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

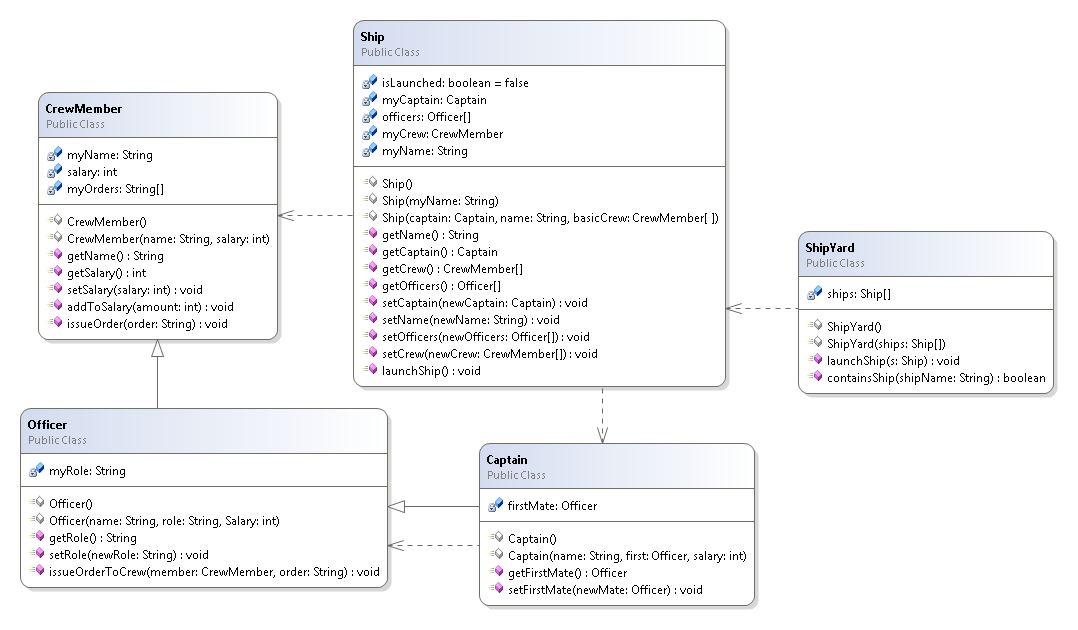


Figure 1 - Sample project class diagram

### 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

### String Helper

Table 0-9

|  |  |  |
| --- | --- | --- |
| Type | Value | Notes |
| String | “TestClass.java” |  |
| String | “QString |  |
| String | “I” |  |
| String | “[I” |  |
| String | “com.test” |  |
| String | “TestClass” |  |

### Object Bench Utility

Table 0-10

|  |  |  |
| --- | --- | --- |
| Type | Value | Notes |
| String | “I” |  |
| String | “asdfawe” |  |
| Class | Int |  |
| Class | String |  |
| Class | boolean |  |

# 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 Helper

Throughout development of the project the java model has been tested by using print statements to ensure the correct results throughout. However due to the nature of the class it is very difficult to test independently as it relies on some aspects of the IDE to be working that we have no access to manipulate and set based on parameters we wish to define.

However it stands to reason that if the Java Model Helper were erroneous then the entire system would fail as this is the backbone upon which the majority of the plugin is built. The Java Model Helper allows for traversal of classes and packages throughout a user project, this allows the user to interact with their code. This can be verified as working if the majority of other functions throughout the plugin are working as intended.

### 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 | As expected the instance could be removed from the object bench. | Not required. |
| 2 | The object bench was cleared as expected. | Not required. |
| 3 | As expected a void result is returned and the ships launch status is changed from false to true. | Not required. |
| 4 | The method accepts a parameter and uses it to alter a field within the class. This behaviour is as expected. | Not required. |
| 5 | As expected a selected instances values were shown in the inspector view. | Not required. |
| 6 | The bench accepted all instances however the window did not render fully so some instances are “cut off”. | Re-size the window and enable expandable scrolling in the GUI. |
| 7 | As expected the methods available to be called were placed in a context menu. | Not required. |

## Class Diagram

Table 0-2

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | A new class was created and added to the view. However a mistake on the tester’s part did reveal a bug, if a white space is added to the end of the name the system will reject adding a new class. | A simple check to remove additional white space at the end of a field name will need to be implemented to prevent errors. |
| 2 | As expected a new package is added to the project and reflected on the class diagram view.  There is a minor rendering issue with class names. | Force the number of classes onto a new line. |
| 3 | As expected the class is deleted from both the project and diagram. | Not required. |
| 4 | As expected the package is deleted from the project and diagram. | Not required. |
| 5 | The Ship instance is created with a zero parameter constructor and placed on the object bench. | Not required. |
| 6 | The ship is created with constructor with a single parameter; the value is set as expected and can be seen in the inspector. | Not required. |
| 7 | The package is rejected as expected. | Not required. |
| 8 | The new class is rejected as expected. | Not required. |
| 9 | NOT YET IMPLEMENTED |  |
| 10 | NOT YET IMPLEMENTED |  |
| 11 | NOT YET IMPLEMENTED |  |
| 12 | The system accepted the duplicate class however it is suspected that the new version of the class is not loaded into the classloader so that it can be used. | There will need to be a check implemented to prevent users adding classes with the same name as another class without first removing the offending class. |
| 13 | NOT YET IMPLEMENTED |  |
| 14 | As expected, double clicking a class from the class diagram view opens a maximised code editor. | Not required. |
| 15 | The correct constructors are displayed in the context menu for the constructors. | Not required. |

## Inspector

Table 0-3

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | As expected, an instance can be viewed within the inspector view. | Not required. |
| 2 | The correct controls for the variable types are displayed for editing. | Not required. |
| 3 | The array could be resized and values added. | Not required. |
| 4 | The name is accepted and stored as expected. | Not required. |
| 5 | The new name is accepted and the value stored in a variable can be retrieved by a method returning the value of the stored variable. | Not required. |
| 6 | An instance could be assigned to another instance as a field. | Not required. |

## Dialogs

### Constructor Dialog

Table 0-4

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | The correct controls of TextField and Drop Down Selector were displayed in the dialog. | Not required. |
| 2 | The construction of an instance could be cancelled and the window closed. | Not required. |
| 3 | The spinner would not allow string values to be entered. | Not required. |
| 4 | The long constructor could be used as expected. It was noticed however that the scroll bar did not appear to allow for longer classes. | The scroll bar needs to be investigated to ensure that it is created for longer constructors. |
| 5 | An instance from the object bench could be used in the construction of a new instance. The field was selectable as required. | Not required. |
| 6 | A user field requiring an instance could be set to null as expected. | Not required. |

### Method Call Dialog

Table 0-5

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | As expected the method is called correctly and the name is returned in a dialog. | Not required. |
| 2 | The parameters entered affected the result returned in the dialog as expected. | Not required. |
| 3 | The parameters in the method dialog are represented by the correct controls. In this case the controls created are a drop down menu and a text box as expected. | Not required. |
| 4 | The arrays can be used as a parameter as expected. The array can be resized and the individual fields within can be set and utilised. | Not required. |

### New Class Dialog

Table 0-6

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | A class can be created as expected and added to the class diagram; this class uses the default template. | Not required. |
| 2 | The templates all worked. It could be argued some of the templates may need a minor amount of user effort to integrate into the project. This would be mitigated and fixed in later versions. | Minor edits need to be made to certain templates to ensure that the details regarding packages and imports work as intended. |
| 3 | A class can be created in a nested package and both names and syntax of the created class are correct. | Not required. |

### New Package Dialog

Table 0-7

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | As expected a packaged called “one” can be created. | Not required. |
| 2 | A multi-depth package can be created as expected. | Not required. |
| 3 | The package name cannot contain the word “package” and an error is thrown. This is expected behaviour. | Not required. |

## Utilities

### String Helper

Table 0-9

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | The string is stripped of the extension “.java” as expected resulting in a string of “TestClass”. | Not required. |
| 2 | As expected the fix type method returned the correct class names for abstract variable type names such as “I” for int/integer, “QString” for String and “[I” for int[] (an array of integers) | Not required. |
| 3 | A fully qualified name is returned as expected. | Not required. |

### Object Bench Utility

Table 0-10

|  |  |  |
| --- | --- | --- |
|  | Result | Action |
| 1 | The test returns true, the class is known as expected. In this case the string “I” is known to be an integer. | Not required. |
| 2 | The correct controls are returned as expected. In this instance a Spinner, Text Field, Check Box and Drop Down menu are returned as required. | Not required. |
| 3 | The method returns false as expected. | Not required. |
| 4 | The inspector view will initialise as expected, instances can be selected and viewed in the inspector. This proves that the inspector can be set in the object bench utility. | Not required. |

# Unit Testing

# Beta Testing

## Client & User Feedback

# Observations During Testing

* Dialog titles.

# Educational Testing

# Action Plan

# Conclusions

# Appendix A: Test Screenshots

## Object Bench

### Test 1

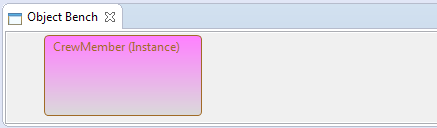


Figure 2 - Before deleted instance.

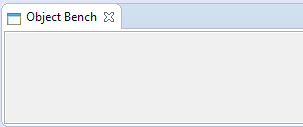


Figure 3 - After deleted instance.

### Test 2

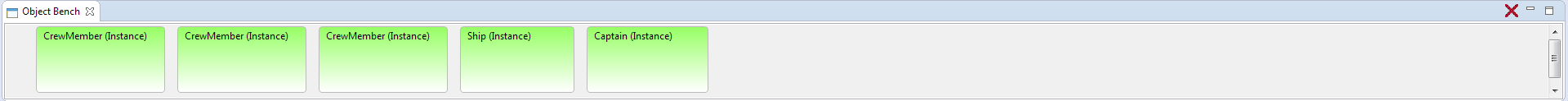


Figure 4 - Before clearing.

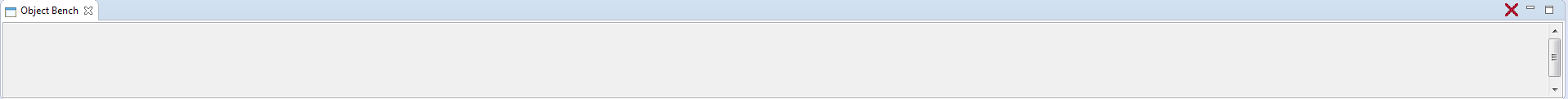


Figure 5 - After clearing.

### Test 3

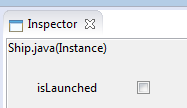


Figure 6 - Before the method call.

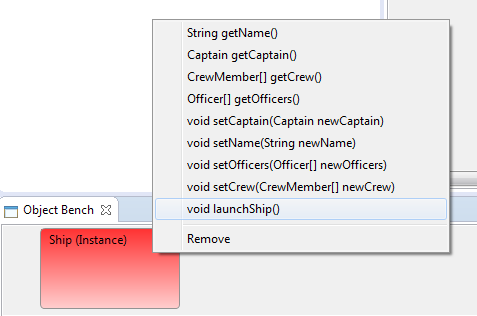


Figure 7 - Calling the method

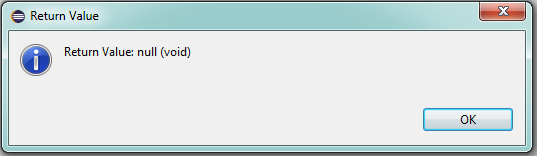


Figure 8 - Example return dialog.

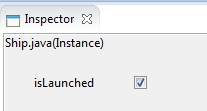


Figure 9 - Value after method call.

### Test 4

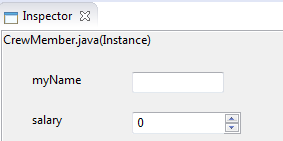


Figure 10 - Before method call.

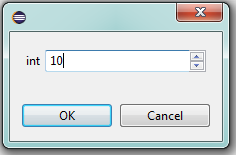


Figure 11 - Taking a parameter.

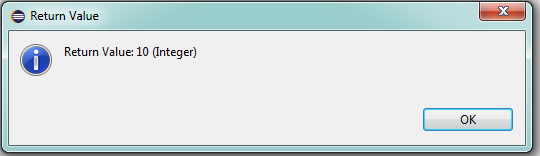


Figure 12 - The return value of the method.

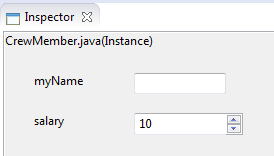


Figure 13 - After method call, value is changed as expected.

### Test 5

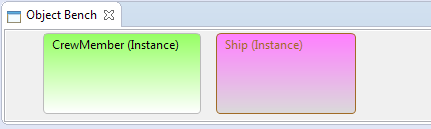


Figure 14 - Selected instance in pink.

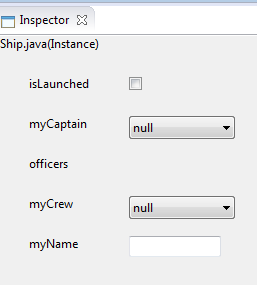


Figure 15 - The values of the instance in the inspector.

### Test 7

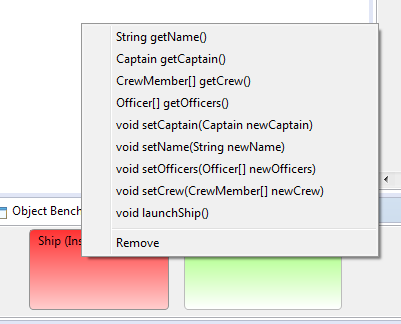


Figure 16 - Showing the available methods in a context menu.

## Class Diagram

### Test 1

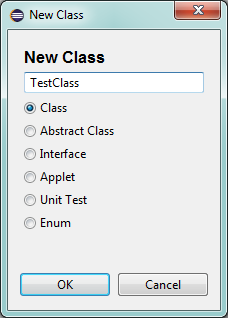


Figure 17 - Creating a new Class

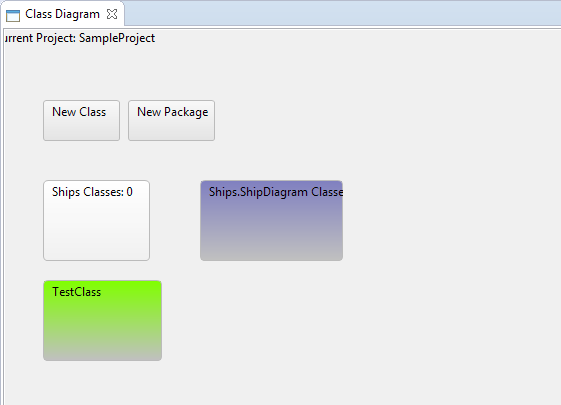


Figure 18 - The class is added to the diagram.

### Test 2

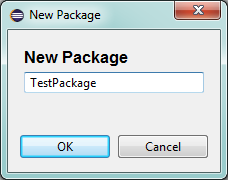


Figure 19 - Creating the test package.

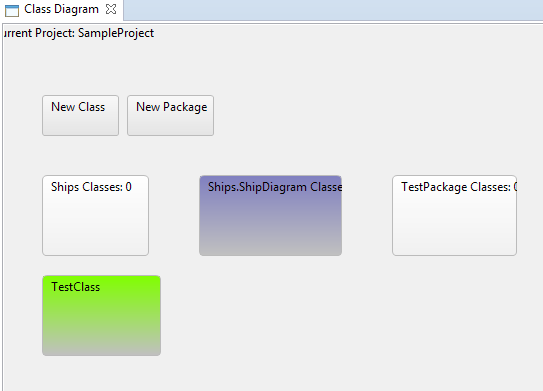


Figure 20 - The package is added.

### Test 3

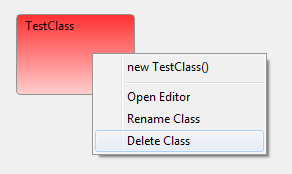


Figure 21 - Preparing to delete class.

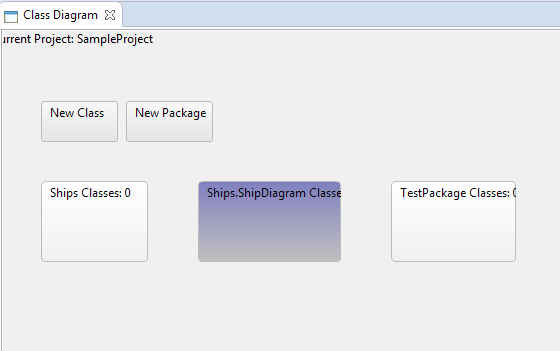


Figure 22 - After class deletion.

### Test 4

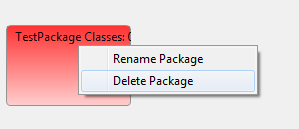


Figure 23 - Preparing to delete package.

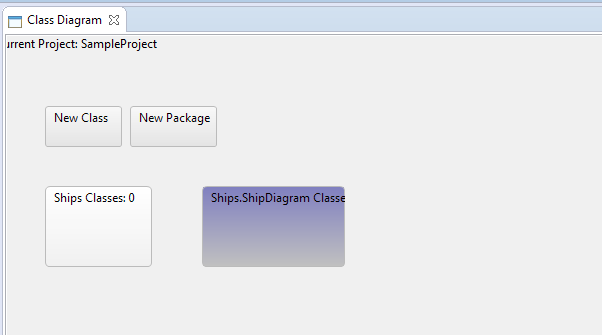


Figure 24 - Class diagram after package deleted.

### Test 5

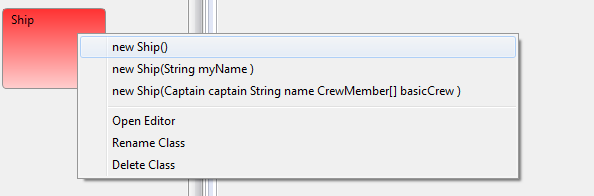


Figure 25 - Calling the constructor.

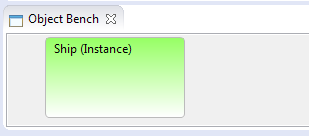


Figure 26 - Instance added to bench.

### Test 6

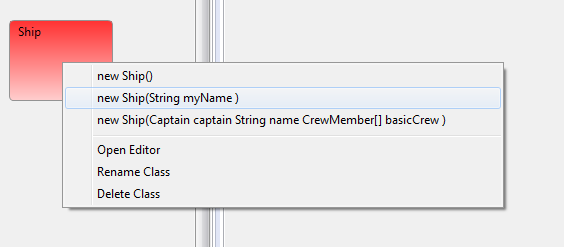


Figure 27 - Selecting the constructor.

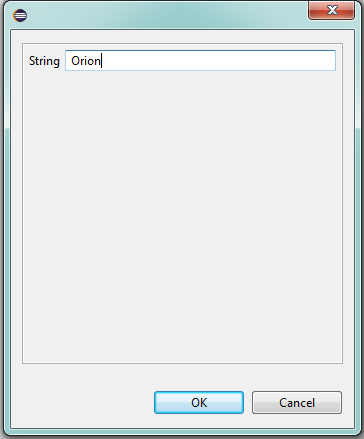


Figure 28 - Entering the parameter.

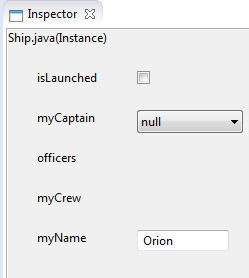


Figure 29 - The instance is created and the parameter is used.

### Test 7

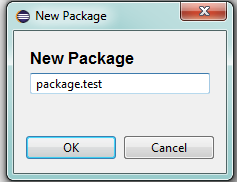


Figure 30 - Entering package name.

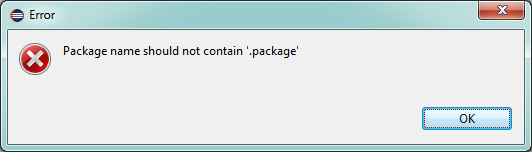


Figure 31 - Package name rejected as expected.

### Test 8

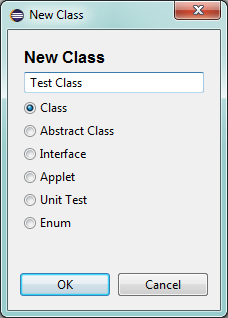


Figure 32 - Entering the class name.

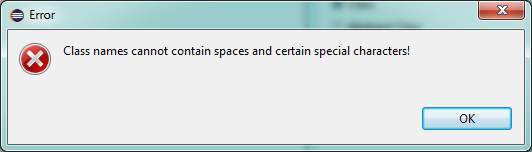


Figure 33 - The class name is rejected as expected.

### Test 9

### Test 10

### Test 11

### Test 12

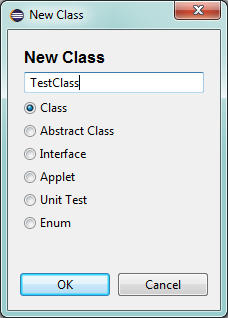


Figure 34 - Entering a class for creation, this is done twice.

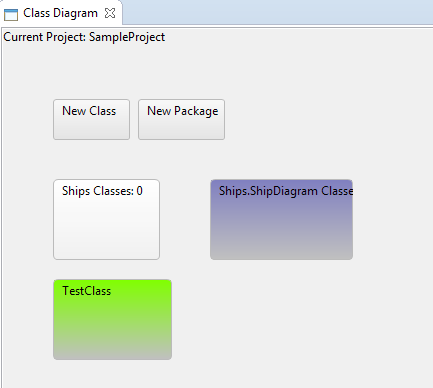


Figure 35 - There is no error and the system seems to accept a duplicate class.

### Test 13

### Test 14

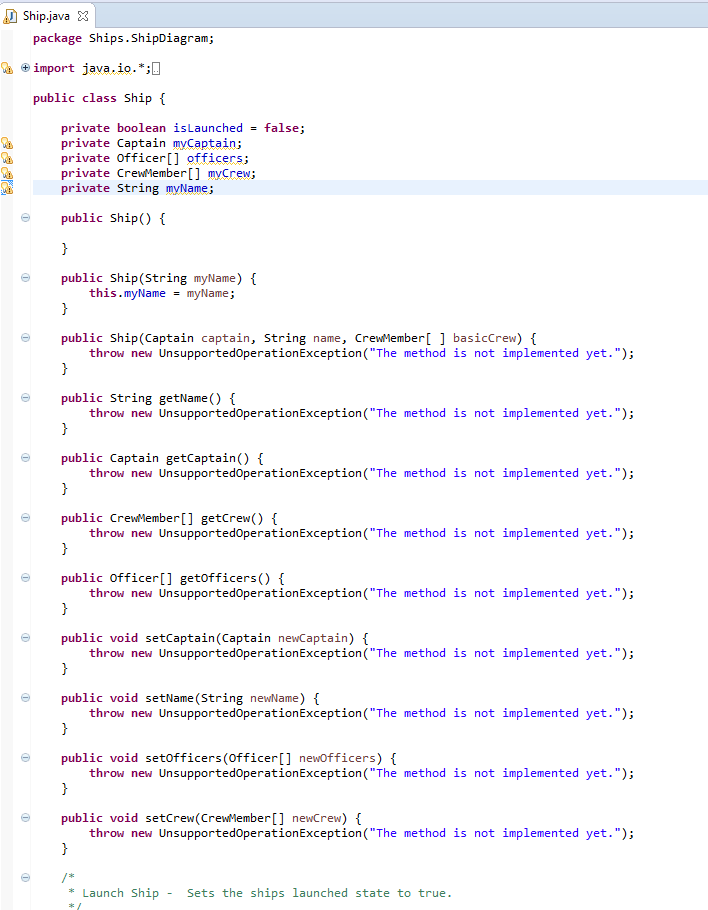


Figure 36 - The editor is opened as expected.

### Test 15

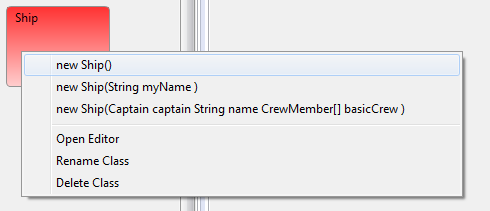


Figure 37 - All "Ship" constructors are present as expected.

## Inpsector

### Test 1

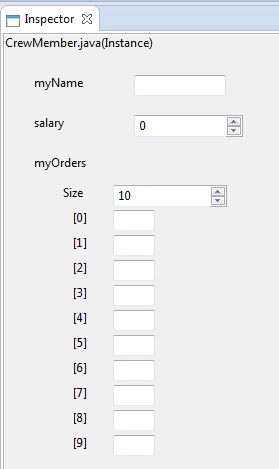


Figure 38 - An instance viewable in the inspector.

### Test 2

http://i.gyazo.com/9a4d887f7b0f64dd6ca8d9d4e5b47175.png

Figure 39 - The field types.

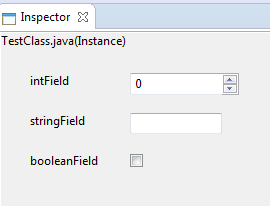


Figure 40 - As expected a Spinner, TextField and CheckBox are created.

### Test 3

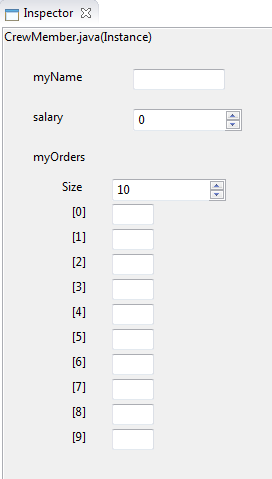


Figure 41 - The instance before editing the array.

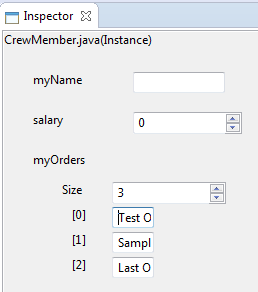


Figure 42 - After editing the array, field controls need to be made bigger.

### Test 4

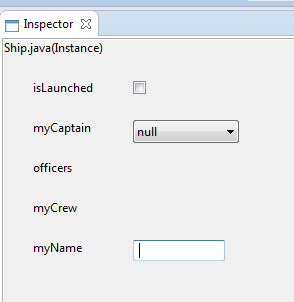


Figure 43 - Before naming the ship.

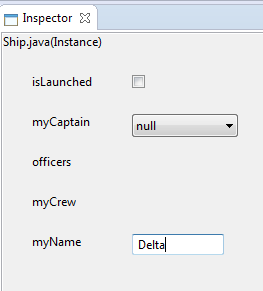


Figure 44 - After naming the ship.

### Test 5

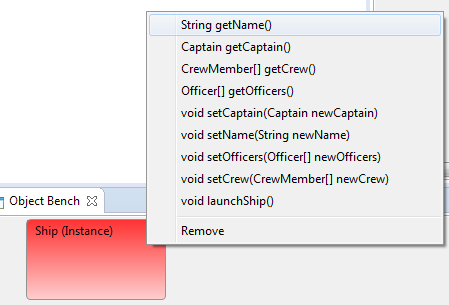


Figure 45 - Selecting the getName method.

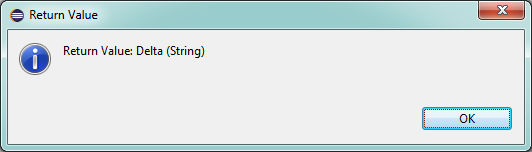


Figure 46 - Getting the correct return value as expected.

### Test 6

http://i.gyazo.com/3deccb89575075c6812a2db8e6cd4deb.png

Figure 47 - Field before setting the value.

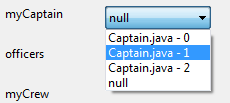


Figure 48 - Selecting the captain.

http://i.gyazo.com/e8e0b2664f6bb942537276536c6ea81a.png

Figure 49 - After setting the captain value. This is the second instance as wanted.

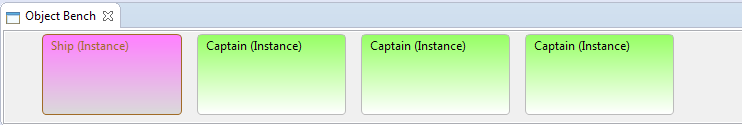


Figure 50 - Showing the object bench.

## COnstructor Dialog

### Test 1

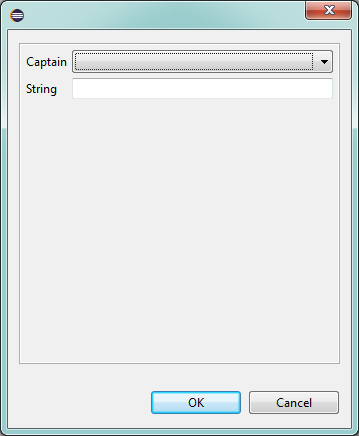


Figure 51 - The correct controls are provided as expected.

### Test 2

No screenshot, but the test was successful.

### Test 3

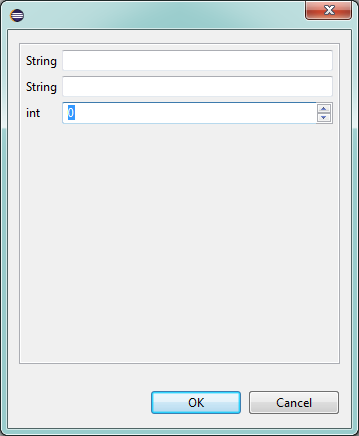


Figure 52 - The field won't allow text entry.

### Test 4

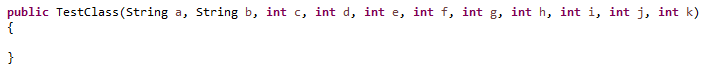


Figure 53 - The constructor signature.

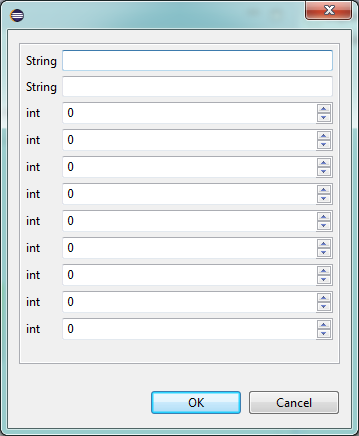


Figure 54 - The correct controls for the long constructor.

### Test 5

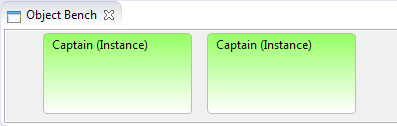


Figure 55 - The object bench containing captains.

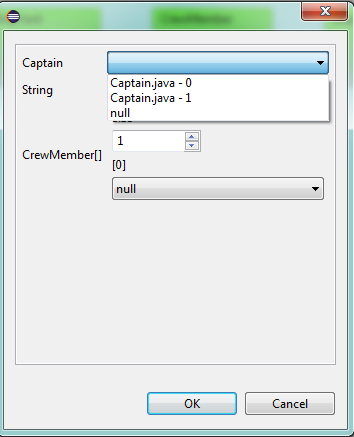


Figure 56 - The captains selectable from the constructor dialog.

### Test 6

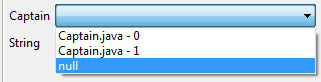


Figure 57 - An instance field can be set to null.

## Method Call Dialog

### Test 1

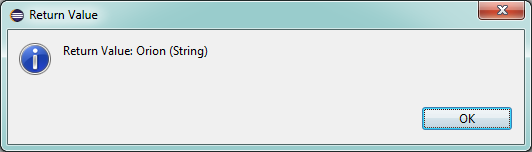


Figure 58 - Getting the return value from a called method.

### Test 2

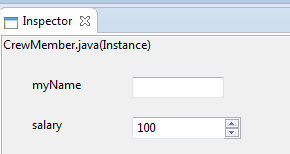


Figure 59 - Showing initial salary.

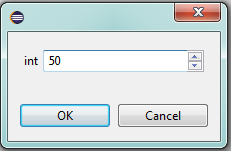


Figure 60 - Adding to the salary by setting the parameter in the method call dialog.

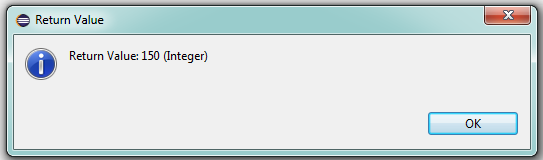


Figure 61 - The return value of the method when called after confirming the method call dialog.

### Test 3

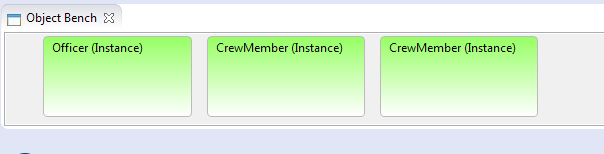


Figure 62 - The Object bench prior.

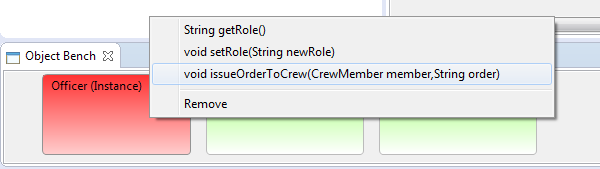


Figure 63 - Preparing to call the method.

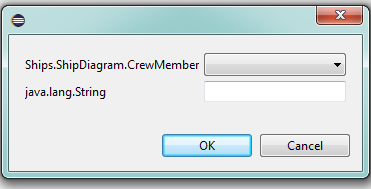


Figure 64 - As expected the controls are a drop down for crew members and a text box for the order string.

### Test 4

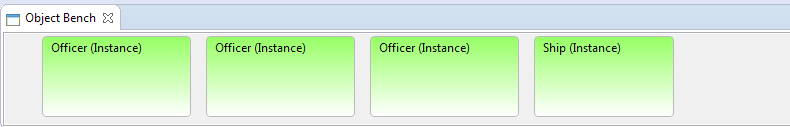


Figure 65 - Setting up the object bench.

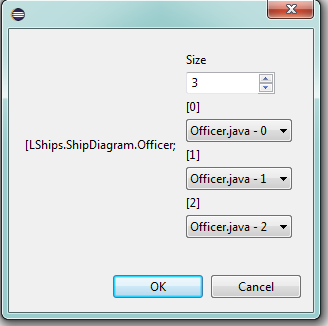


Figure 66 - Setting the fields.

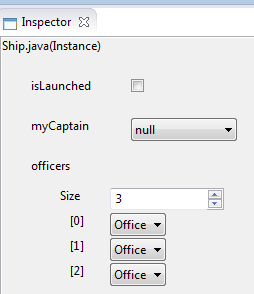


Figure 67 - To show the array is set according to the fields entered in the method dialog.

## New Class Dialog

### Test 1

See Class Diagram Test 1.

### Test 2

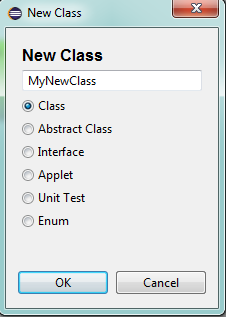


Figure 68 - Selecting templates.

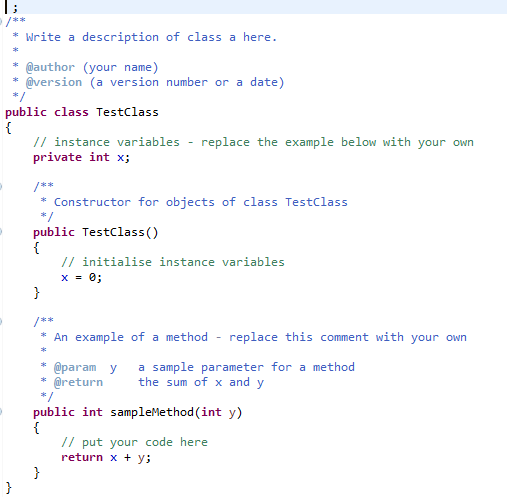


Figure 69 - Default template.

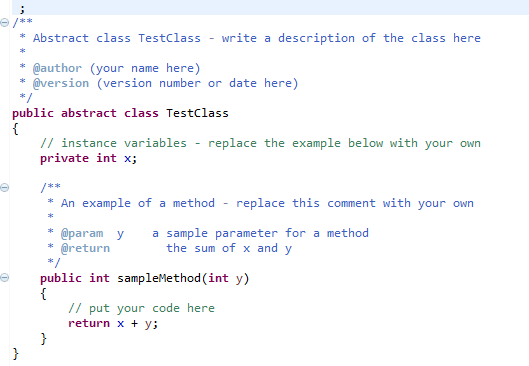


Figure 70 - Abstract template.

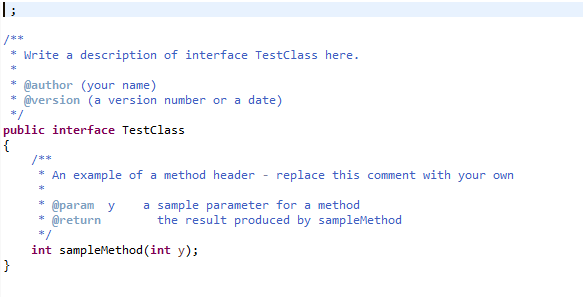


Figure 71 - Interface template.

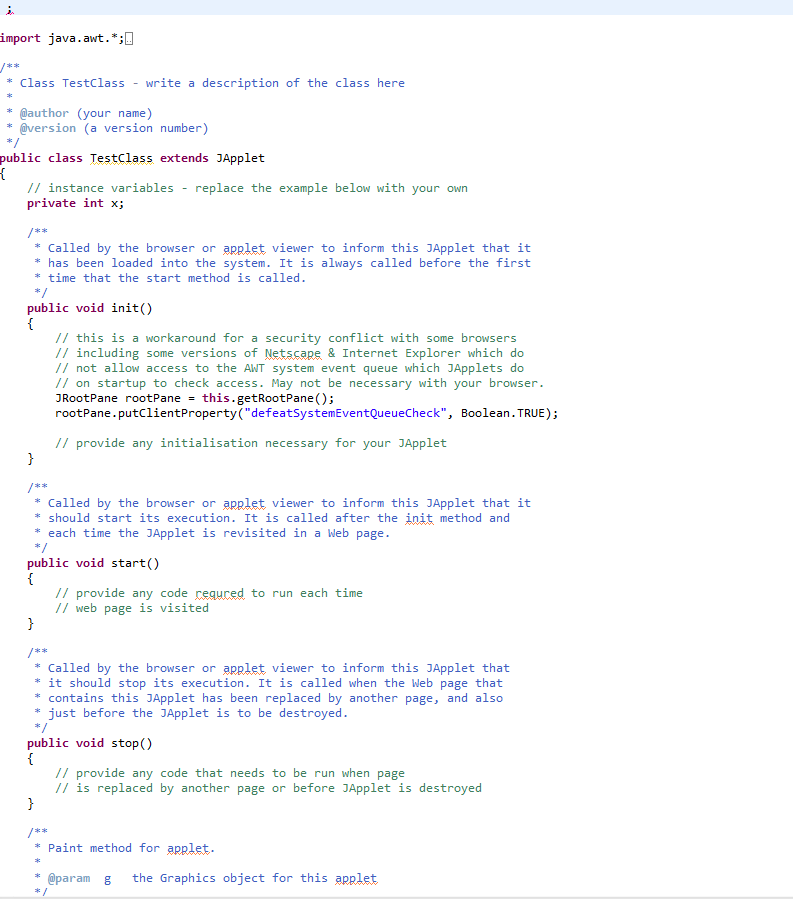


Figure 72 - Applet template.

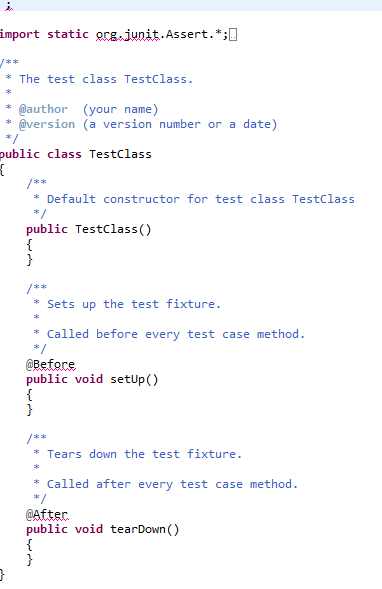


Figure 73 - Unit test template.

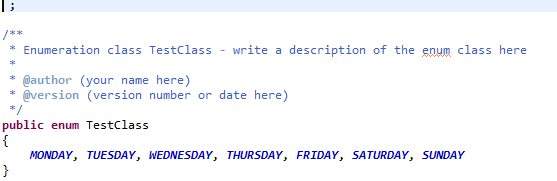


Figure 74 - Enumeration template.

### Test 3

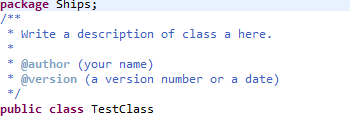


Figure 75 - Both package and class name are correct.

## New Package Dialog

### Test 1

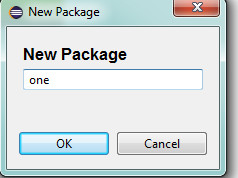


Figure 76 - Package dialog for a new package.

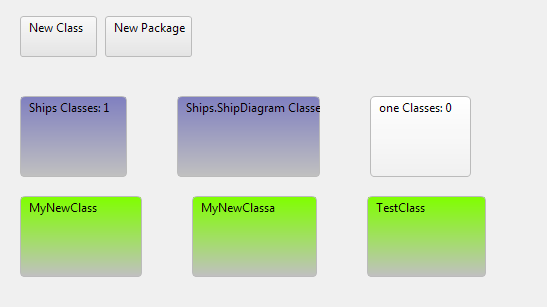


Figure 77 - The new package "one" is created.

### Test 2

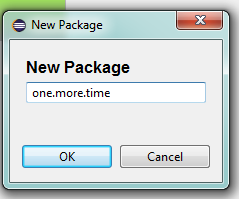


Figure 78 - Creating a nested package with the new package dialog.



Figure 79 -The new packages are added.

### Test 3

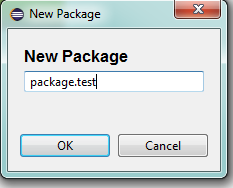


Figure 80 - New package dialog testing if package can be included in the name.

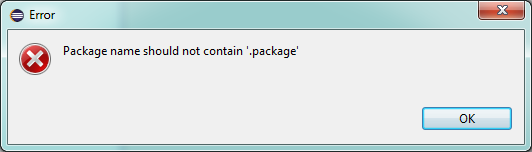


Figure 81 - As expected an error dialog is created.