Refactoring Report

Name: Ruohuan Xu

1. Mockito Tech

Firstly, I set the Dependency for the using of the Mockito in the project. Add the Xml code into the Attribute file.

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>1.9.5</version>
    <scope>test</scope>
</dependency>
```

Then import the Mockito jar file into the project. Import the Mockito to each file that has the test class.

```
import static org.mockito.Mockito.*;
import static org.junit.Assert.*;
```

After that, I used the Mockito to mock the instances of the parent production class. Then added the functions in the mock class to replace the original methods in the test class. Since I created the functions to replace the methods in the original class, I had to verify the output of these functions to make sure that these functions will display the right results. Finally, using the Assert method to check the output was right.

2. Test Class

(1) TestMapEvent

```
public void MockTestMapEvent() {

        private MapEvent MockMapEvent = mock(MapEvent);

        public MockMapEvent(Object source) {
            super(source);
        }
        public MockMapEvent(Object source, String oldName) {
            super(source, oldName);
        }

        when(MockMapEvent.getNewName()).thenReturn("");

        verify(MockMapEvent,time(1)).getNewName();
```

```
            Assert.assertEquals("",MockMapEvent.getNewName());

    }
```

(2) MyCayenneEvent

```
  public void MockTestMapEvent() {

            private MapEvent MockMapEvent = mock(MapEvent);

            public MockMapEvent(Object source) {
                  super(source);
            }
            public MockMapEvent(Object source, String oldName) {
                  super(source, oldName);
            }

            when(MockMapEvent.getNewName()).thenReturn("");

            verify(MockMapEvent,time(1)).getNewName();

            Assert.assertEquals("",MockMapEvent.getNewName());

    }
```

(3) MapEventFixture

```
      public void MockMapEventFixture() {

            String newName;
            private MapEvent MockMapEvent = mock(MapEvent);

            public MockMapEvent(Object source, String newName) {
                     super(source);
                     this.newName = newName;
               }

               public    MockMapEvent(Object    source,    String    newName,    String
oldName) {
                     super(source, oldName);
                     this.newName = newName;
               }
```

```
                when(MockMapEvent.getNewName()).thenReturn(newName);

                verify(MockMapEvent,time(1)).getNewName();

            Assert.assertEquals(newName,MockMapEvent.getNewName());
        }
```

(4) TestBridge

```
    public void MockTestBridge() {

            CayenneEvent lastLocalEvent;
                int startupCalls;
                int shutdownCalls;

            private EventBridge mockBridge = mock(EventBridge.class);

            private mockBridge(EventSubject localSubject, String externalSubject) {
                    super(localSubject, externalSubject);
                }

            when(mockBridge.sendExternalEvent("event")).thenReturn(lastLocalEvent
= event);
            when(mockBridge.shutdownExternal()).thenReturn(shutdownCalls++);
            when(mockBridge.startupExternal()).thenReturn(startupCalls++);

            verify(mockBridge,time(1)).sendExternalEvent("event");
            verify(mockBridge,atLeastOnce()).shutdownExternal();
            verify(mockBridge,atLeastOnce()).startupExternal();

        }
```

(5) MockChannelListener

```
    public void MockChannelListener(){

            boolean graphChanged;
                boolean graphCommitted;
                boolean graphRolledBack;

            private          DataChannelListener          MockChannelListener          =
mock(DataChannelListener.class);
```

```
        when(MockChannelListener.graphChanged("event")).thenReturn(graphChang
ed = true);

        when(MockChannelListener.graphFlushed("event")).thenReturn(graphCommi
tted = true);

        when(MockChannelListener.graphRolledback("event")).thenReturn(graphRoll
edBack = true);

                verify(MockChannelListener,time(1)).graphChanged("event");
                verify(MockChannelListener,time(1)).graphFlushed("event");
                verify(MockChannelListener,time(1)).graphRolledback("event");

                assertEquals(MockChannelListener.graphChanged("event"),true);
                assertEquals(MockChannelListener.graphFlushed("event"),true);
                assertEquals(MockChannelListener.graphRolledback("event"),true);



            }
```

(6) MockMappingNamespace

```
       public class MappingNamespacetest {

        private MockMappingNamespace mapspace;

        public void setUp() {
                MockitoAnnotations.initMocks(this);
        }

        when(mapspace.getEmbeddable(anyString())).thenReturn(null);
        when(mapspace.getResult(anyString())).thenReturn(null);

        when(mapspace.addDbEntity(DbEntity())).thenReturn(dbEntities.put(entity.ge
tName(), DbEntity()));
        when(mapspace.addObjEntity(ObjEntity())).thenReturn(objEntities.put(entity.g
etName(), ObjEntity()));
        when(mapspace.addQueryDescriptor(QueryDescriptor())).thenReturn(queryD
escriptors.put(queryDescriptor.getName(), QueryDescriptor()));
        when(mapspace.addProcedure(Procedure())).thenReturn(procedures.put(pro
cedure.getName(), Procedure()));

        when(mapspace.getDbEntity(anyString())).thenReturn(dbEntities.get(anyStrin
g()));
```

```
        when(mapspace.getObjEntity(anyString())).thenReturn(objEntities.get(anyStri
ng()));
        when(mapspace.getProcedure(anyString())).thenReturn(procedures.get(anySt
ring()));
        when(mapspace.getQueryDescriptor(anyString())).thenReturn(queryDescripto
rs.get(anyString()));

        when(mapspace.getDbEntities()).thenReturn(dbEntities.values());
        when(mapspace.getObjEntities()).thenReturn(objEntities.values());
        when(mapspace.getProcedures()).thenReturn(procedures.values());
        when(mapspace.getQueryDescriptors()).thenReturn(queryDescriptors.values()
);
        when(mapspace.getEmbeddables()).thenReturn(null);

        when(mapspace.getInheritanceTree(anyString())).thenReturn(null);
        when(mapspace.getResults()).thenReturn(null);
        when(mapspace.getObjEntity(Class<>)).thenReturn(null);
        when(mapspace.getObjEntity(Persistent())).thenReturn(null);



    }
```