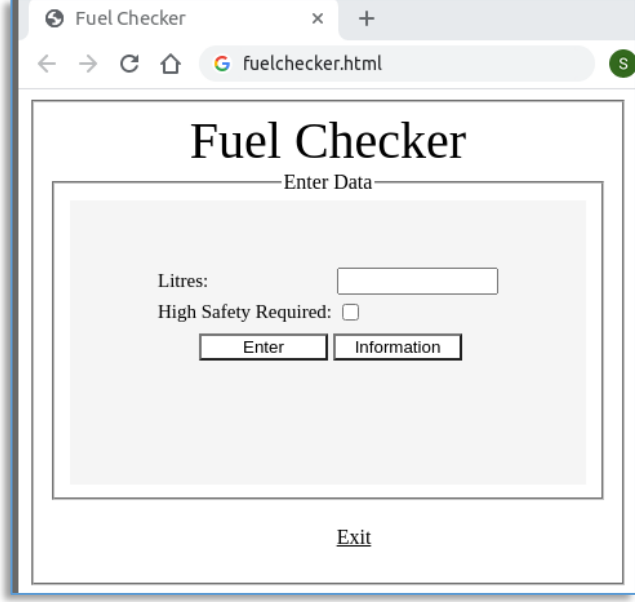
Page Title	HTML Element/Type	id
Fuel Checker	<input type="text">	litres



Continue

HTML Element Information

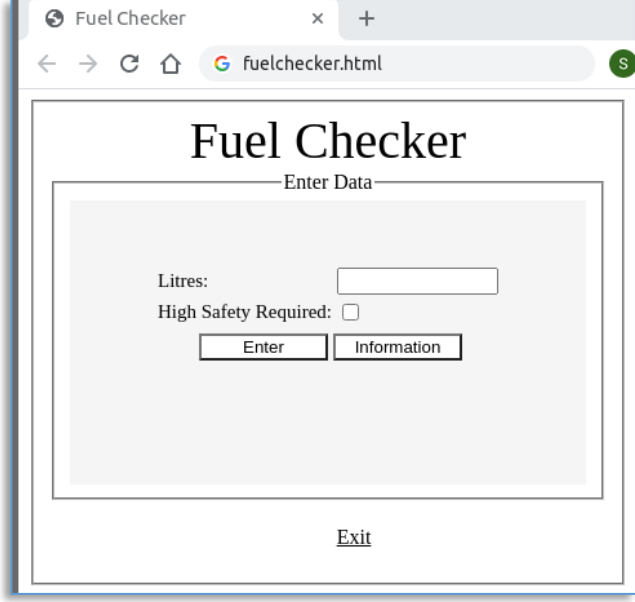
Page Title	HTML Element/Type	id
Fuel Checker	<input type="text">	litres
	<input type="checkbox">	highsafety



Continue

HTML Element Information

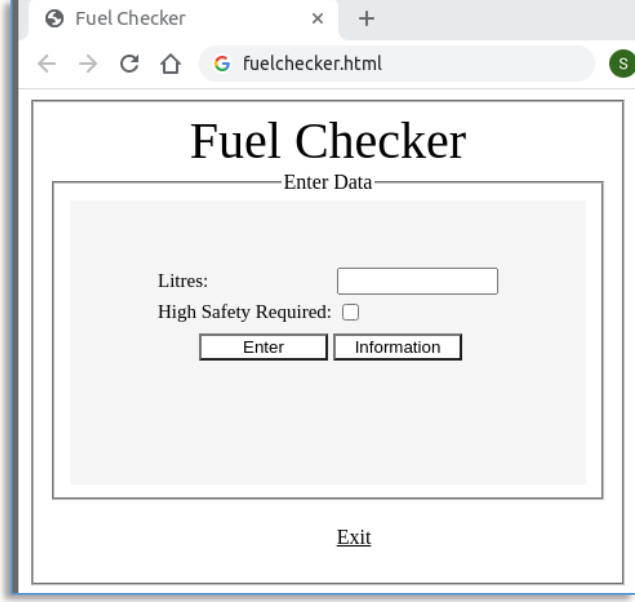
Page Title	HTML Element/Type	id
Fuel Checker	<input type="text">	litres
	<input type="checkbox">	highsafety
	<button type="button">	Enter



Continue

HTML Element Information

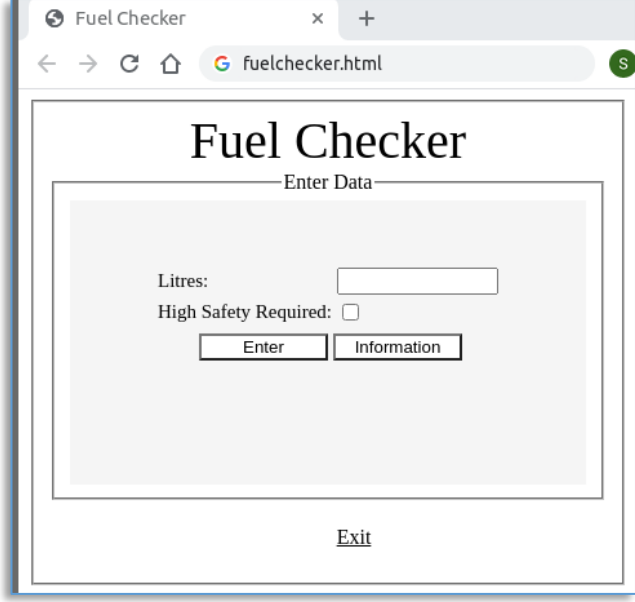
Page Title	HTML Element/Type	id
Fuel Checker	<input type="text">	litres
	<input type="checkbox">	highsafety
	<button type="button">	Enter
	<button type="button">	Info



Continue

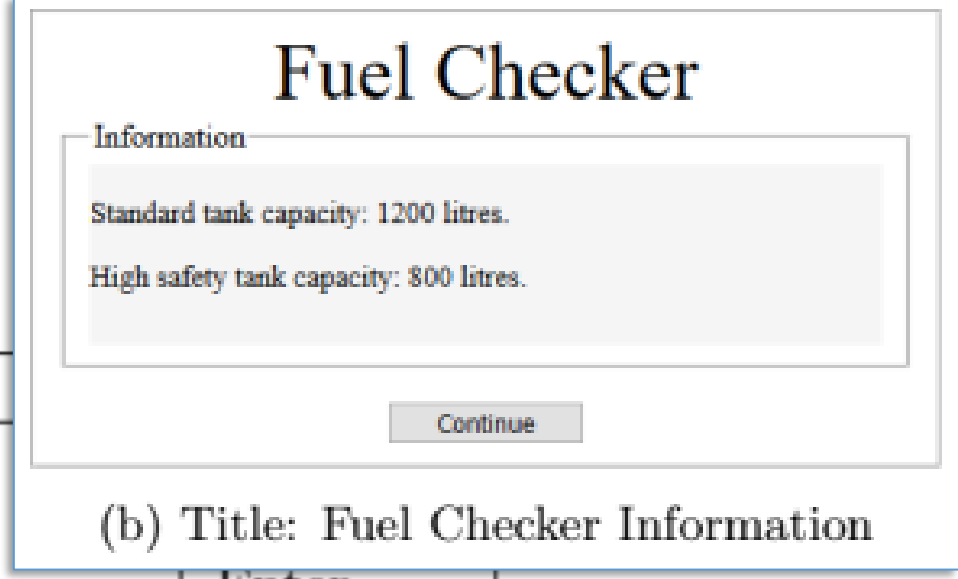
HTML Element Information

Page Title	HTML Element/Type	id
Fuel Checker	<input type="text">	litres
	<input type="checkbox">	highsafety
	<button type="button">	Enter
	<button type="button">	Info
	<a>	exitlink



Continue

HTML Element Information

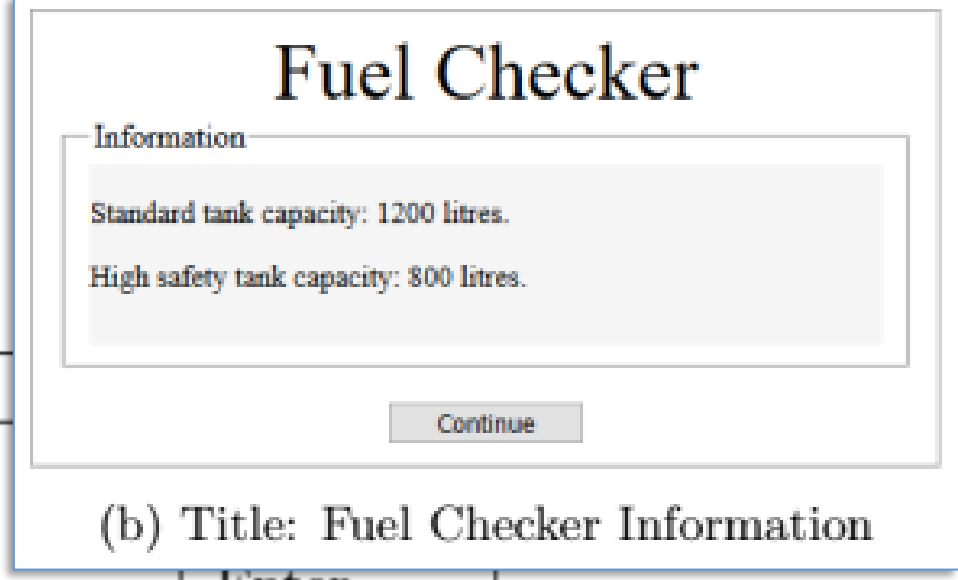


(b) Title: Fuel Checker Information

Page Title	HTML Element/Type	
Fuel Checker	<input type="text"> <input type="checkbox">> <button type="button">> <button type="button">> <a>	 Enter Info exitlink
Fuel Check	<button type="button">>	goback

Continue

HTML Element Information



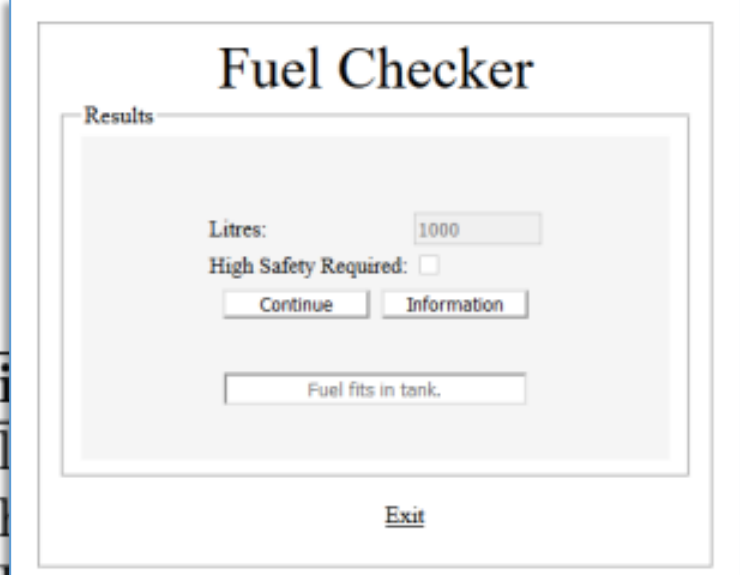
(b) Title: Fuel Checker Information

Page Title	HTML Element/Type	
Fuel Checker	<input type="text"> <input type="checkbox">> <button type="button">> <button type="button">> <a>	Enter Info exitlink
Fuel Check Information	<button type="button">> 	goback body

Continue

HTML Element Information

Page Title	HTML Element/Type	i
Fuel Checker	<input type="text"> <input type="checkbox">> <button type="button">> <button type="button">> <a>	l l l Info exitlink
Fuel Check Information	<button type="button">> 	goback body
Results	<input type="text" disabled>	litres



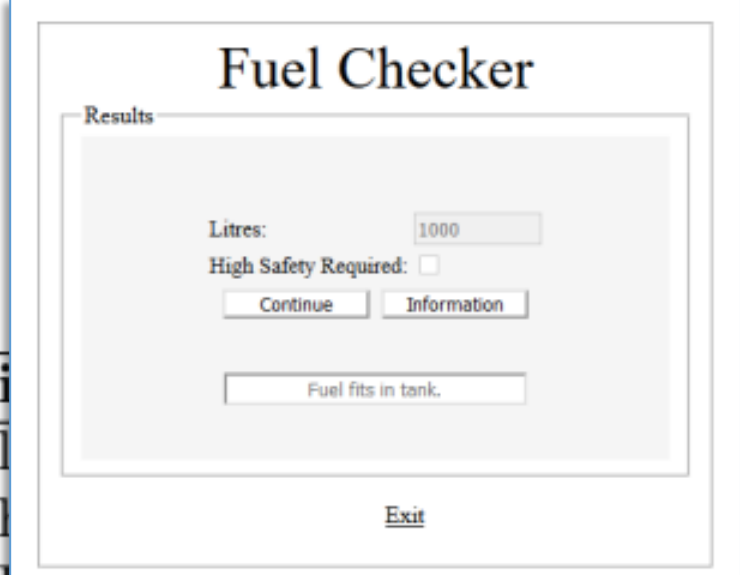
(f) Title: Results

Continue

HTML Element Information

Page Title	HTML Element/Type	i
Fuel Checker	<input type="text"> <input type="checkbox"> <button type="button"> <button type="button"> <a>	l l l Info exitlink
Fuel Check Information	<button type="button"> 	goback body
Results	<input type="text" disabled> <input type="checkbox" disabled>	litres highsafety

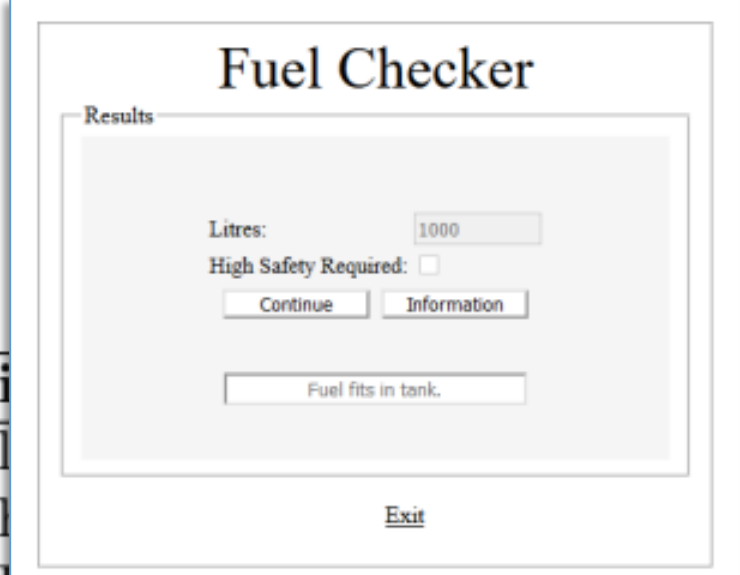
Continue



(f) Title: Results

HTML Element Information

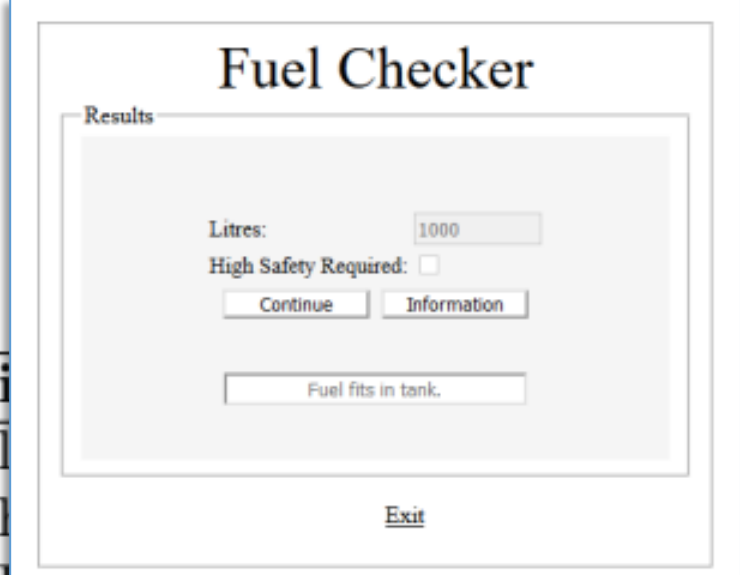
Page Title	HTML Element/Type	i
Fuel Checker	<input type="text"> <input type="checkbox"> <button type="button"> <button type="button"> <a>	l l l Info exitlink
Fuel Check Information	<button type="button"> 	goback body
Results	<input type="text" disabled> <input type="checkbox" disabled> <button type="button">	litres highsafety Continue



(f) Title: Results

HTML Element Information

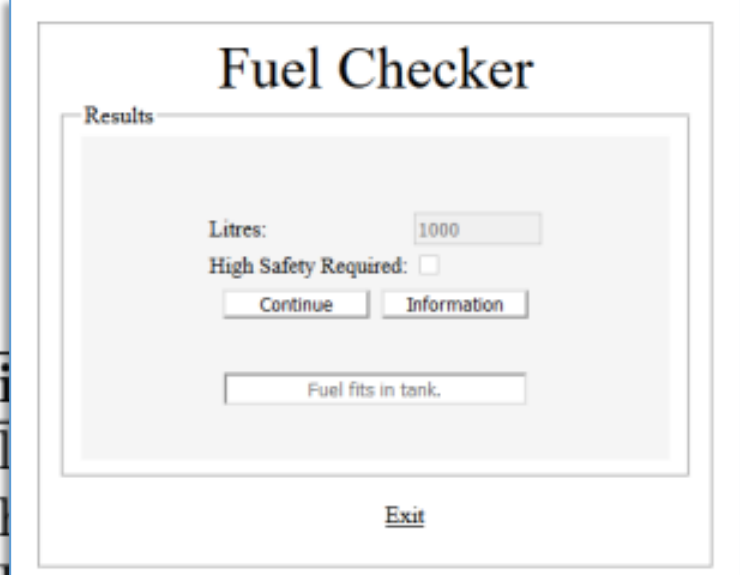
Page Title	HTML Element/Type	i
Fuel Checker	<input type="text"> <input type="checkbox"> <button type="button"> <button type="button"> <a>	l l l Info exitlink
Fuel Check Information	<button type="button"> 	goback body
Results	<input type="text" disabled> <input type="checkbox" disabled> <button type="button"> <input type="text">	litres highsafety Continue result



(f) Title: Results

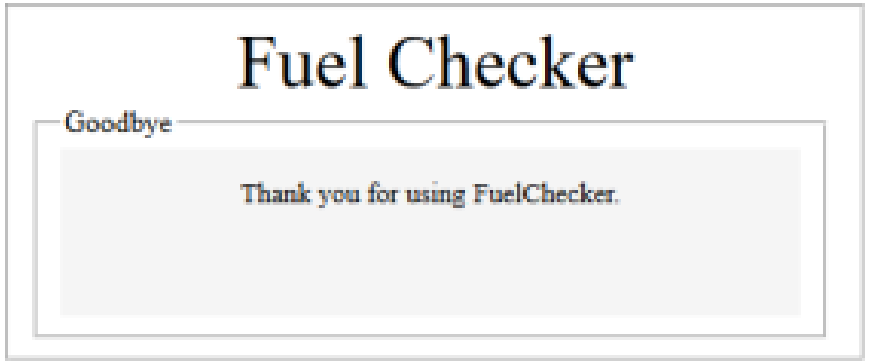
HTML Element Information

Page Title	HTML Element/Type	i
Fuel Checker	<input type="text"> <input type="checkbox"> <button type="button"> <button type="button"> <a>	l l l Info exitlink
Fuel Check Information	<button type="button"> 	goback body
Results	<input type="text" disabled> <input type="checkbox" disabled> <button type="button"> <input type="text"> <a>	litres highsafety Continue result exitlink



(f) Title: Results

HTML Element Information



(h) Title: Thank you

Page Title	HTML Element/Type	
Fuel Checker	<input type="text">	
	<input type="checkbox">	highsafety
	<button type="button">	Enter
	<button type="button">	Info
	<a>	exitlink
Fuel Check Information	<button type="button">	goback
		body
Results	<input type="text" disabled>	litres
	<input type="checkbox" disabled>	highsafety
	<button type="button">	Continue
	<input type="text">	result
	<a>	exitlink
Thank you		body

Note on Input Elements

- The input elements for litres and highsafety are disabled initially
- They are then dynamically enabled as required by the application, producing the screens shown in the trial runs

Data Representation: highSafety and litres

- The data representation used for the inputs and outputs is determined by examining the HTML elements and their appearance on the screen
- The input **highSafety**:
  - A checkbox element
  - This represents a boolean value
- The input **litres:**
  - A text input element (a string)
  - This represents an integer value

Data Representation: result

- The output **result:**
  - A non-editable text element (a string)
- The text can take one of three possible outputs:
  - "Fuel fits in tank."
  - "Fuel does not fit in tank."
  - "Invalid data entered."

Data Representation: body

- The output **body** of the information screen:
  - A non-editable text element (a string)
- The content of body is a complex HTML expression (as shown)
- We are not testing the text formatting is correct, we just need to check that this element contains the correct text. The other HTML tags can be ignored
- The correct text contains the two important phrases:
  - "Standard tank capacity: 1200 litres"
  - "High safety tank capacity: 800 litres"

Analysis Results

- The analysis is now complete – we have identified:
  - Each screen displayed by the application
  - Each web element on each screen required for testing
  - The data representation

Information from the Inspector

- Use the inspector (or view the HTML) when these are not specified in detail (often the case):
  - Web elements and their IDs
  - Data representation
- Risks:
  - The app may not be correctly designed or implemented
  - So the inspector may not show the correct information…
  - Much higher quality if these details are specified by the web page designer

Data Values

- A simple user story test case will use a single data value for each acceptance criterion
- In more advanced testing, the analysis can be extended to identify equivalence partitions, boundary values, and combinations for testing

Test Coverage Items

- Each acceptance criterion (AC) for each user story (US) is a test coverage item

TCI	Acceptance Criteria	Test Case
US1	S1A1	To be completed
US2	S1A2	
US3	S1A3	
US4	S1A4	
US5	S1A5	
US6	S1A6	
US7	S1A7	

- Even though US7 reports an error back to the user, it is not an error case in the same sense as in EP testing. Each user story/acceptance criteria is tested separately, so the issue of error hiding does not apply, and we do not need to identify error cases.

Test Cases/Selecting Data Values

- In a manner similar to equivalence partition testing, typical data values are selected for the test cases
- These may be selected in advance, as show a few slides previous
- Or selected during the development of the test cases
- Selecting standard values makes reviewing the completed test cases easier
- For application testing, it is **not usual** to perform the detailed analysis using value lines that is shown for unit testing (unless it is required for complicated data values)

Test Cases/Data Values

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit
- S1A4 Check a high volatility fuel load that does not fit
- S1A5 List tank capacities
- S1A6 Exit when done
- S1A7 Identify a user input data error

TCI	Input	Value
US1	litres	"1000"
US2	litres	"400"
US3	litres	"2000"
US4	litres	"1000"
US7	litres	"xxx"

Test Cases/Data Values–Invalid Number String

- There are many possible invalid strings that may be entered for an invalid integer value in US7
- Discuss later in the module
- The expected results (the correct outputs, and their data representation) have already been identified during analysis of the application

TCI	Input	Value
US1	litres	"1000"
US2	litres	"400"
US3	litres	"2000"
US4	litres	"1000"
US7	litres	"xxx"

Test Cases/Output Data Values

TCI	Output	Value
US1	result	"Fuel fits in tank."
US2	result	"Fuel fits in tank."
US3	result	"Fuel does not fit in tank."
US4	result	"Fuel does not fit in tank."
US5	body	contains "Standard tank capacity: 1200 litres" and "High safety tank capacity: 800 litres"
US6	body	"Thank you for using fuelchecker"
US7	result	"Invalid data values."

Developing the Test Cases

- The test data for each test case is specified as a sequence of user actions to be simulated by the automated test
- Each TCI is a separate test case
- Develop TC1 in detail
- Note: details such as opening the web page are not necessary here, as the test implementer may chose to reopen the page for each test or not (for efficiency purposes it is better not to). If leave open, the test implementation must either leave the app on a specific screen, or each test mush move to the correct screen first

Test Case: TC1

- **S1A1 Check a low volatility fuel load that fits in a tank**

- After starting the application, enter 1000 into litres.
- Make sure that highsafety is deselected.
- Then click on Enter, and make sure the application moves to the Results screen.
- Check that Result contains the text "Fuel fits in the tank."

ID	TCI Covered	Inputs	Exp. Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	 Moved to Results screen Result is "Fuel fits in tank."

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit
- S1A4 Check a high volatility fuel load that does not fit

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T4	US4	Enter "1000" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit
- S1A4 Check a high volatility fuel load that does not fit
- S1A5 List tank capacities

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T4	US4	Enter "1000" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T5	US5	Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit
- S1A4 Check a high volatility fuel load that does not fit
- S1A5 List tank capacities
- S1A6 Exit when done

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T4	US4	Enter "1000" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T5	US5	Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"
T6	US6	Click on exitlink	Moved to Thank you screen body contains "Thank you for using fuelchecker."

Completed Test Cases

- S1A1 Check a low volatility fuel load that fits
- S1A2 Check a high volatility fuel load that fits
- S1A3 Check a low volatility fuel load that does not fit
- S1A4 Check a high volatility fuel load that does not fit
- S1A5 List tank capacities
- S1A6 Exit when done
- S1A7 Identify a user input data error

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T4	US4	Enter "1000" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T5	US5	Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"
T6	US6	Click on exitlink	Moved to Thank you screen body contains "Thank you for using fuelchecker."
T7	US7	Enter "xxx" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Invalid data values."

Verifying the Test Cases

- Complete the TCI table
- Check: all the TCIs are covered by a Test Case

TCI	Acceptance Criteria	Test Case
US1	S1A1	T1
US2	S1A2	T2
US3	S1A3	T3
US4	S1A4	T4
US5	S1A5	T5
US6	S1A6	T6
US7	S1A7	T7

BREAK

Test Implementation

- TestNG is used to run the tests and collect the results
- In order to simulate user input into a web application, and to collect the output for verification, a web automation test library must be used
- A good and widely used example is Selenium (www.selenium.dev)
- This is used in the test implementation as a representative example to demonstrate the principles of test automation for web applications
- Note: in this case the tester has decided to always leave the app at the “Fuel Checker” screen at the end of each test using a method returnToMain()

Viewing the Implementation

- I suggest you bring up the test implementation on your screens for reference while I describe how the test code works
- \ch10\FuelCheckerWebStoryTest.java and on Moodle/Slides
- Note: Selenium has been updated since the book was published: it now automatically downloads the ChromeDriver executable

Implementation: Test T1

First, view how data values are used

```
// Tests go here

@Test(timeOut=60000)
public void test1() {
  String litres = "1000";
  boolean highsafety = false;
  String result = "Fuel fits in tank.";
  wait.until(ExpectedConditions.titleIs("Fuel Checker"));
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("litres")));
  driver.findElement(By.id("litres")).sendKeys(litres);
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("highsafety")));
  if (driver.findElement(
        By.id("highsafety")).isSelected()!=highsafety)
    driver.findElement(By.id("highsafety")).click();
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("Enter")));
  driver.findElement(By.id("Enter")).click();
  wait.until(ExpectedConditions.titleIs("Results"));
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("result")));
  assertEquals( driver.findElement(
        By.id("result")).getAttribute("value"),result );
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("Continue")));
  driver.findElement(By.id("Continue")).click();
  wait.until(ExpectedConditions.titleIs("Fuel Checker"));
}
```

Implementation: Test T1

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```


```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
```

Implementation: Test T1

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```

```
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLo
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```

Implementation: Test T1

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
// Tests go here

@Test(timeOut=60000)
public void test1() {
  String litres = "1000";
  boolean highsafety = false;
  String result = "Fuel fits in tank.";
  wait.until(ExpectedConditions.titleIs("Fuel Checker"));
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("litres")));
  driver.findElement(By.id("litres")).sendKeys(litres);
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("highsafety")));
  if (driver.findElement(
        By.id("highsafety")).isSelected()!=highsafety)
    driver.findElement(By.id("highsafety")).click();
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("Enter")));
  driver.findElement(By.id("Enter")).click();
```

```
    driver.findElement(By.id("highsafety")).click();
  wait.until(ExpectedConditions.visibilityOfElementLo
        By.id("Enter")));
  driver.findElement(By.id("Enter")).click();
  wait.until(ExpectedConditions.titleIs("Results"));
  wait.until(ExpectedConditions.visibilityOfElementLo
        By.id("result")));
```

Implementation: Test T1

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```

```
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementL
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),resul
```

Implementation: Test T1

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```

```
            By.id("result")));
75        assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result )
76        wait.until(ExpectedConditions.visibilityOfElementLoca
            By.id("Continue")));
77        driver.findElement(By.id("Continue")).click();
78        wait.until(ExpectedConditions.titleIs("Fuel Checker")
```

Implementation: Test T1

Next, examine the test step-by-step

```
58   // Tests go here
59
60   @Test(timeOut=60000)
61   public void test1() {
62     String litres = "1000";
63     boolean highsafety = false;
64     String result = "Fuel fits in tank.";
65     wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("litres")));
67     driver.findElement(By.id("litres")).sendKeys(litres);
68     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("highsafety")));
69     if (driver.findElement(
             By.id("highsafety")).isSelected()!=highsafety)
70       driver.findElement(By.id("highsafety")).click();
71     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("Enter")));
72     driver.findElement(By.id("Enter")).click();
73     wait.until(ExpectedConditions.titleIs("Results"));
74     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("result")));
75     assertEquals( driver.findElement(
             By.id("result")).getAttribute("value"),result );
76     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("Continue")));
77     driver.findElement(By.id("Continue")).click();
78     wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79   }
```

Implementation: Test T1

- Line 60
- Web-based tests require a timeout: in case the browser does not respond, or the test hangs indefinitely waiting for a specific response
- In this test, a timeout of 60 seconds is selected
- The value depends on connectivity and is contextual, and may require a few test runs

```
58   // Tests go here
59
60   @Test(timeOut=60000)
61   public void test1() {
62     String litres = "1000";
63     boolean highsafety = false;
64     String result = "Fuel fits in tank.";
65     wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("litres")));
67     driver.findElement(By.id("litres")).sendKeys(litres);
68     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("highsafety")));
69     if (driver.findElement(
             By.id("highsafety")).isSelected()!=highsafety)
70       driver.findElement(By.id("highsafety")).click();
71     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("Enter")));
72     driver.findElement(By.id("Enter")).click();
73     wait.until(ExpectedConditions.titleIs("Results"));
74     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("result")));
75     assertEquals( driver.findElement(
             By.id("result")).getAttribute("value"),result );
76     wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("Continue")));
77     driver.findElement(By.id("Continue")).click();
```

```
60    @Test(timeOut=60000)
```

Implementation: Test T1

- Line 65
- First the test makes sure the browser is on the correct screen
- Where web page titles are used, this can be best achieved by checking the title

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("highsafety")));
69      if (driver.findElement(
              By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("result")));
75      assertEquals( driver.findElement(
              By.id("result")).getAttribute("value"),result );
79    }
```

```
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
```

Implementation: Test T1

- Lines 66-67
- Next value for litres entered
- The browser may not have finished rendering the window, so the test must wait for this element to appear
- Then simulate user entry using sendkeys()

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("highsafety")));
69      if (driver.findElement(
              By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("result")));
75      assertEquals( driver.findElement(
              By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
```

```
66      wait.until(ExpectedConditions.visibilityOfElementLocated
              By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
```

Implementation: Test T1

- Line 67
- Each HTML element used in the test is found by calling the method By.id()

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77      driver.findElement(By.id("Continue")).click();
78      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
```

```
67      driver.findElement(By.id("litres")).sendKeys(litres);
```

Implementation: Test T1

- Lines 69-70
- The highsafety checkbox must be deselected
- First, current value of the checkbox is checked (line 69)
- If already selected, then click to deselect it (line 70)

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
```

```
69        if (driver.findElement(
                By.id("highsafety")).isSelected()!=highsafety)
70          driver.findElement(By.id("highsafety")).click();
```

Implementation: Test T1

- Lines 71-72
- Again, the browser may not have finished rendering the window, so the test must wait for the Enter button to appear before clicking it

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
```

```
71      wait.until(ExpectedConditions.visibilityOfElementLocated
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
```

Implementation: Test T1

- Line 73
- The test must check that the application has moved to the correct screen (title "Results")

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77      driver.findElement(By.id("Continue")).click();
78      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79    }
```

```
73        wait.until(ExpectedConditions.titleIs("Results"));
```

Implementation: Test T1

- Lines 64 and 75
- The test verifies the expected results: that the attribute value of the textfield **result** has the expected value, as held in the variable result ("Fuel fits in tank.")

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77      driver.findElement(By.id("Continue")).click();
```

```
75      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
```

Implementation: Test T1

- Line
- The
  expe
  attri
  textf
  expe
  the v
  ("Fu

```
58    // Tests go here
59
60    @Test(timeOut=60000)
61    public void test1() {
62      String litres = "1000";
63      boolean highsafety = false;
64      String result = "Fuel fits in tank.";
        ons.titleIs("Fuel Checker"));
        ons.visibilityOfElementLocated(

        'litres")).sendKeys(litres);
        ons.visibilityOfElementLocated(
        );

        .isSelected()!=highsafety)
        d("highsafety")).click();
        ons.visibilityOfElementLocated(

        'Enter")).click();
        ons.titleIs("Results"));
        ons.visibilityOfElementLocated(

        lement(
        Attribute("value"),result );
        ons.visibilityOfElementLocated(

        'Continue")).click();
```

Selenium Update

getAttribute() has been deprecated [in book]

use

getDomProperty("value") instead [in provided code]

```
75
        By.id("result")).getAttribute("value"),result );
```

Implementation: Test T1

- Lines 76-78
- The test now waits for the Continue button
- Clicks on it
- And verifies that the application has moved to the Fuel Checker screen

```
65      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("litres")));
67      driver.findElement(By.id("litres")).sendKeys(litres);
68      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("highsafety")));
69      if (driver.findElement(
              By.id("highsafety")).isSelected()!=highsafety)
70        driver.findElement(By.id("highsafety")).click();
71      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("Enter")));
72      driver.findElement(By.id("Enter")).click();
73      wait.until(ExpectedConditions.titleIs("Results"));
74      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("result")));
75      assertEquals( driver.findElement(
              By.id("result")).getAttribute("value"),result );
76      wait.until(ExpectedConditions.visibilityOfElementLocated(
              By.id("Continue")));
77      driver.findElement(By.id("Continue")).click();
78      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79    }
```

```
76      wait.until(ExpectedConditions.visibilityOfElementLocated
              By.id("Continue")));
77      driver.findElement(By.id("Continue")).click();
78      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
```

DEMO: Run the test

Test Results

```
Test started at: 2020-08-28T13:04:28.398050100
For URL: ch10\fuelchecker\fuelchecker.html

Starting ChromeDriver 84.0.4147.30 (48b3e868b4cc0aa7e8149519690b6f6949e110a8-
    refs/branch-heads/4147@{#310}) on port 35538
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for
    suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Aug 28, 2020 1:04:31 P.M. org.openqa.selenium.remote.ProtocolHandshake
    createSession
INFO: Detected dialect: W3C
PASSED: test1
===================================================
Command line suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===================================================
```

- The test T1 has passed

Test Results

```
Test started at: 2020-08-28T13:04:28.398050100
For URL: ch10\fuelchecker\fuelchecker.html

Starting ChromeDriver 84.0.4147.30 (48b3e868b4cc0aa7e8149519690b6f6949e110a8-
    refs/branch-heads/4147@{#310}) on port 35538
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for
    suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Aug 28, 2020 1:04:31 P.M. org.openqa.selenium.remote.ProtocolHandshake
    createSession
INFO: Detected dialect: W3C
```

- The time the test started and the URL are printed by the test code
- WebDriver startup/connection information confirms the web browser has started properly, and Selenium session to the browser has started
- These details are not important to the test result

Mapping:
Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	 Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Prepare the test

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	 Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	 Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres	
Deselect highsafety	
Click on Enter	Moved to Results screen
	Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres	
Deselect highsafety	
Click on Enter	Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres	
Deselect highsafety	
Click on Enter	Moved to Results screen
	Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Mapping: Test Cases to Test Code

Inputs	Exp. Results
Enter "1000" into litres Deselect highsafety Click on Enter	 Moved to Results screen Result is "Fuel fits in tank."

```
58  // Tests go here
59
60  @Test(timeOut=60000)
61  public void test1() {
62    String litres = "1000";
63    boolean highsafety = false;
64    String result = "Fuel fits in tank.";
65    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
66    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
67    driver.findElement(By.id("litres")).sendKeys(litres);
68    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
69    if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
70      driver.findElement(By.id("highsafety")).click();
71    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
72    driver.findElement(By.id("Enter")).click();
73    wait.until(ExpectedConditions.titleIs("Results"));
74    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
75    assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
76    wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
77    driver.findElement(By.id("Continue")).click();
78    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
79  }
```

Return to “Fuel Checker” page

Reminder: Test Cases

- Note that Test Cases TC1-4 and T7 have the same structure

ID	TCI Covered	Inputs	Expected Results
T1	US1	Enter "1000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T2	US2	Enter "400" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel fits in tank."
T3	US3	Enter "2000" into litres Deselect highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T4	US4	Enter "1000" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Fuel does not fit in tank."
T5	US5	Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"
T6	US6	Click on exitlink	Moved to Thank you screen body contains "Thank you for using fuelchecker."
T7	US7	Enter "xxx" into litres Select highsafety Click on Enter	Moved to Results screen Result is "Invalid data values."

Adding Tests T2-4 and T7

- Test 1 can be run on its own
- Adding further tests requires decisions about how the sequence of tests is to run, and how code duplication can be avoided
- Restarting the web application each time is very slow, so it is usual to run the tests in sequence
- This requires that each test leaves the application on a selected screen
- For this application, it is easiest to always return the application to the Fuel Checker screen at the end of each test
- Tests T1-4 and T7 have the same structure; code duplication avoided by using parameterised tests

Data Provider (Tests T1-4, T7)

```
@DataProvider(name="testset1") // Data for test cases T1-T4,T7
public Object[][] getdata() {
  return new Object[][] {
    { "T1", "1000", false, "Fuel fits in tank." },
    { "T2", "400", true, "Fuel fits in tank." },
    { "T3", "2000", false, "Fuel does not fit in tank." },
    { "T4", "1000", true, "Fuel does not fit in tank." },
    { "T7", "xxx", true, "Invalid data values." },
  };
}
```

Code

- The code structure is based on T1
- But with the input parameters
  - litres
  - highsafety
  - result
- from the data provider

```
85    @Test(timeOut=60000, dataProvider="testset1")
86    public void testEnterCheckView(String tid, String litres, boolean
            highsafety, String result) {
87      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
88      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("litres")));
89      driver.findElement(By.id("litres")).sendKeys(litres);
90      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("highsafety")));
91      if (driver.findElement(
            By.id("highsafety")).isSelected()!=highsafety)
92        driver.findElement( By.id("highsafety")).click();
93      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Enter")));
94      driver.findElement( By.id("Enter")).click();
95      wait.until(ExpectedConditions.titleIs("Results"));
96      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("result")));
97      assertEquals( driver.findElement(
            By.id("result")).getAttribute("value"),result );
98      wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.id("Continue")));
99      driver.findElement( By.id("Continue")).click();
100     wait.until(ExpectedConditions.titleIs("Fuel Checker"));
101   }
```

Test T5

Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"

```
@Test(timeOut=60000)
public void test_T5() {
  // Info -> "Standard tank capacity: 1200 litres" and "High safety
        tank capacity: 800 litres"
  wait.until(ExpectedConditions.titleIs("Fuel Checker"));
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("Info")));
  driver.findElement(By.id("Info")).click();
  wait.until(ExpectedConditions.titleIs("Fuel Checker
        Information"));
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("body")));
  assertTrue(
      driver.findElement(
            By.id("body")).getAttribute("innerHTML").contains(
            "Standard tank capacity: 1200 litres")
      &&
      driver.findElement(
            By.id("body")).getAttribute("innerHTML").contains(
            "High safety tank capacity: 800 litres")
  );
  wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("goback")));
  driver.findElement(By.id("goback")).click();
  wait.until(ExpectedConditions.titleIs("Fuel Checker"));
}
```

Test T5

Click on Info	Moved to Information screen body contains "Standard tank capacity: 1200 litres" body contains "High safety tank capacity: 800 litres"

```
103   @Test(timeOut=60000)
104   public void test_T5() {
105     // Info -> "Standard tank capacity: 1200 litres" and "High safety
              tank capacity: 800 litres"
                                                    ("Fuel Checker"));
                                                    ityOfElementLocated(

                                                    ck();
                                                    ("Fuel Checker

                                                    ityOfElementLocated(


                                                    ("innerHTML").contains(
                                                    00 litres")


                                                    ("innerHTML").contains(
                                                     800 litres")

                                                    ityOfElementLocated(

117     driver.findElement(By.id("goback")).click();
118     wait.until(ExpectedConditions.titleIs("Fuel Checker"));
119   }
```

Selenium Update

getAttribute() has been deprecated [in book]

use

getDomProperty("innerHTML") instead [in provided code]

Test T6

Click on exitlink	Moved to Thank you screen body contains "Thank you for using fuelchecker."

```
121  @Test(timeOut=60000)
122  public void test_T6() {
123    // exit -> "Thank you for using FuelChecker."
124    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
125    wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("exitlink")));
126    driver.findElement(By.id("exitlink")).click();
127    wait.until(ExpectedConditions.titleIs("Thank you"));
128    wait.until(ExpectedConditions.visibilityOfElementLocated(
             By.id("body")));
129    assertTrue(driver.findElement(
             By.id("body")).getAttribute("innerHTML").contains( "Thank
             you for using FuelChecker."));
130  }
```

What happens if a test fails?

- It exits immediately on an assertion (or timeout or web interaction failure)
- Leaving the application on whatever page it happened to be on
- So need to make sure all tests (pass & fail) return to the main page

Return to Main

```
58    @AfterMethod
59    public void returnToMain() {
60      // If test has not left app at the main window, try to return
            there for the next test
61      if ("Results".equals(driver.getTitle()))
62        driver.findElement(By.id("Continue")).click();
63      else if ("Fuel Checker Information".equals(driver.getTitle()))
64        driver.findElement(By.id("goback")).click();
65      else if ("Thank you".equals(driver.getTitle()))
66        driver.get( url ); // only way to return to main screen from
              here
67      wait.until(ExpectedConditions.titleIs("Fuel Checker"));
68    }
```

- Making sure that a test leaves the application at the main screen, even if the test fails, requires a method to be run after each test
- @AfterMethod returnToMain() is run immediately after each @Test method

How it Works

- Lines 59-67
- If a test fails, it is important to return the application to the Fuel Checker screen
- This allows subsequent tests to run correctly
- We have defined all our tests to start from the main screen
- To handle this, an @AfterMethod method returnToMain() is provided

```
58  @AfterMethod
59  public void returnToMain() {
60    // If test has not left app at the main window, try to return
          there for the next test
61    if ("Results".equals(driver.getTitle()))
62      driver.findElement(By.id("Continue")).click();
63    else if ("Fuel Checker Information".equals(driver.getTitle()))
64      driver.findElement(By.id("goback")).click();
65    else if ("Thank you".equals(driver.getTitle()))
66      driver.get( url ); // only way to return to main screen from
            here
67    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
68  }
```

How it Works

- Lines 65-66
- If the application is left at the Thank you screen after a failure, there is no link or button for the user to click to return to the main screen
- The @AfterMethod code reloads the main application url to handle this

```
58  @AfterMethod
59  public void returnToMain() {
60    // If test has not left app at the main window, try to return
          there for the next test
61    if ("Results".equals(driver.getTitle()))
62      driver.findElement(By.id("Continue")).click();
63    else if ("Fuel Checker Information".equals(driver.getTitle()))
64      driver.findElement(By.id("goback")).click();
65    else if ("Thank you".equals(driver.getTitle()))
66      driver.get( url ); // only way to return to main screen from
            here
67    wait.until(ExpectedConditions.titleIs("Fuel Checker"));
68  }
```

DEMO: Run the tests

Test Results

```
Test started at: 2020-09-24T19:15:14.618050100
For URL: ch10\fuelchecker\fuelchecker.html

Starting ChromeDriver 84.0.4147.30 (48b3e868b4cc0aa7e8149519690b6f6949e110a8-
    refs/branch-heads/4147@{#310}) on port 1388
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for
    suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1600971316.989][WARNING]: This version of ChromeDriver has not been tested with
     Chrome version 85.
Sep 24, 2020 7:15:18 P.M. org.openqa.selenium.remote.ProtocolHandshake
    createSession
INFO: Detected dialect: W3C

===============================================
    Command line test
    Tests run: 7, Failures: 0, Skips: 0
```

- All the tests pass

Test/Selenium Setup

```
@BeforeClass
public void setupDriver() throws Exception {
    System.out.println("Test started at: "+LocalDateTime.now());
    if (url==null)
        throw new Exception("Test URL not defined: use -Durl=<url>");
    System.out.println("For URL: "+url);
    System.out.println();
    // Create web driver (this code uses chrome)
    driver = new ChromeDriver();
    // Create wait
    wait = new WebDriverWait( driver, Duration.ofSeconds(5) );
    // Open web page
    driver.get( url );
}
```

Test/Selenium Setup

```
@BeforeClass
public void setupDriver() throws Exce
    System.out.println("Test started a
    if (url==null)
        throw new Exception("Test URL r                                    );
    System.out.println("For URL: "+url
    System.out.println();
    // Create web driver (this code us
    driver = new ChromeDriver();
    // Create wait
    wait = new WebDriverWait( driver, Duration.ofSeconds(5) );
    // Open web page
    driver.get( url );
}
```

NOTE

The latest version of Selenium automatically downloads the program Chromedriver.exe when new ChromeDriver() is called

Unlike in the book
(based on a previous version)

Test/Selenium Shutdown

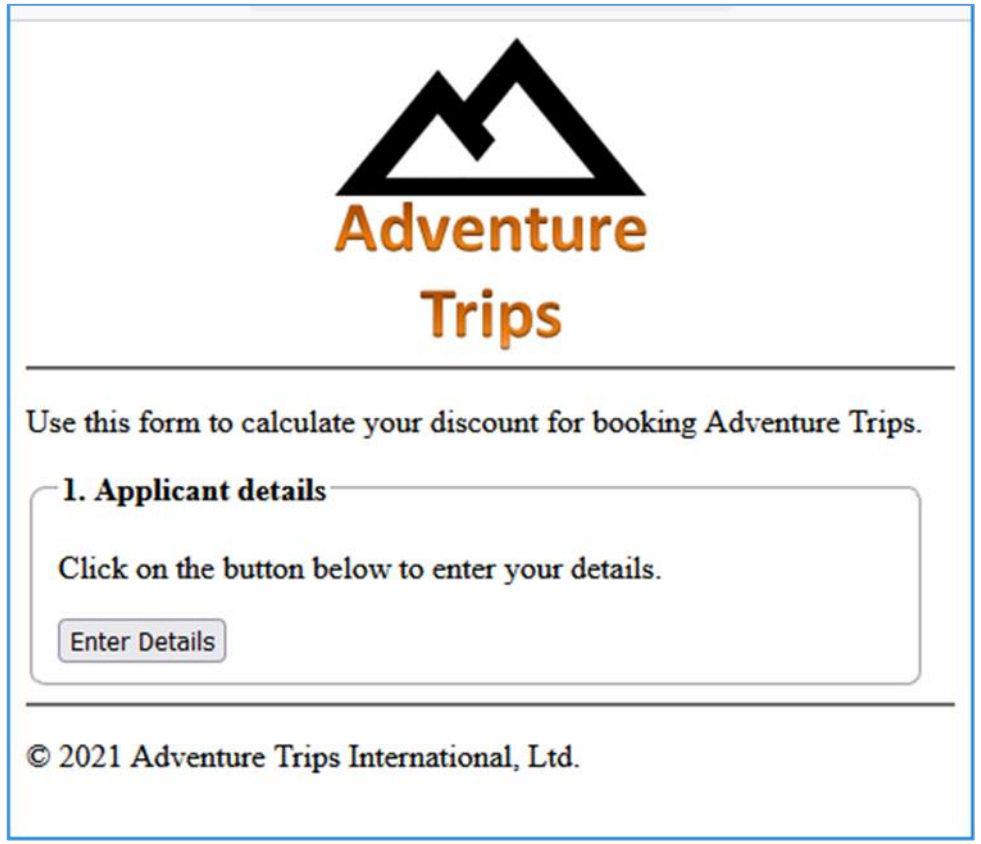
```
@AfterClass
public void shutdown() {
    driver.quit();
}
```

Latest Updates 2025

- Selenium suggest that the new W3C **WebDriver BiDi** Protocol will become the standard for browser automation
- "Selenium is updating its entire implementation from WebDriver Classic to WebDriver BiDi (while maintaining backwards compatibility as much as possible)"
- The traditional WebDriver model involves strict **request/response** commands which only allows for communication to happen in one direction at any given time
- WebDriver BiDi will provide for Asynchronous interactions. This will including streaming events from the user agent to the controlling software via WebSockets (for **event-based** testing)
- See https://www.selenium.dev/documentation/webdriver/bidi/

This Afternoon's Lab

- User Story Testing
- Web Application: AdventureTrips
- Test each user story once (using EP values)
- Quiz
- Submit:
  - PDF file
  - Test code
  - Output from running the tests
  - Log of **user actions** (every click, select, etc.) and test checks (**assert)** with a printf() statement – see lab instructions



Pulldown Menu (HTML <SELECT> TAG)

```
wait.until(ExpectedConditions.
   visibilityOfElementLocated(By.id("years")));

Select years=newSelect(
   driver.findElement(By.id("years")));

years.selectByVisibleText("Less than 5 years");
```

Click on Button by Text Displayed

```
wait.until(ExpectedConditions.
   visibilityOfElementLocated(
   By.xpath("//button[text()=\"Submit\"]")));

driver.findElement(
   By.xpath("//button[text()='Submit']")).click();
```

Special Notes- 1

- I have modified the @BeforeClass method from that shown in the book (Listing 10.6) to support the updated Selenium

Special Notes - 2

- I have also added test logging (automated marking) to keep a record of:
  - The simulated user actions
  - The checks made (assertions)
  - Example – test T1 is implemented for you, and you will see this as part of the output when you execute the command **@gradlew –rerun-tasks test** on the provided test template

```
T1+click(details)+sendkeys(39)+click(member)+select(lessthan
5years)+click(submit)+check(10%)+click(continue)
```

- Make sure to add this test logging to your own tests: lowercase,no spaces

Slowing Down the Tests

- To assist in debugging, you may want to slow down the tests
- You can do this by adding a Thread.sleep(millis) statement where needed – usually just before a user action such as click
- Example:

```
try {
      Thread.sleep(2000);
} catch (Exception e) {
      System.out.println(e.toString());
}
```

Special Notes - 3

- So you can see what is happening, I have configured **gradlew** to show you the downloads for the Selenium libraries

```
CS608> @gradlew test --rerun-tasks

> Task :test
T:\labs\cs608-labs-gradle-2025\lab8-app\build\classes\java\main
T:\labs\cs608-labs-gradle-2025\lab8-app\build\resources\main
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-java\4.27.0\2cc38c8dbef9d6e8774f05c
2d2e8088180858ed4\selenium-java-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-chrome-driver\4.27.0\bbd1b1defdf733
cfc6bcf7eb8f04ef6752dfaf12\selenium-chrome-driver-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-devtools-v129\4.27.0\daf042ab385011
82bc3c4a54f2398d8ecd8ca5e7\selenium-devtools-v129-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-devtools-v130\4.27.0\2ebfb80448483e
cb967bfb2bfc5e83d929c84c98\selenium-devtools-v130-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-devtools-v131\4.27.0\9071645ff0b06b
d25bb6aeafa31eeda8e0e524b8\selenium-devtools-v131-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-firefox-driver\4.27.0\872ea55cb20d0
73b4f5da1d401ee059322e59ddf\selenium-firefox-driver-4.27.0.jar
C:\Users\stephen\.gradle\caches\modules-2\files-2.1\org.seleniumhq.selenium\selenium-devtools-v85\4.27.0\786a93d6de154df
7c02afc77888a14b5b9068a51\selenium-devtools-v85-4.27.0.jar
```

Special Notes-4

- For the lab, gradlew dependencies are in build.gradle

```
dependencies {
   testImplementation(libs.slf4jApi)
   testImplementation(libs.testngTestng)
   implementation(libs.seleniumJava)
}
```

- And gradle\libs.versions.toml

```
v_slf4j = "1.7.36"
v_testng = "7.10.2"
v_jacoco = "0.8.12"
v_selenium = "4.27.0"
```

- Ignore requests from gradlew to upgrade Selenium