**Test Plan**

Team LLL - Zhimin Lin, YaLing Lian, Linjian Chen

Class Name: EdgeFieldTest.java

|  |  |  |  |
| --- | --- | --- | --- |
| Test Method | Test plan | Expect | Got |
| testGetNumFigure() | EdgeField is the parameterized constructor, and the numFigure equal to 1, which is the value from the parameterized constructor. Therefore, the expected integer will be 1 and the actual integer will be the value from getNumFigure() method. Use the assertEquals() to compare whether the expected and the actual are equal. | 1 | 1 |
| testGetName() | EdgeField is the parameterized constructor, and the string of name is ‘test’ and it’s value comes from the parameterized constructor. The expected String will be ‘test’ and the getName() method will return the actual string. Use the assertEquals() to compare whether the expected and the actual are equal. | test | test |
| testGetTableId() | The default value of tableID is 0, therefore, the getTableId() method will return the actual integer - 0. Set the expected integer is 0. Use the assertEquals() method to compare the expected integer and the actual integer. | 0 | 0 |
| testSetTableId() | Call the setTableID(int value) method and set the int value equal to 2. And then the getTableID() method will return 2 as the actual integer. Set the expected integer equal to 2. Use the assertEquals() method to compare the expected integer and the actual integer. | 2 | 2 |
| testGetTableBound() | The default value of tableBound is 0, therefore, the getTableBound() method will return 0 as the actual integer. Set the expected integer is 0. Use the assertEquals() method to compare the expected integer and the actual integer. | 0 | 0 |
| testSetTableBound() | Call the setTableBound(int value) method and set the int value equal to 3. And then the getTableBound() method will return 3 as the actual integer. Set the expected integer equal to 3. Use the assertEquals() method to compare the expected integer and the actual integer. | 3 | 3 |
| testGetFieldBound() | The default value of fieldBound is 0, therefore, the getFieldBound() method will return 0. Set the expected integer is 0. Use the assertEquals() method to compare the expected integer and the actual integer. | 0 | 0 |
| testSetFieldBound() | Call the setFieldBound(int value) method and set the int value equal to 4. And then the getFieldBound() method will return the actual integer - 4. Set the expected integer equal to 4. Use the assertEquals() method to compare the expected integer and the actual. | 4 | 4 |
| testGetDisallowNull() | The default value of disallowNull is false, therefore, the getDisallowNull() method will return false as the actual boolean. Set the expected boolean is false. Use the assertEquals() method to compare the expected boolean and the actual boolean. | false | false |
| testSetDisallowNull() | Call the setDisallowNull(boolean value) method and set the boolean value equal to true. And then the getDisallowNull() method will return true as the actual boolean. Set the expected boolean equal to true. Use the assertEquals() method to compare the expected boolean and the actual boolean. | true | true |
| testGetIsPrimaryKey() | The default value of isPrimaryKey is false, therefore, the getIsPrimaryKey() method will return false as the actual boolean. Set the expected boolean is false. Use the assertEquals() method to compare the expected boolean and the actual boolean. | false | false |
| testSetIsPrimaryKey() | Call the setIsPrimaryKey(boolean value) method and set the boolean value equal to true. And then the getIsPrimaryKey() method will return true as the actual boolean. Set the expected boolean equal to true. Use the assertEquals() method to compare the expected boolean and the actual boolean. | true | true |
| testGetDefaultValue() | The default value of defaultValue is “”, therefore, the getDefaultValue() method will return “” as the actual String. Set the expected String is “”. Use the assertEquals() method to compare the expected String and the actual String. | “” | “” |
| testSetDefaultValue() | Call the setDefaultValue(String value) method and set the String value equal to ‘LLL’. And then the getDefaultValue() method will return ‘LLL’ as the actual String. Set the expected String equal to ‘LLL’. Use the assertEquals() method to compare the expected String and the actual String. | LLL | LLL |
| testGetVarcharValue() | The default value of varcharValue is 0, therefore, the getVarcharValue() method will return 0. Set the expected integer is 0. Use the assertEquals() method to compare the expected integer and the actual integer. | 1 | 1 |
| testSetVarcharValue() | Call the setVarcharValue(int value) method and set the int value equal to 2. And then the getVarcharValue() method will return 2 as the actual integer . Set the expected integer equal to 2. Use the assertEquals() method to compare the expected integer and the actual. | 2 | 2 |
| testGetDataType() | The default value of DataType is 0, therefore, the getDataType() method will return 0 as the actual integer. Set the expected integer is 0. Use the assertEquals() method to compare the expected integer and the actual integer. | 0 | 0 |
| testSetDataType() | Call the setDataType(int value) method and set the int value equal to 1. And then the getDataType() method will return 1 as the actual integer . Set the expected integer equal to 1. Use the assertEquals() method to compare the expected integer and the actual. | 1 | 1 |
| testGetStrDataType() | Create a String array name type. The values in the type have "Varchar", "Boolean", "Integer", "Double". The type as the expected and the getStrDataType() method as the actual. Use the assertArrayEquals() method to compare the expected array and the actual array. | {"Varchar", "Boolean", "Integer", "Double"} | {"Varchar", "Boolean", "Integer", "Double"} |
| testToString() | Create a new toString to return the value from EdgeField class. The new toString as the expected and the toString() method as the actual. Use the assertEquals() method to compare the expected and the actual. | 1|test|0|0|0|0|1|false|false| | 1|test|0|0|0|0|1|false|false| |

Class Name: EdgeTableTest.java

|  |  |  |  |
| --- | --- | --- | --- |
| Test Method | Test plan | Expect | Got |
| testGetNumFigure() | EdgeTable is the parameterized constructor, it take a string value as the parameter, and delimit by “|”. Set the string as “1|TEST”, getNumFigure() method should return the first value(must be an integer) after delimiter by “|”, which should be integer 1. Use assertEquals() method to check is the return value is correct. | 1 | 1 |
| testGetName() | Set the string as “1|TEST”, getName() method should return the second value after delimiter by “|”, which should be “TEST”. Use assertEquals() method to check is the return value is correct. | TEST | TEST |
| testAddRelatedTable() | addRelatedTable() method take an integer parameter, it will add the integer to alRelatedTables arraylist. In this test method, I call the addRelatedTable() method and set the parameter as an integer 2, then I call the makeArrays() method, which will convert the alRelatedTables arraylist to relatedTables array. After it converted, relatedTables[0] should be integer 2. Use assertEquals() method to check is that correct, use getRelatedTablesArray() method to find the relatedTables[0].  \*\*\* If the test pass, means addRelatedTable() method, makeArray() method and getRelatedTableArray() method are all working correctly. | 2 | 2 |
| testAddNativeField() | addNativeField() method take an integer parameter, it will add the integer to alNativeField arraylist. In this test method, I call the addNativeField() method four times and set the parameter as integer 4,0,0,0, then I call the makeArrays() method, which will convert the alNativeField arraylist to nativeFields array. After it converted, nativeFields [0] should be integer 4, and nativeFields[1],[2],[3] should be integer 0. Use assertEquals() method to check is that correct, use getNativeFieldsArray() method to find the items in nativeFields array.  \*\*\* If the test pass, means addNativeField() method, makeArray() method and getNativeFieldsArray() method are all working correctly. | 4  0  0  0 | 4  0  0  0 |
| testSetRelatedField() | addRelatedField() method take two integer parameters, it will add the second integer parameters to the relatedFields array in a specially index(first integer parameter). In this test method, I call setRelatedField() method twice, and set the parameter as (0,3) and (1,5). So now, relatedFields[0] should have integer 3 and relatedFields[1] should have 5. Use assertEquals() method to check is that correct. | 3  5 | 3  5 |
| testMoveFieldUp() | moveFieldUp() method take an integer parameter, this method will move the field(integer parameter) closer to the beginning of the list. In this test method, I call setRelatedField() method to set the second index as integer 9, then I call moveFieldUp() method, set the integer parameter to 2. So now, this method will move the second index to first index, so the relatedFields[1] should be 9. Use assertEquals() method to check is that correct. | 9 | 9 |
| testMoveFieldDown() | moveFieldDown() method take an integer parameter, this method will move the field(integer parameter) closer to the end of the list. In this test method, I call setRelatedField() method to set the second index as integer 8, then I call moveFieldDown() method, set the integer parameter to 2. So now, this method will move the second index to third index, so the relatedFields[3] should be 8. Use assertEquals() method to check is that correct. | 8 | 8 |
| testToString() | toString() method will return a string value which includes numFigure, name,nativeFields array, relatedTables array and relatedFields array with special format. Use assertEquals() method to check is that correct. | Table: 1  {  TableName: TEST NativeFields: 4|0|0|0  RelatedTables: 2  RelatedFields:3|9|0|8  } | Table: 1  {  TableName: TEST NativeFields: 4|0|0|0  RelatedTables: 2  RelatedFields:3|9|0|8  } |

Class: createDDLMySQLTest.java

|  |  |  |  |
| --- | --- | --- | --- |
| Test Method | Test Plan | Expect | Got |
| testCreateDDL() | Call createDDL() method from createDDLMySQL class | Working, No error | Working, No error |
| testGenerateDatabaseName() | Compare User input name to MySQLDB use method generateDatabaseName() from createDDLMySQL, then compare Get database name use method getDatabaseName() from createDDLMySQL class. Both Junit comapre method set default compare value: MySQL | Working, No error | Working, No error |
| testConvertStrBooleanToInt() | Test input value “true”, should return 1 back by use method convertStrBooleanToInt(“String input”) from createDDLMySQL class, the String input set as “true” and the method is boolean type. If integer return is 1 then that is correct, else if integer return is 0 then that is wrong. | Return 1 | Return 1 |
| testGetProductName() | useCompare the return product name by use getProductName() method from createDDLMySQL class and set expect producname is MySQL, if get product name is MySQL, then that is correct, else is wrong | Return “MySQL” | Return “MySQL” |
| testGetSQLString() | test getSQLString() method from createDLLMySQL, use assertNotNull, if GetSQLString is not null, then that is correct, else that wrong | Working, No error | Working, No error |

Class: EdgeConvertCreateDDLTest.java

|  |  |  |  |
| --- | --- | --- | --- |
| Test Method | Test plan | Expect | Got |
| testInitialize() | Because EdgeConvertCreateDDL is an abstract class, therefore， EdgeConvertCreateDDLTest class could extends EdgeConvertCreateDDL class. Create a new EdgeConvertCreateDDLTest name testObj, and call initialize() method in EdgeConvertCreateDDL class to test is work or not. | Working, No error | Working, No error |
| testGetTable() | First method assertNotNull to test getTable(“input integer”) method is not null, if the getTable method is not null, so that passing the test. Then use second method assertEquals method to compare the first table from already set up in CreateFieldsAndTables class and first table. Call CreateFieldsAndTables then use getEdgeTables() method to get array table. The array table set as EdgeTable[0] from EdgeConvertCreateDDL class. And call getTable method and set input is one to compare. If both output is same, then that is working, else is wrong. | First table in  Working/ output not null  EdgeTable:  Table: 1  {  TableName: table1  NativeFields: 2|3|4  RelatedTables: 2|3|4  RelatedFields: 0|2|0  } | First table in  Working/ output not null  EdgeTable:  Table: 1  {  TableName: table1  NativeFields: 2|3|4  RelatedTables: 2|3|4  RelatedFields: 0|2|0  } |
| testGetField() | First method assertNotNull to test getField(“input integer”) method is not null, if the getField() method is not null, so that passing the test. Then use second method assertEquals() method to compare the first field from already set up in CreateFieldsAndTables class and first field. Call CreateFieldsAndTables then use getEdgeFields() method to get array field. The array field set as EdgeField[0] from EdgeConvertCreateDDL class. And call getField() method and set input is one to compare. If both output is same, then that is working, else is wrong. | Working/ output not null  First Field in EdgeField:  1|oli|0|0|0|0|1|true|true| | Working/ output not null  First Field in EdgeField:  1|oli|0|0|0|0|1|true|true| |