Lab 5 – Test Execution using Loader Manager

Manual testing can be painstakingly dull and error prone activities. In this lab, student will be exposed to a unit test execution tool based on Java. The tool called LoaderManager is mere research prototype developed as part of the government eScience fund.

In a nut shell, LoaderManager takes the input test cases in a text file, called fault file. A fault file is a special file for defining the methods input parameters in a special format defined by a specialized markup language. The markup language works in similar fashion as the Hypertext Markup Language (HTML). A number of keywords are defined that can be understood by the LoaderManager parser for test case specification and execution (see Figure below). The description of each of this keyword (see Tables below).

@FaultFile

/////////////////////////////////////////

Common Header Definition

/////////////////////////////////////////

**className :** CollAccept

**methodName :** testAcceptance

**specifier** : public

**paramTypes :** 5

**returnType :** void

**parameter** : partypes [0]=Double.TYPE

**parameter :** partypes [1]=Double.TYPE

**parameter :** partypes [2]=Double.TYPE

**parameter** : partypes [3]=Double.TYPE

**parameter :** partypes [4]=Double.TYPE

/////////////////////////////////////////

Body - Test case 0

/////////////////////////////////////////

**arglist:arglist [0]=**new Double(49)

**arglist:arglist [1]=**new Double(49)

**arglist:arglist [2]=**new Double(49)

**arglist:arglist [3]=**new Double(49)

**arglist:arglist [4]=**new Double(49)

/////////////////////////////////////////

Body - Test case 1

/////////////////////////////////////////

**arglist:arglist [0]=**new Double(74)

.......

Sample Keywords Definition in a Fault File Keywords Description

|  |  |
| --- | --- |
| **Keywords** | **Description** |
| **className** | Using this keyword, the class name to be tested can be specified |
| **methodName** | Here, the method name to be tested can be specified using methodName keyword. |
| **Specifier** | This specifier keyword defines the type of method to be tested (i.e. private, public, and protected) |
| **paramTypes** | paramTypes keyword is used to define the number of parameters. |
| **returnType** | As the name suggest, return type permit the specification of the return call. |
| **Parameter** | Parameter keyword allows the definition of the data type according to the defined paramTypes. |
| **arglist:arglist [n]** | With this keyword, each individual argument value can be specified accordingly. |

The formatting for each individual argument value as part of the **arglist: arglist [n]** definition depends on the data type use (i.e. Basic Data Types, Array of Basic Data Types, Class, and Array of Class). Details of the definition are summarized in the following tables along with sample examples.

Specifying Input with Basic Data Types

|  |  |  |
| --- | --- | --- |
| **Basic Data Types** | **Format of Writing** | **Examples** |
| Integer | Integer.TYPE | new Integer (100) |
| Character | Character.TYPE | new Character(‘A’) |
| Double | Double.TYPE | new Double (1.4) |
| Long | Long.TYPE | new Long (23345555) |

Specifying Input with Array of Basic Data Types

|  |  |  |
| --- | --- | --- |
| **Array of Basic Data Types** | **Format of Writing** | **Examples** |
| Integer | int [].class | new int [] {200,100,10} |
| Character | char [].class | new char [] {‘A’,’B’,C’} |
| Double | double [].class | new double [] {2.1,1.2,2.5} |
| Long | long [].class | new long []{2335,1234, 111} |

Specifying Input with Class

|  |  |  |
| --- | --- | --- |
| **Class** | **Format of Writing** | **Examples** |
| Class String | String.class | String (“USM”) |
| Class Integer | Integer.class | new Integer(10) |
| Class User\_Defined | User\_Defined.class | new User\_Defined ( ) |

Specifying Input with Array of Class

|  |  |  |
| --- | --- | --- |
| **Array of Class** | **Format of Writing** | **Examples** |
| Class String | String [].class | String [] (“UMP”,”KTN”) |
| Class Integer | Integer [].class | Integer [] {10,34} |
| Class Character | Character [].class | Character [] {‘k’,’z’,’z’} |
| Class User\_Defined | User\_Defined [].class | User\_Defined []{new class1( ), new class2( )… new classN( )} |

Revisiting the quadratic equation with the developed test oracle only on equivalence partitioning (see Lab 2), write the appropriate fault file to automated the unit testing process.