CM3110 – Mobile App Development

Coursework: *Will the Sun Shine Again?* App Testing Document

Code/Method Tested	Test Data	Expected Output	Actual Output
WelcomeActivity onCreate() Method The purpose of this test is to check that the onCreate method runs correctly when performing tasks that only need to happen once during the activity's life. This includes assigning values to variables, calling any helper methods, and logging any messages once commands have been successfully executed. From a broader perspective, this is a test to make sure the app launches correctly.	n/a	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_welcome.xml. A debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The updateLocations() helper method is called. A debug log	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_welcome.xml. A debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The fetchLocations() helper method is called. A debug log

		message is made saying “Initialised the shared preferences okay”. The btn variables are assigned Button components from the interface. These all have an onClickListener set. A debug log message is made saying “initialised the buttons and their click listeners okay”. The spinLocations variable is assigned the spinnerLocations View component from the interface. The adapter variable is assigned a new ArrayAdapter object, using the ArrayList of chosen locations as one of the parameters. The spinLocations variable has its adapter set to adapter. The spinner has an onItemSelectedListener attached. A debug log message is made saying “Set up spinners alright”.	message is made saying “Initialised the shared preferences okay”. The btn variables are assigned Button components from the interface. These all have an onClickListener set. A debug log message is made saying “initialised the buttons and their click listeners okay”. The spinLocations variable is assigned the spinnerLocations View component from the interface. The adapter variable is assigned a new ArrayAdapter object, using the ArrayList of chosen locations as one of the parameters. The spinLocations variable has its adapter set to adapter. The spinner has an onItemSelectedListener attached. A debug log message is made saying “Set up spinners alright”.
WelcomeActivity onStart() Method The purpose of this test is to check that the onStart method runs correctly when making the activity visible to the user and preparing it for interactivity.	n/a	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.

WelcomeActivity onPause() Method The purpose of this test is to check that the onPause method runs correctly when indicating that this activity is not in the foreground but should not be destroyed as it is expected to be resumed soon.	n/a	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.
WelcomeActivity onStop() Method The purpose of this test is to check that the onStop method runs correctly when a new activity has covered the screen, or when this activity has been terminated.	n/a	The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.	The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.
WelcomeActivity onDestroy() Method The purpose of this test is to check that the onDestroy method runs correctly when the user has completely finished using the activity.	n/a	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.
WelcomeActivity onResume() Method The purpose of this test is to check that the onResume method runs correctly when the user returns to this activity from another or from outside of the app.	n/a	The updateLocations helper method is called, the super class onDestroy() method is called and a debug log message is made saying “In the onResume event handler”.	The updateLocations helper method is called, the super class onDestroy() method is called and a debug log message is made saying “In the onResume event handler”.

WelcomeActivity onClick() Method The purpose of this test is to check that the onClick method does not run unless a button is pressed.	No button on the interface is pressed.	Nothing happens as no methods are called after the interface is displayed.	Nothing happens as no methods are called after the interface is displayed.
WelcomeActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The button to move to the Preferences screen is pressed.	The onClick method/event handler is called, sending the buttonPreferences view as a parameter. A new context object is created and assigned the value of the WelcomeActivity context. A debug log message is made saying "In the onClick method". The if statement is entered. The condition should match the initial condition (i.e. the id of the view is buttonPreferences) so the body of the if condition is executed. A Class object is created to store the PreferencesActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the preferences screen.	The onClick method/event handler is called, sending the buttonPreferences view as a parameter. A new context object is created and assigned the value of the WelcomeActivity context. A debug log message is made saying "In the onClick method". The if statement is entered. The condition should match the initial condition (i.e. the id of the view is buttonPreferences) so the body of the if condition is executed. A Class object is created to store the PreferencesActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the preferences screen.

WelcomeActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The button to move to the Search screen is pressed.	The onClick method/event handler is called, sending the buttonSearch view as a parameter. A new context object is created and assigned the value of the WelcomeActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. The condition doesn’t match the initial condition, so the body of the if condition is skipped. The condition should match the else if condition (i.e. the id of the view is buttonSearch) so the body of the else if condition is executed. A Class object is created to store the SearchActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the search screen.	The onClick method/event handler is called, sending the buttonSearch view as a parameter. A new context object is created and assigned the value of the WelcomeActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. The condition doesn’t match the initial condition, so the body of the if condition is skipped. The condition should match the else if condition (i.e. the id of the view is buttonSearch) so the body of the else if condition is executed. A Class object is created to store the SearchActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the search screen.
WelcomeActivity onClick() Method	The button to download new forecasts is pressed.	The onClick method/event handler is called, sending the buttonDownload view as a parameter. A new context object is created and assigned the	The onClick method/event handler is called, sending the buttonDownload view as a parameter. A new context object is created and assigned the

The purpose of this test is to check that the onClick method processes the clicking of each button correctly.		value of the WelcomeActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. The condition doesn’t match the initial condition, so the body of the if condition is skipped. The condition of the else if condition isn’t met, so the body of the else if condition is skipped. The condition should match the else condition (i.e. the id of the view is buttonDownload) so the body of the else condition is executed. The deleteForecastsDB method is called and a debug log message is made saying “Old forecasts deleted”. A new DownloaderTask object is created and then a new String is created. The String is created by concatenating the JSON_WEB_ADR_PART1, currentLocation, and JSON_WEB_ADR_PART2 Strings. If there are any spaces in the	value of the WelcomeActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. The condition doesn’t match the initial condition, so the body of the if condition is skipped. The condition of the else if condition isn’t met, so the body of the else if condition is skipped. The condition should match the else condition (i.e. the id of the view is buttonDownload) so the body of the else condition is executed. The deleteForecastsDB method is called and a debug log message is made saying “Old forecasts deleted”. A new DownloaderTask object is created and then a new String is created. The String is created by concatenating the JSON_WEB_ADR_PART1, currentLocation, and JSON_WEB_ADR_PART2 Strings. If there are any spaces in the

		currentLocation String, these are replaced with +s before the concatenation. The complete URL is then used in a debug log message. The try/catch statement is entered. A URL object is created used the concatenated String. A debug message is made after the URL is made. The DownloaderTask object is then sent the URL object for its execute method. If the URL is incorrect in any way and cannot be used, the catch statement is executed and a debug log message printed to say so.	currentLocation String, these are replaced with +s before the concatenation. The complete URL is then used in a debug log message. The try/catch statement is entered. A URL object is created used the concatenated String. A debug message is made after the URL is made. The DownloaderTask object is then sent the URL object for its execute method. If the URL is incorrect in any way and cannot be used, the catch statement is executed and a debug log message printed to say so.
WelcomeActivity onItemSelected() Method The purpose of this test is to check that the onItemSelected method correctly processes the selection of any item in the spinner.	An item is selected from the spinner.	The item at the selected position in the spinner is converted to a String and stored in the currentLocation variable. The getForecastsForLocation() method is then called. A Toast telling the user which location they have just selected appears.	The item at the selected position in the spinner is converted to a String and stored in the currentLocation variable. The getForecastsForLocation() method is then called. A Toast telling the user which location they have just selected appears.
WelcomeActivity onNothingSelected() Method	No item has been newly selected (it is not possible to	Nothing happens as this method has no body.	Nothing happens as this method has no body.

The purpose of this test is to check that the onNothingSelected method correctly processes the lack of selection of any item in the spinner.	have no item selected at all).		
WelcomeActivity updateLocations() Method The purpose of this test is to check that the user's chosen locations are retrieved from the shared preferences.	n/a	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. A counter variable is initialised. A while loop is entered. Which goes through all the LOCATION_X Strings in the shared preferences until one returns a null. If a String does not return a null, it is logged with a debug message before it is added to the chosenLocations ArrayList. This ArrayList's size is then printed upon completion of the while loop. A new ArrayAdapter is created using the updated ArrayList. The same adapter is then set for the spinner.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. A counter variable is initialised. A while loop is entered. Which goes through all the LOCATION_X Strings in the shared preferences until one returns a null. If a String does not return a null, it is logged with a debug message before it is added to the chosenLocations ArrayList. This ArrayList's size is then printed upon completion of the while loop. A new ArrayAdapter is created using the updated ArrayList. The same adapter is then set for the spinner.
WelcomeActivity saveLocations() Method	n/a	The sharedPrefs variable is initialised with the getSharedPreferences() method and sharedPrefFile variable as the parameter. The mEditor is also	The sharedPrefs variable is initialised with the getSharedPreferences() method and sharedPrefFile variable as the parameter. The mEditor is also

The purpose of this test is to check that the user's chosen locations are saved in the shared preferences.		initialised as the return from the edit() method of the sharedPrefs object. A for loop is entered for the length of the chosenLocations ArrayList. For each item in the ArrayList, it is added to the shared preferences in a String called LOCATION_X (where X is the counter/ArrayList position + 1). The mEditor then applies these changes.	initialised as the return from the edit() method of the sharedPrefs object. A for loop is entered for the length of the chosenLocations ArrayList. For each item in the ArrayList, it is added to the shared preferences in a String called LOCATION_X (where X is the counter/ArrayList position + 1). The mEditor then applies these changes.
WelcomeActivity updateMainSummary() Method The purpose of this test is to check that the main summary section of the interface is correctly updated to show the current forecast for the current location.	A Forecast object.	All the relevant View components from the interface are stored in appropriately named variables using the findViewById method. The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated and any necessary conversion are done through the Converter helper class. The setWeatherIcon helper method is called to change the current weather icon/image.	All the relevant View components from the interface are stored in appropriately named variables using the findViewById method. The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated and any necessary conversion are done through the Converter helper class. The setWeatherIcon helper method is called to change the current weather icon/image.

WelcomeActivity updateHourlyForecasts () Method The purpose of this test is to check that the hourly forecasts section of the interface is correctly updated to show the current and next 5 (3 hour) increments for the current location.	A List of Forecast objects.	All the needed Forecast objects from the List are stored in their own individual Forecast objects/variable for ease of access later. All the relevant View components from the interface are stored in appropriately named variables using the findViewById method. The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated. The setWeatherIcon helper method is called to change the weather icons/images.	All the Forecast objects from the List are stored in their own individual Forecast objects/variable for ease of access later. All the relevant View components from the interface are stored in appropriately named variables using the findViewById method. The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated. The setWeatherIcon helper method is called to change the weather icons/images.
WelcomeActivity updateDailyForecasts() Method The purpose of this test is to check that the hourly forecasts section of the interface is correctly updated to show the current and next 5 (3 hour) increments for the current location.	A List of Forecast objects.	All the needed Forecast objects from the List are stored in their own individual Forecast objects/variable for ease of access later. All the relevant View components from the interface are stored in appropriately named variables using the findViewById method.	All the needed Forecast objects from the List are stored in their own individual Forecast objects/variable for ease of access later. All the relevant View components from the interface are stored in appropriately named variables using the findViewById method.

		The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated.	The values for the TextViews are set using the setText() method. Each value is accessed through the Forecast object's getter methods. Any addition Strings are concatenated.
WelcomeActivity setWeatherIcon() Method The purpose of this test is to check that the specified ImageView has its source changed to accurately reflect the weather given for the location/forecast time.	A weather id from a Forecast object and an ImageView.	The weather id is converted to a smaller code by dividing it by 100. An if statement is entered. If the original id value is equal to 800, the ImageView has its source changed to display a sun icon. If not, the else statement's body is executed. Within this is a switch statement based on the converted id. Depending on the value of the converted id, an image resource is set that accurately corresponds to the reference material from the API codes.	The weather id is converted to a smaller code by dividing it by 100. An if statement is entered. If the original id value is equal to 800, the ImageView has its source changed to display a sun icon. If not, the else statement's body is executed. Within this is a switch statement based on the converted id. Depending on the value of the converted id, an image resource is set that accurately corresponds to the reference material from the API codes.
WelcomeActivity insertForecastsDB() Method The purpose of this test is to check that the method correctly calls the right task to insert the newly downloaded forecasts into the database.	An ArrayList of Forecast objects.	A new InsertForecastsTask object is created. The ArrayList of Forecasts objects is converted into an array for use with the DAO. The task is told to execute using the newly created array.	A new InsertForecastsTask object is created. The ArrayList of Forecasts objects is converted into an array for use with the DAO. The task is told to execute using the newly created array.

WelcomeActivity deleteForecastsDB() Method The purpose of this test is to check that the method correctly calls the right task to delete all existing forecasts from the database.	n/a	A new DeleteForecastsTask object is created and then told to execute.	A new DeleteForecastsTask object is created and then told to execute.
WelcomeActivity getForecastsForCurrentLocation() Method The purpose of this test is to check that the method correctly calls the right task to retrieve the required forecasts from the database.	n/a	A new UpdateGUIForLocationTask object is created and then told to execute.	A new UpdateGUIForLocationTask object is created and then told to execute.
WelcomeActivity - InsertForecastsTask doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to add forecasts to the database.	An array of Forecast objects.	The forecastDAO object is used to call its insertForecasts() method using the array of Forecast objects supplied in the parameters.	The forecastDAO object is used to call its insertForecasts() method using the array of Forecast objects supplied in the parameters.
WelcomeActivity - InsertForecastsTask onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A Void object.	The super class's onPostExecute method is called using the Void object as the parameter.	The super class's onPostExecute method is called using the Void object as the parameter.
WelcomeActivity - DeleteForecastsTask doInBackground() Method	n/a	The forecastDAO object is used to call its deleteAllForecastsFor()	The forecastDAO object is used to call its deleteAllForecastsFor()

The purpose of this test is to check that the method interacts with the DAO to delete all forecasts from the database for a specific location		method using the global currentLocation variable as its parameter.	method using the global currentLocation variable as its parameter.
WelcomeActivity - DeleteForecastsTask onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A Void object.	The super class's onPostExecute method is called using the Void object as the parameter.	The super class's onPostExecute method is called using the Void object as the parameter.
WelcomeActivity - UpdateGUIForLocationTask doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to get all forecasts from the database for a specific location	n/a	The forecastDAO object is used to call its getAllForecastsForLocation() method using the global currentLocation variable as its parameter. This List of Forecast objects is returned.	The forecastDAO object is used to call its getAllForecastsForLocation() method using the global currentLocation variable as its parameter. This List of Forecast objects is returned.
WelcomeActivity - UpdateGUIForLocationTask onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A List of Forecast objects.	The super class's onPostExecute method is called using the List as the parameter. An if else statement is entered. If the if condition is met, i.e. the size of List passed in is greater than 0, the body of the if statement is executed. In this body, the updateMainSummary() helper	The super class's onPostExecute method is called using the List as the parameter. An if else statement is entered. If the if condition is met, i.e. the size of List passed in is greater than 0, the body of the if statement is executed. In this body, the updateMainSummary() helper

		method is called and the first object in the List is sent through as the parameter. Then the updateHourlyForecasts() helper method is called, followed by the updateDailyForecasts() helper method. Both use the whole List as their parameter. If the if condition was not met, the body of the else statement is executed instead. A message is stored in a String object, and a Toast object created using the app's base context and the message String. This Toast is then shown to the user to tell them they haven't got any forecasts stored/downloaded for the selected location.	method is called and the first object in the List is sent through as the parameter. Then the updateHourlyForecasts() helper method is called, followed by the updateDailyForecasts() helper method. Both use the whole List as their parameter. If the if condition was not met, the body of the else statement is executed instead. A message is stored in a String object, and a Toast object created using the app's base context and the message String. This Toast is then shown to the user to tell them they haven't got any forecasts stored/downloaded for the selected location.
WelcomeActivity - DownloaderTask onPostExecute() Method The purpose of this test is to check that the method correctly precedes the job of the task.	An ArrayList of ForecastLocation objects	The super class's onPostExecute() method is called using the ArrayList as the parameter. A message is also made in the debug log. An ArrayList of Forecast objects is created. A for loop the length of the amount of ForecastLocation objects is	The super class's onPostExecute() method is called using the ArrayList as the parameter. A message is also made in the debug log. An ArrayList of Forecast objects is created. A for loop the length of the amount of ForecastLocation objects is

		entered. For each object, a new Forecast object is made and its setters are used, in combination with the getters of the ForecastLocation objects, to insert data into the object. Some of these are converted using the Converter helper class. The Forecast object then has its toString printed in the debug log. The Forecast object is then added to the new ArrayList. The insertForecastsDB() helper method is called using the Forecast object ArrayList as its parameter, then the getForecastsForCurrentLocation() helper method is called.	entered. For each object, a new Forecast object is made and its setters are used, in combination with the getters of the ForecastLocation objects, to insert data into the object. Some of these are converted using the Converter helper class. The Forecast object then has its toString printed in the debug log. The Forecast object is then added to the new ArrayList. The insertForecastsDB() helper method is called using the Forecast object ArrayList as its parameter, then the getForecastsForCurrentLocation() helper method is called.
WelcomeActivity - DownloaderTask doInBackground() Method The purpose of this test is to check that the method correctly downloads the JSON i.e. the main point of the task.	A URL object.	The URL is stored in a singular object and a String to store the JSON is initialised as null. An ArrayList of ForecastLocation objects is created. A try/catch statement is entered. The JSON downloaded using the getResponseFromHttpUrl method from the HttpDownloader class is stored in the downloaded String.	The URL is stored in a singular object and a String to store the JSON is initialised as null. An ArrayList of ForecastLocation objects is created. A try/catch statement is entered. The JSON downloaded using the getResponseFromHttpUrl method from the HttpDownloader class is stored in the downloaded String.

		If an IOException occurs, this is caught in the caught statement. If there was JSON successfully downloaded, it is passed to the parser helper method. If there wasn't, it isn't. The ArrayList of ForecastLocations is returned.	If an IOException occurs, this is caught in the caught statement. If there was JSON successfully downloaded, it is passed to the parser helper method. If there wasn't, it isn't. The ArrayList of ForecastLocations is returned.
WelcomeActivity - DownloaderTask parseJSON() Method The purpose of this test is to check that the method correctly parses the JSON downloaded from the doInBackground method.	A String of JSON.	An ArrayList of ForecastLocation objects is created and a try/catch statement is entered. The whole JSON String is stored in as JSONObject. The JSONObject called city is extracted and stored in its own object. The JSONArray called list is extract and stored in its own object. An ArrayList of JSONObjects is created. A for loop is entered for then length of the JSONArray. Each object within the JSONArray is stored as an individual JSONObject in the ArrayList. The city's name and its country are extracted from the city object and stored in variables.	An ArrayList of ForecastLocation objects is created and a try/catch statement is entered. The whole JSON String is stored in as JSONObject. The JSONObject called city is extracted and stored in its own object. The JSONArray called list is extract and stored in its own object. An ArrayList of JSONObjects is created. A for loop is entered for then length of the JSONArray. Each object within the JSONArray is stored as an individual JSONObject in the ArrayList. The city's name and its country are extracted from the city object and stored in variables.

		Another for loop is entered, this time for the length of the JSONObject ArrayList. For each object, the remaining forecast details are extracted and stored in variables. A new ForecastLocation object is created using all of these variables as parameters in the constructor. This new object is added to the ArrayList. If a JSONException occurs, the catch statement is entered. The ArrayList of ForecastLocation objects is returned.	Another for loop is entered, this time for the length of the JSONObject ArrayList. For each object, the remaining forecast details are extracted and stored in variables. A new ForecastLocation object is created using all of these variables as parameters in the constructor. This new object is added to the ArrayList. If a JSONException occurs, the catch statement is entered. The ArrayList of ForecastLocation objects is returned.
PreferencesActivity onCreate() Method The purpose of this test is to check that the onCreate method runs correctly when performing tasks that only need to happen once during the activity's life. This includes assigning values to variables, calling any helper methods, and logging any messages once commands have been successfully executed.	n/a	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_preferences.xml. A debug log message is made saying "In the onCreate event handler". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_preferences.xml. A debug log message is made saying "In the onCreate event handler". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from

From a broader perspective, this is a test to make sure the activity loads correctly.		the edit() method of the sharedPrefs field. The spinLocations variable is assigned the spinnerLocations View component from the interface. The adapter variable is assigned a new ArrayAdapter object, using the String array resource of available locations as one of the parameters. The spinLocations variable has its adapter set to adapter. The spinner has an onItemSelectedListener attached. All the TextViews are stored in their appropriate variables. All the buttons are stored in their appropriate variables. These all have an onClickListener set. The prepTextViews() and preButtons() helper methods are called, followed by the fetchLocations() helper method. Finally, the other helper methods, locationsButtonsVisible() locationsButtonsInvisible() are called.	the edit() method of the sharedPrefs field. The spinLocations variable is assigned the spinnerLocations View component from the interface. The adapter variable is assigned a new ArrayAdapter object, using the String array resource of available locations as one of the parameters. The spinLocations variable has its adapter set to adapter. The spinner has an onItemSelectedListener attached. All the TextViews are stored in their appropriate variables. All the buttons are stored in their appropriate variables. These all have an onClickListener set. The prepTextViews() and preButtons() helper methods are called, followed by the fetchLocations() helper method. Finally, the other helper methods, locationsButtonsVisible() locationsButtonsInvisible() are called.

PreferencesActivity onStart() Method The purpose of this test is to check that the onStart method runs correctly when making the activity visible to the user and preparing it for interactivity.	n/a	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.
PreferencesActivity onPause() Method The purpose of this test is to check that the onPause method runs correctly when indicating that this activity is not in the foreground but should not be destroyed as it is expected to be resumed soon.	n/a	The saveLocations() helper method is called. The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.	The saveLocations() helper method is called. The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.
PreferencesActivity onStop() Method The purpose of this test is to check that the onStop method runs correctly when a new activity has covered the screen, or when this activity has been terminated.	n/a	The saveLocations() helper method is called. The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.	The saveLocations() helper method is called. The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.
PreferencesActivity onDestroy() Method The purpose of this test is to check that the onDestroy method runs correctly	n/a	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.

when the user has completely finished using the activity.			
PreferencesActivity onResume() Method The purpose of this test is to check that the onResume method runs correctly when the user returns to this activity from another or from outside of the app.	n/a	The super class onResume() method is called and a debug log message is made saying “In the onResume event handler”.	The super class onResume() method is called and a debug log message is made saying “In the onResume event handler”.
PreferencesActivity onClick() Method The purpose of this test is to check that the onClick method does not run unless a button is pressed.	No button on the interface is pressed.	Nothing happens as no methods are called after the interface is displayed.	Nothing happens as no methods are called after the interface is displayed.
PreferencesActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The button to add the currently selected location to the user’s list.	The onClick method/event handler is called, sending the buttonPreferences view as a parameter. A Toast and String message are declared, ready for use. The if statement is entered. The condition should match the initial condition (i.e. the id of the view is buttonAdd) so the body of the if condition is executed. If the user does not have less than 10 locations, the message is set and the Toast is displayed to tell them they	The onClick method/event handler is called, sending the buttonPreferences view as a parameter. A Toast and String message are declared, ready for use. The if statement is entered. The condition should match the initial condition (i.e. the id of the view is buttonAdd) so the body of the if condition is executed. If the user does not have less than 10 locations, the message is set and the Toast is displayed to tell them they

		can't have any more locations until they delete one. If they do have less than 10 locations, the list is checked to see if the selected location is already included. If it is, the user is told they cannot add a duplicate. If it isn't, the location is added to the next available spot in the ArrayList (replacing a null value). The number of locations counter is increased. The locationsButtonsVisible() and locationsButtonsInvisible() helper methods are called. The message is set and the Toast displayed to tell the user they've added the location to their list.	can't have any more locations until they delete one. If they do have less than 10 locations, the list is checked to see if the selected location is already included. If it is, the user is told they cannot add a duplicate. If it isn't, the location is added to the next available spot in the ArrayList (replacing a null value). The number of locations counter is increased. The locationsButtonsVisible() and locationsButtonsInvisible() helper methods are called. The message is set and the Toast displayed to tell the user they've added the location to their list.
PreferencesActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The button to remove the first item in the user's list.	The number of locations counter is checked. If it is equal to 1, the user is told via a Toast they cannot delete this location until they have added another. If they have more than 1, the location is removed from the ArrayList and the locationsButtonsVisible() and	The number of locations counter is checked. If it is equal to 1, the user is told via a Toast they cannot delete this location until they have added another. If they have more than 1, the location is removed from the ArrayList and the locationsButtonsVisible() and

		locationsButtonsInvisible() helper methods are called.	locationsButtonsInvisible() helper methods are called.
PreferencesActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The button to any other item in the user's list.	The location is removed from the ArrayList, the number of locations counter is decremented, and the visible and invisible helper methods are called.	The location is removed from the ArrayList, the number of locations counter is decremented, and the visible and invisible helper methods are called.
PreferencesActivity onItemSelected() Method The purpose of this test is to check that the onItemSelected method correctly processes the selection of any item in the spinner.	An item is selected from the spinner.	The item at the selected position in the spinner is converted to a String and stored in the currentLocation variable.	The item at the selected position in the spinner is converted to a String and stored in the currentLocation variable.
PreferencesActivity onNothingSelected() Method The purpose of this test is to check that the onNothingSelected method correctly processes the lack of selection of any item in the spinner.	No item has been newly selected (it is not possible to have no item selected at all).	Nothing happens as this method has no body.	Nothing happens as this method has no body.
PreferencesActivity updateLocations() Method The purpose of this test is to check that the user's chosen locations are retrieved from the shared preferences.	n/a	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A counter variable is initialised at 0. The LOCATION_X Strings in the shared preferences are iterated	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A counter variable is initialised at 0. The LOCATION_X Strings in the shared preferences are iterated

		through with a for loop, storing even the nulls in an ArrayList. This ArrayList is iterated through and for every value that isn't a null, the counter is incremented.	through with a for loop, storing even the nulls in an ArrayList. This ArrayList is iterated through and for every value that isn't a null, the counter is incremented.
PreferencesActivity prepTextViews() Method The purpose of this test is to check that the text views are all stored properly.	n/a	All the 10 location text views are stored in an ArrayList.	All the 10 location text views are stored in an ArrayList.
PreferencesActivity prepButtons() Method The purpose of this test is to check that the buttons are all stored properly.	n/a	All the 10 location removal buttons are stored in an ArrayList.	All the 10 location removal buttons are stored in an ArrayList.
PreferencesActivity locationsButtonsVisible() Method The purpose of this test is to check that the buttons and text views that should be visible to the user are made visible.	n/a	The ArrayLists of buttons and text views are iterated through, both for the length of number of locations counter. The visibility to all of these is set to visible. The text for each text view is set to the corresponding value in the locationsList ArrayList.	The ArrayLists of buttons and text views are iterated through, both for the length of number of locations counter. The visibility to all of these is set to visible. The text for each text view is set to the corresponding value in the locationsList ArrayList.
PreferencesActivity locationsButtonsInvisible() Method The purpose of this test is to check that the buttons and text views that should be visible to the user are made invisible.	n/a	The ArrayLists of buttons and text views are iterated through, both for the length of the difference between the length of the ArrayLists and the number of locations counter. The visibility to all of these is set to invisible.	The ArrayLists of buttons and text views are iterated through, both for the length of the difference between the length of the ArrayLists and the number of locations counter. The visibility to all of these is set to invisible.

PreferencesActivity saveLocations() Method The purpose of this test is to check that the user's chosen locations are saved in the shared preferences.	n/a	The sharedPrefs variable is initialised with the getSharedPreferences() method and sharedPrefFile variable as the parameter. The mEditor is also initialised as the return from the edit() method of the sharedPrefs object. The editor is used to clear any existing sharedPreferences. A for loop is entered for the length of the number of locations counter. The locationsList ArrayList is iterated through and for each item in the ArrayList, it is added to the shared preferences in a String called LOCATION_X (where X is the counter/ArrayList position + 1). The mEditor then applies these changes.	The sharedPrefs variable is initialised with the getSharedPreferences() method and sharedPrefFile variable as the parameter. The mEditor is also initialised as the return from the edit() method of the sharedPrefs object. The editor is used to clear any existing sharedPreferences. A for loop is entered for the length of the number of locations counter. The locationsList ArrayList is iterated through and for each item in the ArrayList, it is added to the shared preferences in a String called LOCATION_X (where X is the counter/ArrayList position + 1). The mEditor then applies these changes.
SearchActivity onCreate() Method The purpose of this test is to check that the onCreate method runs correctly when performing tasks that only need to happen once during the activity's life. This includes assigning values to	n/a	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_search.xml. A debug log message is made saying "In the onCreate event handler".	

variables, calling any helper methods, and logging any messages once commands have been successfully executed. From a broader perspective, this is a test to make sure the activity loads correctly.		The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The spinner variables are all assigned their corresponding spinner View components. Each also has a unique adapter made for it, using either a String array resource, or an ArrayList. The spinners all have an onItemSelectedListener attached. All the Buttons are stored in their appropriate variablesThese all have an onClickListener set.	

SearchActivity onStart() Method The purpose of this test is to check that the onStart method runs correctly when making the activity visible to the user and preparing it for interactivity.	n/a	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.	
SearchActivity onPause() Method The purpose of this test is to check that the onPause method runs correctly when indicating that this activity is not in the foreground but should not be destroyed as it is expected to be resumed soon.	n/a	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.	
SearchActivity onStop() Method The purpose of this test is to check that the onStop method runs correctly when a new activity has covered the screen, or when this activity has been terminated.	n/a	The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.	
SearchActivity onDestroy() Method The purpose of this test is to check that the onDestroy method runs correctly when the user has completely finished using the activity.	n/a	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.	

SearchActivity onResume() Method The purpose of this test is to check that the onResume method runs correctly when the user returns to this activity from another or from outside of the app.	n/a	The super class onResume() method is called and a debug log message is made saying “In the onResume event handler”.	
SearchActivity onClick() Method The purpose of this test is to check that the onClick method does not run unless a button is pressed.	No button on the interface is pressed.	Nothing happens as no methods are called after the interface is displayed.	Nothing happens as no methods are called after the interface is displayed.
SearchActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The first button is clicked.	The onClick method/event handler is called, sending the buttonSearch1 view as a parameter. A new context object is created and assigned the value of the SearchActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. A Class object is created to store the SearchResultsByHourActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using	The onClick method/event handler is called, sending the buttonSearch1 view as a parameter. A new context object is created and assigned the value of the SearchActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. A Class object is created to store the SearchResultsByHourActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using

		the newly created intent as the parameter, launching the search results screen.	the newly created intent as the parameter, launching the search results screen.
SearchActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The second button is clicked.	The onClick method/event handler is called, sending the buttonSearch2 view as a parameter. A new context object is created and assigned the value of the SearchActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. A Class object is created to store the SearchResultsValueActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the search results screen.	The onClick method/event handler is called, sending the buttonSearch2 view as a parameter. A new context object is created and assigned the value of the SearchActivity context. A debug log message is made saying “In the onClick method”. The if statement is entered. A Class object is created to store the SearchResultsValueActivity.class object. An Intent object is created using the context object and the destination class object. The startActivity() method is called using the newly created intent as the parameter, launching the search results screen.
SearchActivity onClick() Method The purpose of this test is to check that the onClick method processes the clicking of each button correctly.	The third button is clicked.	The onClick method/event handler is called, sending the buttonDeleteAll view as a parameter. The deleteForecastsDB() helper method is called.	The onClick method/event handler is called, sending the buttonDeleteAll view as a parameter. The deleteForecastsDB() helper method is called.

		A String message is declared and initialised, shown in a Toast object to tell the user their Forecasts have been deleted.	A String message is declared and initialised, shown in a Toast object to tell the user their Forecasts have been deleted.
SearchActivity onItemSelected() Method The purpose of this test is to check that the onItemSelected method correctly processes the selection of any item in the spinner.	An item is selected from the spinner.	The item from the spinner which has just been selected is stored in the corresponding global variable. Some of these are converted before doing so.	The item from the spinner which has just been selected is stored in the corresponding global variable. Some of these are converted before doing so.
SearchActivity onNothingSelected() Method The purpose of this test is to check that the onNothingSelected method correctly processes the lack of selection of any item in the spinner.	No item has been newly selected (it is not possible to have no item selected at all).	Nothing happens as this method has no body.	Nothing happens as this method has no body.
SearchActivity fetchLocations() Method The purpose of this test is to check that the user's chosen locations are retrieved from the shared preferences.	n/a	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. A counter variable is initialised. A while loop is entered. Which goes through all the LOCATION_X Strings in the shared preferences until one returns a null.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. A counter variable is initialised. A while loop is entered. Which goes through all the LOCATION_X Strings in the shared preferences until one returns a null.

		If a String does not return a null, it is logged with a debug message before it is added to the chosenLocations ArrayList.	If a String does not return a null, it is logged with a debug message before it is added to the chosenLocations ArrayList.
SearchActivity saveSearchItems() Method The purpose of this test is to check that the user's chosen search terms are saved in the shared preferences.	An id to tell the method which button has been pressed.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. The editor is also set up. An if statement is used to determine which sets of values are being put into the shared preferences. Each of these are put into Strings with appropriate labels and the editor applies these changes.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. The editor is also set up. An if statement is used to determine which sets of values are being put into the shared preferences. Each of these are put into Strings with appropriate labels and the editor applies these changes.
SearchActivity deleteForecastsDB() Method The purpose of this test is to check that the task is executed.	n/a	A new DeleteForecastsTask object is created and told to execute.	A new DeleteForecastsTask object is created and told to execute.
SearchActivity - DeleteForecastsTask doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to delete all forecasts from the database for a specific location	n/a	The forecastDAO object is used to call its deleteAllForecasts() method.	The forecastDAO object is used to call its deleteAllForecasts() method.

SearchActivity - *DeleteForecastsTask* onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A Void object.	The super class's onPostExecute method is called using the Void object as the parameter.	The super class's onPostExecute method is called using the Void object as the parameter.
ForecastDatabase As this code is the same as that used in the labs, minus the modifications of the names for classes etc, this class was not individually tested, but observed during its interactions with other classes/methods. The class performed as expected and operated perfectly.	n/a		
Forecast As this code is simply getters and setters generated by Android Studio, this class was not tested individually, but observed during its interactions with other classes/methods. The class performed as expected and operated perfectly.	n/a		

ForecastLocation As this code is simply getters and setters generated by Android Studio, this class was not tested individually, but observed during its interactions with other classes/methods. The class performed as expected and operated perfectly.	n/a		
HttpDownloader As this code is the same as that used in the labs, this class was not individually tested, but observed during its interactions with other classes/methods. The class performed as expected and operated perfectly.	n/a		
Converter weatherDesc() Method The purpose of this test is to check that the weather description is correctly formatted.	A String containing the original description, all in lowercase.	The String is checked for the character at the start being converted to uppercase. Then, any additional characters which follow a space are also converted to uppercase. The converted String is returned.	The String is checked for the character at the start being converted to uppercase. Then, any additional characters which follow a space are also converted to uppercase. The converted String is returned.
Converter windDirection() Method	A double containing the original value for	The double is put through an if statement to determine which description should be returned.	The double is put through an if statement to determine which description should be returned.

The purpose of this test is to check that the wind direction is accurately converted to a description.	the wind direction.	The values used were calculated by hand.	
Converter windSpeed() Method The purpose of this test is to check that the wind speed is accurately converted to a description.	A double containing the original value for the wind direction.	The double is put through an if statement to determine which description should be returned. The categories used were from the Beaufort scale as described on Wikipedia.	The double is put through an if statement to determine which description should be returned.
Converter time() Method The purpose of this test is to check that the time of the forecast is correctly extracted from the input.	A String containing the date and time in the format YYYY-MM-DD HH:MM:SS	The characters 11-15 (HH:MM) are extracts from the String as a substring and returned.	The characters 11-15 (HH:MM) are extracts from the String as a substring and returned.
Converter date() Method The purpose of this test is to check that the date of the forecast is correctly extracted from the input	An ArrayList containing the date in parts.	The parts are concatenated together as a single String and returned in the format DDMMYYYY.	The parts are concatenated together as a single String and returned in the format DDMMYYYY.
Converter date() Method The purpose of this test is to check that the date of the forecast is correctly converted to a different format.	A String containing the date in the format DDMMYYYY.	The date parts are extracted into different substrings. The day of the week is found by calling the findDayOfWeek() method. 2 case statements and the day of the week are concatenated together to	The date parts are extracted into different substrings. The day of the week is found by calling the findDayOfWeek() method. 2 case statements and the day of the week are concatenated together to

		form the returned String in the format e.g. Monday 1st December.	form the returned String in the format e.g. Monday 1st December.
Converter dateParts() Method The purpose of this test is to check that the date of the forecast is correctly separated into different parts within an ArrayList.	A String containing the date in the format YYYY-MM-DD.	An ArrayList is returned with the day, month, and year all as separate Strings within the ArrayList.	An ArrayList is returned with the day, month, and year all as separate Strings within the ArrayList.
Converter findDayOfWeek() Method The purpose of this test is to check that the correct day of the week is found for date of the forecast using the Key/Vaulue method from mathforum.org	3 Strings: the day, the month and the year. OR A String with the date in the format DDMMYYYY.	The Key/Value method is utilised through a series of switch and if statements to return a day of the week.	The Key/Value method is utilised through a series of switch and if statements to return a day of the week.
ForecastDao insertForecasts() Method The purpose of this test is to check that the array of Forecast objects is inserted correctly into the database.	An array of Forecast objects.	The Room @Insert tag will deal with the implementation of the method, but all Forecast objects within the given array are added to the database.	The Room @Insert tag will deal with the implementation of the method, but all Forecast objects within the given array are added to the database.
ForecastDao getAllForecastsForLocation() Method	A String with the city name.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct	The Room @Query tag will deal with the implementation of the method, but the query itself is correct

The purpose of this test is to check that the return list of Forecast objects is correct		meaning that all Forecast objects returned are those relating to the given location.	meaning that all Forecast objects returned are those relating to the given location.
ForecastDao deleteAllForecasts() Method The purpose of this test is to check that all Forecasts are deleted.	n/a	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects are deleted.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects are deleted.
ForecastDao deleteAllForecastsFor() Method The purpose of this test is to check that the Forecasts relating to the given location are all deleted.	A String with the city name.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects related to the given location are deleted.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects related to the given location are deleted.
ForecastDao getAllForecastsHourly() Method The purpose of this test is to check that the Forecasts relating to the given location within the specified time range are returned.	A String with the city name and an int specifying how many rows to return.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects related to the given location in the time range are returned.	The Room @Query tag will deal with the implementation of the method, but the query itself is correct meaning that all Forecast objects related to the given location in the time range are returned.
ForecastDao getAllForecastsGreaterValue() Method The purpose of this test is to check that the Forecasts relating to the given location with the search term having a value greater than or equal to the specified value are returned.	A String with the city name and an int specifying how many rows to return.	The Room @Query tag will deal with the implementation of the method, but all Forecast objects related to the given location with the search term having a value greater than or equal to the specified value are returned.	The Room @Query tag will deal with the implementation of the method, but the query itself is incorrect somewhere as ALL Forecast objects for the specified location are returned.

ForecastDao getAllForecastsLessValue() Method The purpose of this test is to check that the Forecasts relating to the given location with the search term having a value less than or equal to the specified value are returned.	A String with the city name and an int specifying how many rows to return.	The Room @Query tag will deal with the implementation of the method, but all Forecast objects related to the given location with the search term having a value less than or equal to the specified value are returned.	The Room @Query tag will deal with the implementation of the method, but the query itself is incorrect somewhere as ALL Forecast objects for the specified location are returned.
SearchResultsByHourActivity onCreate() Method The purpose of this test is to check that the onCreate method runs correctly when performing tasks that only need to happen once during the activity's life. This includes assigning values to variables, calling any helper methods, and logging any messages once commands have been successfully executed. From a broader perspective, this is a test to make sure the activity loads correctly.	n/a	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_search_results_by_hour.xml. A debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_search_results_by_hour.xml. A debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from

		getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The fetchParams() helper method is called. The search results text view is stored in a variable and then the searchForecastsDB() helper method is called.	getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The fetchParams() helper method is called. The search results text view is stored in a variable and then the searchForecastsDB() helper method is called.
SearchResultsByHourActivity onStart() Method The purpose of this test is to check that the onStart method runs correctly when making the activity visible to the user and preparing it for interactivity.	n/a	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.	The super class onStart() method is called and a debug log message is made saying “In the onStart event handler”.
SearchResultsByHourActivity onPause() Method The purpose of this test is to check that the onPause method runs correctly when indicating that this activity is not in the foreground but should not be destroyed as it is expected to be resumed soon.	n/a	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.
SearchResultsByHourActivity onStop() Method	n/a	The super class onStop() method is called and a debug log message is	The super class onStop() method is called and a debug log message is

The purpose of this test is to check that the onStop method runs correctly when a new activity has covered the screen, or when this activity has been terminated.		made saying “In the onStop event handler”.	made saying “In the onStop event handler”.
SearchResultsByHourActivity onDestroy() Method The purpose of this test is to check that the onDestroy method runs correctly when the user has completely finished using the activity.	n/a	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.
SearchResultsByHourActivity onResume() Method The purpose of this test is to check that the onResume method runs correctly when the user returns to this activity from another or from outside of the app.	n/a	The super class onResume() method is called and a debug log message is made saying “In the onResume event handler”.	The super class onResume() method is called and a debug log message is made saying “In the onResume event handler”.
SearchResultsByHourActivity fetchParams() Method The purpose of this test is to check that the search terms are retrieved correctly from the shared preferences.	n/a	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. The current location and the amount of hours to search for are retrieved from the shared preferences.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. The current location and the amount of hours to search for are retrieved from the shared preferences.
SearchResultsByHourActivity searchForecastsDB() Method	n/a	A new SearchForecastsTask object is created and told to execute.	A new SearchForecastsTask object is created and told to execute.

The purpose of this test is to check that the search task is made and told to execute.			
SearchResultsByHourActivity - *SearchForecastsTask* doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to find all forecasts for a specification to show the temperature within a certain time frame.	n/a	The forecastDAO object is used to call its getAllForecastsHourly() method.	The forecastDAO object is used to call its getAllForecastsHourly () method.
SearchResultsByHourActivity - *SearchForecastsTask* onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A List of Forecast objects.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox.
SearchResultsValueActivity onCreate() Method The purpose of this test is to check that the onCreate method runs correctly when performing tasks that only need to	n/a	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_search_results_value.xml. A	The super class onCreate() method is called with the savedInstanceState Bundle, and the layout of the Activity is set to activity_search_result_value.xml. A

happen once during the activity's life. This includes assigning values to variables, calling any helper methods, and logging any messages once commands have been successfully executed. From a broader perspective, this is a test to make sure the activity loads correctly.		debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The fetchParams() helper method is called. The search results text view is stored in a variable and then the searchForecastsDB() helper method is called.	debug log message is made saying "In the onCreate event handler". The forecastDB field is assigned a value – the return value from the getDatabase() method in the ForecastDatabase class. The forecastDAO field is assigned the return of the forecastDao() method from the forecastDB field. A log debug message is made saying "Initialised database okay". The sharedPrefs field is assigned the return value from getSharedPreferences, using the sharedPrefFile String as a parameter. mEditor is assigned the return from the edit() method of the sharedPrefs field. The fetchParams() helper method is called. The search results text view is stored in a variable and then the searchForecastsDB() helper method is called.
SearchResultsByHourActivity onStart() Method	n/a	The super class onStart() method is called and a debug log message is	The super class onStart() method is called and a debug log message is

The purpose of this test is to check that the onStart method runs correctly when making the activity visible to the user and preparing it for interactivity.		made saying “In the onStart event handler”.	made saying “In the onStart event handler”.
SearchResultsByHourActivity onPause() Method The purpose of this test is to check that the onPause method runs correctly when indicating that this activity is not in the foreground but should not be destroyed as it is expected to be resumed soon.	n/a	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.	The super class onPause() method is called and a debug log message is made saying “In the onPause event handler”.
SearchResultsByHourActivity onStop() Method The purpose of this test is to check that the onStop method runs correctly when a new activity has covered the screen, or when this activity has been terminated.	n/a	The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.	The super class onStop() method is called and a debug log message is made saying “In the onStop event handler”.
SearchResultsByHourActivity onDestroy() Method The purpose of this test is to check that the onDestroy method runs correctly when the user has completely finished using the activity.	n/a	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.	The super class onDestroy() method is called and a debug log message is made saying “In the onDestroy event handler”.
SearchResultsByHourActivity onResume() Method	n/a	The super class onResume() method is called and a debug log message is	The super class onResume() method is called and a debug log message is

The purpose of this test is to check that the onResume method runs correctly when the user returns to this activity from another or from outside of the app.		made saying “In the onResume event handler”.	made saying “In the onResume event handler”.
SearchResultsByHourActivity fetchParams() Method The purpose of this test is to check that the search terms are retrieved correctly from the shared preferences.	n/a	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. The current location, the search term, the search value, and whether the user wants the results greater than/equal to are retrieved from shared preferences.	The sharedPrefs variable is initialised using the getSharedPreferences() method and the sharedPrefFile parameter. A debug log message is made to confirm this has happened. The current location, the search term, the search value, and whether the user wants the results greater than/equal to are retrieved from shared preferences.
SearchResultsByHourActivity searchForecastsDB() Method The purpose of this test is to check that the search task is made and told to execute.	n/a	Depending on the searchTerm value, an if/else if/else statement will be used to guide which task object is created. Nested if statements help further this guidance based on whether the user wants greater/less than values. The chosen task object is then created and told to execute.	Depending on the searchTerm value, an if/else if/else statement will be used to guide which task object is created. Nested if statements help further this guidance based on whether the user wants greater/less than values. The chosen task object is then created and told to execute.
SearchResultsByHourActivity - *SearchGreaterTemperatureForecastsTask* *SearchGreaterHumidityForecastsTask* *SearchGreaterWindSpeedForecastsTask*	n/a	The forecastDAO object is used to call its getAllForecastsGreaterValue() method.	The forecastDAO object is used to call its getAllForecastsGreaterValue() method.

doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to find all forecasts matching the user's specification.			
SearchResultsByHourActivity - *SearchGreaterTemperatureForecastsTask* *SearchGreaterHumidityForecastsTask* *SearchGreaterWindSpeedForecastsTask* onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A List of Forecast objects.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox, but these results are not accurate as they are just for EVERY forecast for the chosen location.
SearchResultsByHourActivity - *SearchLessTemperatureForecastsTask* *SearchLessHumidityForecastsTask* *SearchLessWindSpeedForecastsTask* doInBackground() Method The purpose of this test is to check that the method interacts with the DAO to find all forecasts matching the user's specification.	n/a	The forecastDAO object is used to call its getAllForecastsLessValue() method.	The forecastDAO object is used to call its getAllForecastsLessValue () method.

SearchResultsByHourActivity *- SearchLessTemperatureForecastsTask* *SearchLessHumidityForecastsTask* *SearchLessWindSpeedForecastsTask* onPostExecute() Method The purpose of this test is to check that the method correctly finishes the job of the task.	A List of Forecast objects.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox.	The super class's onPostExecute method is called using the Void object as the parameter. If the array size is 0, the user is told no results could be found. If the array size is greater than 0, the results are displayed in the textbox, but these results are not accurate as they are just for EVERY forecast for the chosen location.