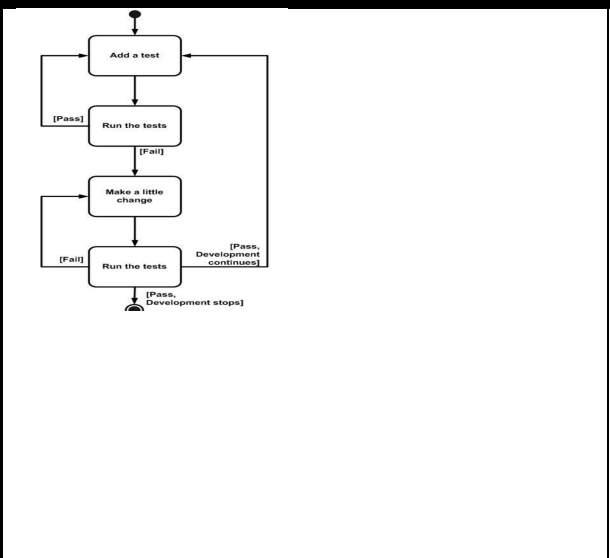
# Test Driven Development (TDD) Documentation

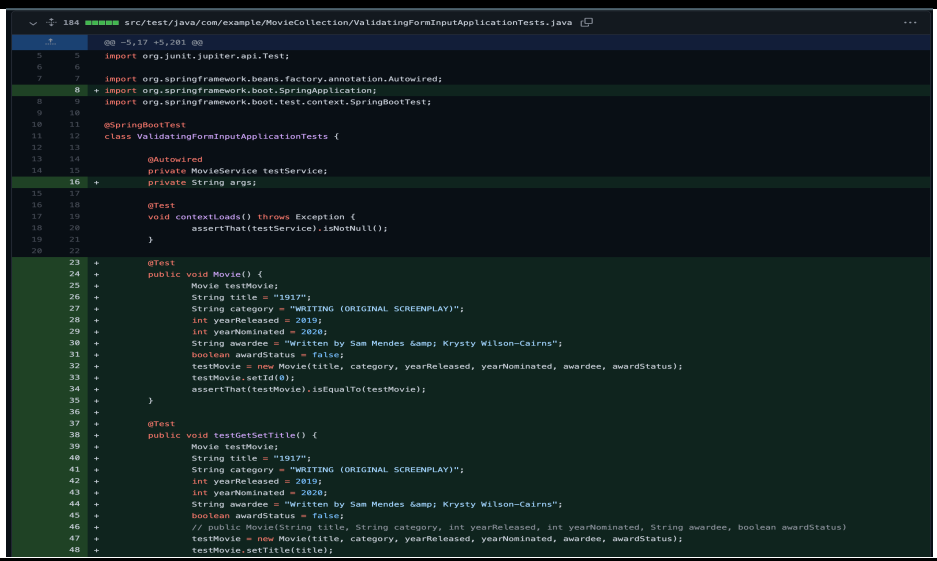
Figure 1's UML activity diagram depicts the processes of test first development (TFD).

The first step is to quickly add a test that just has enough code to fail. Following that, we run project tests, which are generally the whole test suite, but for the purpose of speed, we have chosen to run only a subset to ensure that the new test fails.

The project's functional code is then modified to ensure that it passes the new tests. The fourth step is to run your tests again. We had to update and retest our functional code if they failed. When the tests are successful, the process is restarted (you may first need to refactor any duplication out of your design as needed, turning TFD into TDD).

Unit Test Diagram and Code Performance:



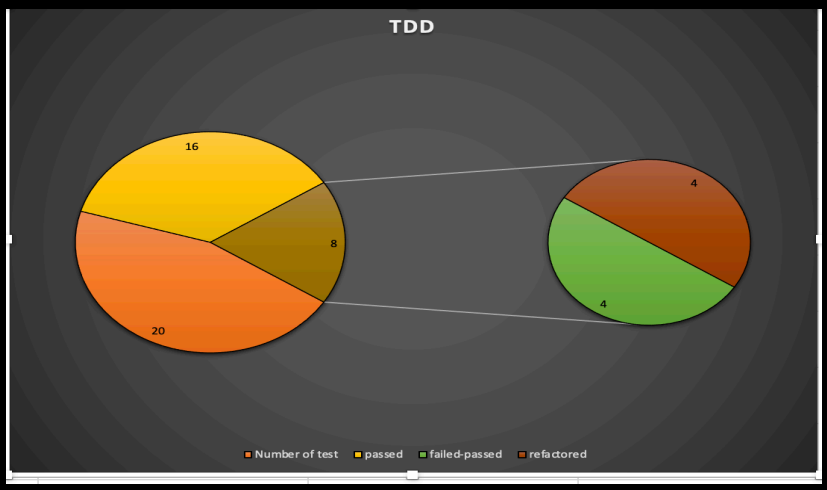



```
src/test/java/com/example/MovieCollection/ValidatingFormInputApplicationTests.java
@@ -5,17 +5,201 @@
import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;
+ import org.springframework.boot.SpringApplication;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ValidatingFormInputApplicationTests {

        @Autowired
        private MovieService testService;
+       private String args;

        @Test
        void contextLoads() throws Exception {
                assertThat(testService).isNotNull();
        }

+       @Test
+       public void Movie() {
+               Movie testMovie;
+               String title = "1917";
+               String category = "WRITING (ORIGINAL SCREENPLAY)";
+               int yearReleased = 2019;
+               int yearNominated = 2020;
+               String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+               boolean awardStatus = false;
+               testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+               testMovie.setId(0);
+               assertThat(testMovie).isEqualTo(testMovie);
+       }
+
+       @Test
+       public void testGetSetTitle() {
+               Movie testMovie;
+               String title = "1917";
+               String category = "WRITING (ORIGINAL SCREENPLAY)";
+               int yearReleased = 2019;
+               int yearNominated = 2020;
+               String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+               boolean awardStatus = false;
+               // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+               testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+               testMovie.setTitle(title);
```

```
+                assertThat(testMovie.getTitle()).contains(title);
+        }
+
+        @Test
+        public void testGetSetCategory() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setCategory(category);
+                assertThat(testMovie.getCategory()).contains(category);
+        }
+
+        @Test
+        public void testGetSetYearReleased() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setYearReleased(yearReleased);
+                assertThat(testMovie.getYearReleased()).isEqualTo(yearReleased);
+        }
+
+        @Test
+        public void testGetSetYearNominated() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setYearNominated(yearNominated);
+                assertThat(testMovie.getYearNominated()).isEqualTo(yearNominated);
+        }
+
```

```
+        @Test
+        public void testGetSetAwardee() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setAwardee(awardee);
+                assertThat(testMovie.getAwardee()).contains(awardee);
+        }
+
+        @Test
+        public void testGetSetIsAwardStatus() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setAwardStatus(awardStatus);
+                assertThat(testMovie.isAwardStatus()).isFalse();
+        }
+
+        @Test
+        public void testGetSetImageLink() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setImageLink(null);
+                assertThat(testMovie.getImageLink()).isNull();
+        }
+
```

```
+        @Test
+        public void testGetSetPlot() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setPlot(null);
+                assertThat(testMovie.getPlot()).isNull();
+        }
+
+        @Test
+        public void testGetSetTmdbId() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                testMovie.setTmdbId(0);
+                assertThat(testMovie.getTmdbId()).isEqualTo(0);
+        }
+
+        @Test
+        public void testToString() {
+                Movie testMovie;
+                String title = "1917";
+                String category = "WRITING (ORIGINAL SCREENPLAY)";
+                int yearReleased = 2019;
+                int yearNominated = 2020;
+                String awardee = "Written by Sam Mendes &amp;amp; Krysty Wilson-Cairns";
+                boolean awardStatus = false;
+                // public Movie(String title, String category, int yearReleased, int yearNominated, String awardee, boolean awardStatus)
+                testMovie = new Movie(title, category, yearReleased, yearNominated, awardee, awardStatus);
+                assertThat(testMovie.toString()).contains("1917");
+        }
+
+        @Test
+        public void main() {
+                MovieApplication.main(new String[] {});
+        }
```




# Data Analysis

| Name of Unit Test | Status of passed or fail | Refactoring |
|---|---|---|
| public void Movie () | Passed | - |
| public void testGetSetTitle() | Passed | - |
| public void testGetSetCategory() | Passed | - |
| public void testGetSetYearReleased() | Passed | - |
| public void testGetSetYearNominated() | Passed | - |
| public void testGetSetAwardee() | Passed | - |
| public void testGetSetIsAwardStatus() | Passed | - |
| public void testGetSetImageLink() | Passed | - |

| | | |
|---|---|---|
| public void testGetSetLink() | Fail-Passed | Refactored |
| public void testGetSetPlot() | Fail - Passed | Refactored |
| public void testGetSetTmdbId() | Passed | Refactored |
| public void testToString() | Passed | - |
| public void main() | Passed | - |
| | | |
| Number of Commits for Unit test and Validation | 5 | |