Testing

# Introduction

As outlined in the design document we will use a variety of techniques to fully test the plugin to ensure that it meets the requirements of our client and users. The testing will comprise of alpha and beta testing using a combination of white and black-box testing.

# Test Types

The tests undertaken will fall into one of the following categories:

* **Validation** – Tests that ensure no erroneous or dangerous data can be entered into the system.
* **Functionality** – Tests that will evaluate a specific function or the functionality of a specified component.
* **Output** – These will test the output result from a component or function.
* **Stress** - Tests that will determine how well the system handles large demanding data.

# Test Environment

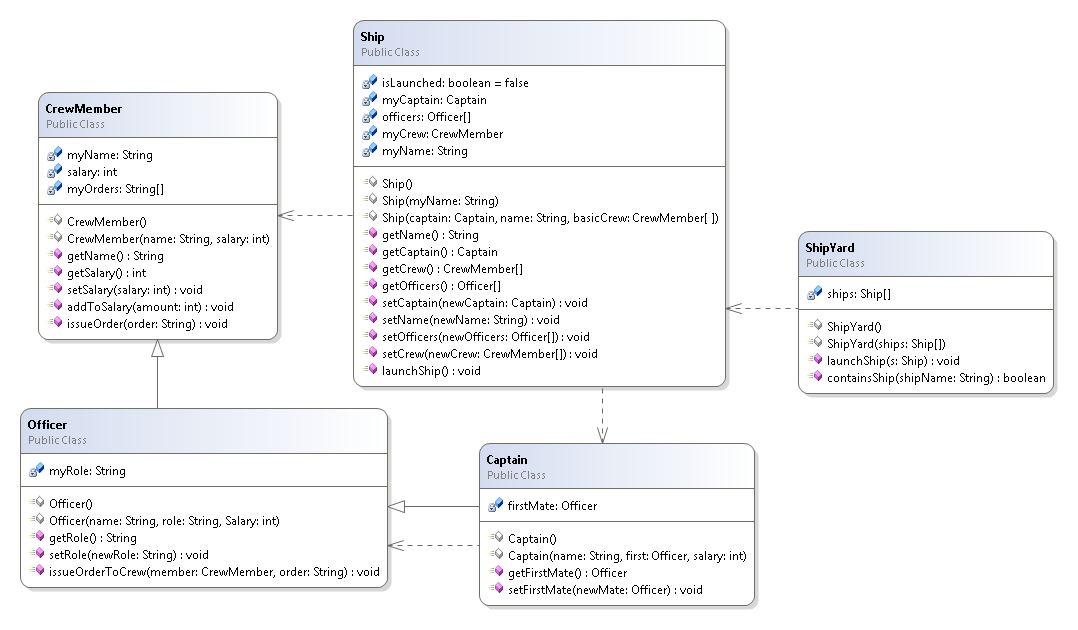
To test the plugin we will be using a clean install of Eclipse Luna with no additional plugins, this ensures that there are no conflicting issues with other tools and external utilities. Alpha testing will be performed on the same PCs the software was developed in but beta testing will be conducted in a computer lab within the University of Huddersfield. A user project will be created with a range of classes, each class will have different constructors and methods along with a variety of variables to test different types and how the user code is handled.

Each test will be run a minimum of one time with repeat tests only being conducted if bugs are fixed before the project reaches a conclusion.

# Test Data

## Sample User Project

### Class Diagram for Sample Project



### Variable Types

We will use a variety of different variable types whilst testing the plugin:

* **Numeric**
* **String**
* **User Class (Name tomorrow)**
* **Array (String, Numeric, User Class)**
* **Boolean**

### Sample Methods

A few of the methods contained in the sample project will be used to determine that the system is working correctly. We will measure whether the return value is of the correct type and the result is correct. We will use some of the methods to both set and get variables and perform actions to ensure that the effects on other instances remain consistent with the objects on the object bench.

### Constructors

Constructors will be created in order to test and ensure a variety of class constructors are supported and that instances can be created onto the object bench with a variety of different parameters.

## Utilities

### Java Model

Table 0-8

|  |  |  |
| --- | --- | --- |
| Type | Value | Notes |
|  |  |  |
|  |  |  |
|  |  |  |

### String Helper

Table 0-9

|  |  |  |
| --- | --- | --- |
| Type | Value | Notes |
|  |  |  |
|  |  |  |
|  |  |  |

### Object Bench Utility

Table 0-10

|  |  |  |
| --- | --- | --- |
| Type | Value | Notes |
|  |  |  |
|  |  |  |
|  |  |  |

# Test Plan

## Object Bench

Table 0-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | Instance(CrewMember) | To test if an instance can be removed from the bench. | The instance will be removed from the bench. |
| 2 | Functionality | Instance(CrewMember)x3  Instance(Ship)  Instance(Captain) | To test if the object bench can be cleared with the clear bench function. | The bench will be cleared and instances destroyed. |
| 3 | Functionality  /Output | Instance(Ship)  Method(launchShip) | To test if a method can be called, in this case the launchShip() method which simply switches a boolean flag from false to true. | The system should return a value of true displayed in a dialog informing the user of the result of the methods return call. |
| 4 | Output | Instance(CrewMember)  Method(addToWage) | To test if a method can be called and output results based on the input parameters. The method here takes a parameter of integer, this number is then added to the salary field, and the result will be the salary plus the integer. | The system will display the added value in a dialog along with the return type. |
| 5 | Functionality | Instance(Ship) | To test when an instance is “selected” via a single click its fields are shown in the inspector view. | The fields of the selected instance will be displayed in the inspector view. |
| 6 | Stress | Various Instances (x100) | A test to see if the object bench can handle 100 instances. | The object bench should handle the instances but may begin to cause minor issues with performance. |
| 7 | Functionality | Instance(Ship) | Testing to ensure that all the methods are available in the right-click context menu ready to be called. The methods that should be displayed can be found in the class diagram of the sample project. | There will be a list of methods that belong to the selected instance (Ship). |

## Class Diagram

Table 0-2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | New Class(Name: “TestClass” Template: Standard) | A simple test to ensure a new class can be added to a selected package. | A new class will be created and displayed in the class diagram view. |
| 2 | Functionality | New Package(Name: “TestPackage”) | A test to ensure a new package can be added to the system. | A new package will be added to the project and displayed in the class diagram view. |
| 3 | Functionality | Class(TestClass) | To test that a class can be deleted. | The class will be deleted and the rendered object removed. |
| 4 | Functionality | Package(TestPackage) | To test that a package can be deleted. | The package will be deleted and removed from view. |
| 5 | Functionality | Class(Ship)  Constructor( | To test that an instance can be added to the object bench using a zero parameter constructor. | An instance of the “Ship” class will be added to the object bench. |
| 6 | Functionality | Class(Ship) Constructor(String “Orion”) | To test that an instance can be created with a parameter. The constructor selected will set the value of a ships name; we will be able to validate this by inspecting the created instance. | The instance will be created and the name selected will be displayed in the relevant field on the inspector. |
| 7 | Validation | NewPackage(“package.test”) | To test whether a package name can contain the package word, an invalid parameter. | The system should reject the entry with an error message. |
| 8 | Validation | NewClass(“Test Class”) | To test whether a class can have a space in its name. | The system should reject the entry with an error message. |
| 9 | Output | Class(Officer, CrewMember) | To test if the inheritance of Officer is shown on the class diagram view. | The inheritance will be represented by the class diagram view. |
| 10 | Output | Class(Ship,CrewMember) | To test if the dependence is drawn between two classes when one class depends on the other. | The class diagram should represent the dependency between the two classes with a dotted line. |
| 11 | Output | Sample Project | To test if the class diagram view accurately draws the relationships between the classes in a project. | The class diagram view should represent the project in a similar manner to the class diagram outlined in the test data. |
| 12 | Validation | NewClass(“TestClass”)  NewClass(“TestClass”) | To test if multiple classes of the same name can be added. | The system should overwrite the previous instance and allow the new version of the class to be loaded. |
| 13 | Stress | Sample Project + 20 new classes. | To test if the system can handle more than 10 classes at a time. | The system should handle it but the diagram may be difficult to read and understand. |
| 14 | Functionality | Class(Ship) | To test if a class can be opened in the code editor when double clicked or selected from the context menu from the view. | A code editor window will open for the selected class. |
| 15 | Functionality | Class(Ship) | To test if the correct constructors are present in the context menu associated with the class. These can be found in the class diagram for the sample project. | The constructors will be correct and selectable. |

## Inspector

Table 0-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | Instance(CrewMember) | A simple test to ensure that a selected instance displays in the inspector. | The instance will be displayed in the inspector. |
| 2 | Validation | Instance(TestClass)  Fields(int,String,boolean) | A test to ensure that the correct controls are retrieved for each type. | The instance will be displayed in the inspector with a Spinner, Text Field and Check Box for entering values for the fields. |
| 3 | Validation/  Output | Instance(CrewMember)  Array(myOrders)  Size(3)  Values([1]”Test Order”  [2]”Sample Order”  [3]”Last Order”) | A test to ensure that an array can be re-sized and values entered into each item within the array. | The values within the array will be set as required and the resizing will work as intended. |
| 4 | Validation | Instance(Ship)  Field(myName)  Value(“Delta”) | A test to enter a name for the ship. | The name will be accepted and the stored. |
| 5 | Output | Instance(Ship)  Method(getName) | Testing to ensure that a newly entered value will be accessible from a method. | The name changed from ”Orion” to “Delta” will be displayed in a dialog when then method is called from the object bench. |
| 6 | Functionality | Instance(Ship)  Instance(Captain) x 3  SelectedInstance(Ship)  Field(myCaptain)  FieldValue(Captain2) | A test to see if an instance can be assigned to another instance as a field. | The instances will be displayed as options within the inspector for the required field. The field can then be set and a Captain selected. |

## Dialogs

### Constructor Dialog

Table 0-4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | Class(Ship)  Constructor(Captain, String)  Instance(Captain) x2 | A test to ensure that the correct controls are provided when creating a new instance through the constructor dialog. | There should be a text field and a drop down menu to select an instance for the construction of the instance. |
| 2 | Functionality | Class(Ship) | Test that a construction of an instance can be cancelled. | The window should close and no instance will be created. |
| 3 | Validation | Class(Officer)  Constructor(String, String, int) | To ensure string values cannot be entered into int spinners. | The value should be rejected. |
| 4 | Stress | NewClass(“TempNewClass”)  Constructor(String, String, int, int, int, int, int, int, int, int, int) | To test a long constructor. | The dialog should expand and accommodate all controls. |
| 5 | Validation/  Output/  Functionality | Class(Ship)  Constructor(Captain, String, CrewMembers) | To test if an instance can be used from the object bench, in this case the Captain field. | The field should be selectable from instances on the object bench. |
| 6 | Validation | Class(Ship)  Constructor(Captain, String, CrewMembers) | To test if a user defined field can be set to null. | The system should allow the user to set an instance value to null. |

### Method Call Dialog

Table 0-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | Class, Instance(Ship)  Method(getName) | Test that a method is called correctly. | The ships name either “Orion” or “Delta” will be presented in a dialog. |
| 2 | Functionality | Class, Instance(CrewMember)  Method(addToSalary) | To test that parameters can be added to a method call and affect the result. | The method will accept the parameter and return a relevant value. |
| 3 | Validation | Class, Instance(Officer)  Instance(CrewMember) x2  Method(giveOrderToCrewMember)  Parameters(CrewMember, String)  Values(CrewMember1, “Test Order”) | To test that the correct controls are created for the parameters in the method. | In this instance the dialog should display controls for a CrewMember represented by a drop down menu and a text field to represent the String value. |
| 4 | Validation/  Output | Instance(Officer) x 3  Instance(Ship)  Method(setOfficers)  Parameters(Officer[])  Array(officers)  Size(3)  Values([1]Officer 1  [2]Officer 2  [3]Officer 3) | To test that arrays can be used as a parameter in methods. | The array will be fully editable and the values will be set as required. |

### New Class Dialog

Table 0-6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | NewClass(“TestClass”)  Template(“Default”) | To test if a class can be created, this is the same process as creating a class from the class diagram so the result should be the same. | A new class will be created using the default template. |
| 2 | Validation/  Output | NewClass(“TestClass”)  Template(ALL) | To test if all templates work when requested. | A class can be created with each template type. |
| 3 | Output | NewClass(“TestClass”) | To test if a new class is named correctly and core syntax is correct. | The core syntax will be correct and both class and package names will be correct. |

### New Package Dialog

Table 0-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | NewPackage(“one”) | To test a package can be created. | The package will be created. |
| 2 | Functionality | NewPackage(“one.more.time”) | To test a multi-depth package can be created. | Multiple packages will be created providing the depth as required. |
| 3 | Validation | NewPackage(“package.test”) | To test whether a package name can contain the package word, an invalid parameter. | The system should reject the entry with an error message. |

## Utilities

### Java Model

Table 0-8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### String Helper

Table 0-9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality/  Validation | “TestClass.java” | Test that a string is stripped of its extension. | The resulting string will be “TestClass” |
| 2 | Functionality/  Validation/  Output | “QString” “I” “[I” | This test will enable us to test the “fixType” method. Fixing type allows for correct class loading. | The string helper will output “String” “int” “int[]”. |
| 3 | Output | “com.test” “TestClass” | Testing the method to get qualified name. | The string helper will return “com.test.TestClass” |

### Object Bench Utility

Table 0-10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Data | Description | Expected Result |
| 1 | Functionality | “I” | Test whether a class is known. This tests the method “isKnown”. | The method will return true as the class is known to be an integer (int). |
| 2 | Functionality | int, String, boolean, Object(Ship) | This is a test to ensure that the correct controls are returned when requested from the getControl method. | Should be returns of Spinner, Text Field, Check Box and Drop Down Menu respectively. |
| 3 | Validation | “asdfawe” | Test whether unknown information is known in method “isKnown”. | The method will return false. |
| 4 | Output | Inspector View | A test to see if the inspector view is set on start-up. This will be proven if we can select instances from the object bench and see their values. | The inspector view will be set as required. |

# Test Results

## Object Bench

Table 0-1

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

## Class Diagram

Table 0-2

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

## Inspector

Table 0-3

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

## Dialogs

### Constructor Dialog

Table 0-4

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

### Method Call Dialog

Table 0-5

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

### New Class Dialog

Table 0-6

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

### New Package Dialog

Table 0-7

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

## Utilities

### Java Model

Table 0-8

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

### String Helper

Table 0-9

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

### Object Bench Utility

Table 0-10

|  |  |  |
| --- | --- | --- |
| Number | Result | Action |
|  |  |  |
|  |  |  |
|  |  |  |

# Unit Testing

# Beta Testing

## Client & User Feedback

# Educational Testing

# Action Plan

# Conclusions

# Appendix A: Test Screenshots

## Object Bench