**CS4227 Assignment 1 – Interceptor**

Quinn Painter  
19234201

**Following Steps 1-7**

**1 - Model the Internal Behaviour of the Framework**

**Diagram

Description automatically generated**

Figure - State machine for the Movie System

**2 - Identify and Model Interception Points**

As seen in the state diagram, the movie rental code doesn’t have significant internal state transitions that would be subject to interception, so we’ll just use an externally visible state transition as an interception point – AddRental.

The interception will provide information about the rental addition for logging etc., so it only needs to be a Reader.

As there is only one interception point, there will only be one interception group.

**3 - Specify the Context Objects**

As there is only one interception point, there only needs to be one context object.

The AddRental interception will need information about the rental that was added, so the context object can have the following functions:

public string getRentalTitle()  
public double getRentalCharge()  
public string getCustomerName()

As these pieces of information change for each new rental that is added, the context object should be passed per-event.

**4 - Specify the Interceptors**

See figure 4.

**5 - Specify the Dispatchers**

See figure 7.

**6 - Implement the Callback Mechanisms in the Concrete Framework**

See figure 7.

**7 - Implement the Concrete Interceptors**

See figures 5 and 6.

**Sequence Diagram**

**A piece of paper with writing on it

Description automatically generated**

Figure - Sequence diagram of program execution

**Code**

Implementation is based on the Movie System example from *Refactoring* (Fowler et al., 2012)

Text

Description automatically generated

Figure - Add Rental function in the Customer class

Graphical user interface, text

Description automatically generated

Figure - Interceptor

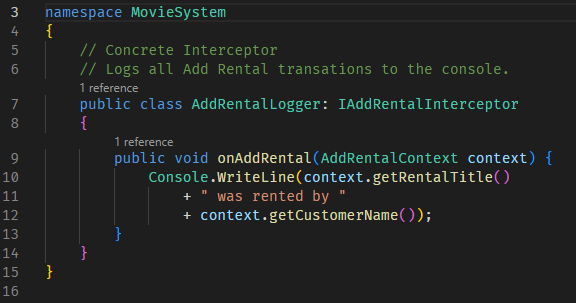


Figure - Concrete Interceptor 1

Text

Description automatically generated

Figure - Concrete Interceptor 2

Text

Description automatically generated

Figure - Dispatcher

Text

Description automatically generated

Figure - Context Object



Figure - Driver

All other code (Rental, Price, RegularPrice, NewReleasePrice, ChildrensPrice, rest of Customer) is unchanged from the Movie System example.

Git Repo: <https://github.com/QuinnPainter/CS4227-Assignment-1>

**Test Case**

**Text

Description automatically generated**

Figure - Test Case Code

**A screenshot of a computer

Description automatically generated with medium confidence**

Figure - Test case running and passing

**Code compiling and running**

**Text

Description automatically generated**

Figure - Running the code

**Evaluation**

The interceptor pattern is useful to add functionality to a framework without having to modify the framework itself. It can help to decouple concerns, and promote usability of interceptors. However, it has the downside that it can add significant complexity to a system and make it harder to maintain.

**References**

Fowler, M., Beck, K., Brant, J., Opdyke, W. and Roberts, D. (2012). *Refactoring*. Addison-Wesley.