# User-Stories

# User-Story 1:

Als „Software Quality Engineer“ möchte ich, dass sich das Entwicklungsteam im zu bearbeitenden Coding auskennt, um eine hohe Entwicklungsqualität und -geschwindigkeit zu gewährleisten.

## Tasks:

1. Zusammensetzen und Code gemeinsam analysieren und kommentieren.
2. Definition of Done (IST-Zustand) gem geklärt (Projektübergreifend)
3. Story-Points vergeben (Grooming)
4. Jedem Teammitglied ist klar, dass es darum geht, den Code zu verstehen
5. In Task(s) unterteilt -> z.B. Einrichten der Entwicklungsumgebung + erstes Öffnen des Projekts (jedem im Team) + gemeinsame Analyse / selbstständige Analyse etc.

# User-Story 2:

Als „Software Quality Engineer“ möchte ich, dass die switch-Anweisung zur Preisberechnung in eine eigene Methode ausgelagert wird, sodass hierdurch die Übersichtlichkeit und Wartbarkeit des Codes gesteigert wird.

## Tasks:

1. Amount-Calculation in eigene Methode auslagern (entsprechende Benennung)
2. Aufruf der neuen Methode in altem Code
3. Interne Variablen der extrahierten Methode umbenennen

# User-Story 3:

Als „Software Quality Engineer“ möchte ich, dass sich die Methode zur Preisberechnung in der hierzu geeigneten Klasse befindet, sodass die Kohäsion des Codes erhöht wird.

## Tasks:

1. Methode in Rental-Klasse verschieben
2. Methodenaufruf anpassen

# User-Story 4:

Als „Software Quality Engineer“ möchte ich, dass temporäre Variablen vermieden und die Methode zur Preisberechnung soweit möglich als Query aufgerufen wird, sodass die Code-Menge reduziert wird.

## Tasks:

1. Temporäre Variablen vermeiden
2. Methode zur Preisberechnung als Query aufrufen

# User-Story 5:

Als „Software Quality Engineer“ möchte ich, dass die Berechnung der frequentRenterPoints in die relevante Klasse ausgelagert (Kombination aus Extract und Move Method), um die Kohäsion des Codes zu erhöhen und die Kopplung zu verringern. Außerdem möchte man das Prinzip „Separation of Concerns“ einhalten.

## Tasks:

1. Methode in Rental-Klasse verschieben
2. Methodenaufruf anpassen

# User-Story 6:

Als „Software Quality Engineer“ möchte ich, dass die temp-Variablen „totalAmount“ und „frequentRenterPoints“ aus dem Code entfernt werden, da temporäre Variablen, die nur für diesen Code-Abschnitt benötigt werden, keinen Mehrwert bieten und die Lesefreundlichkeit erhöht wird.

## Tasks:

1. Temporäre Variablen entfernen

# User-Story 7:

Als „Software Quality Engineer” möchte ich, dass die obsolete Collection „Vector”, die in der Klasse „Customer” genutzt wird, durch eine nicht-obsolete Collection ersetzt wird, da eine obsolete Collection nicht mehr genutzt werden soll. Der Support könnte enden und im schlimmsten Fall das Programm nicht mehr funktionieren.

## Tasks:

1. Obsolete Collection durch eine Liste ersetzen

# User-Story 8:

Als „Software Quality Engineer” möchte ich, dass die unterschiedlichen Arten von Filmen nicht mehr durch eine Konstante erkennbar werden, sondern durch eine Vererbung. Hierzu soll das Pattern „Strategie” genutzt werden.

## Tasks:

1. Die Klasse Movie als abstrakte Klasse definieren
2. Die Unterklassen MovieChildren, MovieRegular und MovieNewRelease anlegen

# User-Story 9:

Als „Software Quality Engineer” möchte ich, dass das Benutzerhandbuch auf Vollständigkeit und

Korrektheit überprüft wird.

## Tasks:

1. Vollständigkeit und Korrektheit des Benutzerhandbuchs überprüfen und bei Bedarf anpassen

# User-Story 10:

Als „Software Quality Engineer” möchte ich, dass eine höhere Kohäsion erzeugt wird. Dazu wird die Statement-Erzeugung in eigene Klassen ausgelagert. Außerdem möchte man das Prinzip „Separation of Concerns“ einhalten.

Tasks:

1. Anlegen von Statement, TextStatement, HtmlStatement
2. Ausprogrammieren der Logik von den jeweiligen Unterklassen.
3. Die Methoden getTotalCharge, getRentals und getTotalFrequentRentalPoints in der Klasse Customer ausprogrammieren