CarProject - Dokumentation

# 1. Überblick

Das Projekt 'CarProject' simuliert ein Auto-Modell, das Geschwindigkeit und Gang berücksichtigt und zufällige Komponenten bei der Beschleunigung über ein Würfel-Interface einbezieht. Die Logik umfasst mehrere Klassen wie 'Car', 'DefaultDice', und 'Section' sowie zugehörige Unit-Tests, die das Verhalten des Autos überprüfen.

# 2. Klassenerklärungen

## 2.1 Klasse Car

Die Klasse 'Car' repräsentiert das Auto mit den folgenden Hauptmerkmalen:

* • \_gear: Ein privates Feld, das den aktuellen Gang des Autos speichert. Standard ist 0.
* • \_speed: Ein privates Feld, das die Geschwindigkeit des Autos speichert. Standard ist 0.
* • \_dice: Ein Würfelobjekt (vom Typ 'IDice'), das zufällige Werte liefert, um die Geschwindigkeit zu beeinflussen.

Die 'Accelerate' Methode führt einen Würfelwurf durch und berechnet die Geschwindigkeit basierend auf dem aktuellen Gang und den gewürfelten Augen. Die Gang-Eigenschaft hat eine Validierung, die sicherstellt, dass nur Werte zwischen 0 und 6 akzeptiert werden.

## 2.2 Interface IDice

Das 'IDice' Interface definiert die Struktur eines Würfels, der in diesem Projekt verwendet wird. Es enthält die Eigenschaften und Methoden, die für die Würfel-Implementierungen notwendig sind:

* • Dots: Eine Eigenschaft, die die Anzahl der gewürfelten Augen darstellt.
* • Roll(): Eine Methode, die den Würfelwurf durchführt und eine neue zufällige Zahl setzt.

## 2.3 Klasse DefaultDice

Die Klasse 'DefaultDice' implementiert das 'IDice' Interface und verwendet einen Zufallsgenerator, um eine Zahl zwischen 1 und 6 zu würfeln. Das Ergebnis wird in der 'Dots'-Eigenschaft gespeichert. Dies erlaubt es, den Würfelwurf zu kontrollieren und das Verhalten des Autos realistisch zu simulieren.

## 2.4 Klasse Section

Die Klasse 'Section' repräsentiert einen Straßenabschnitt, der durch die maximale erlaubte Geschwindigkeit ('MaxSpeed') und die Länge ('Length') des Abschnitts beschrieben wird. Dies erlaubt eine realistische Simulation verschiedener Straßenzustände und Einschränkungen.

# 3. Unit-Tests

Unit-Tests validieren das Verhalten der Klassen und Methoden und stellen sicher, dass die Anforderungen erfüllt werden. Die wichtigsten Tests sind wie folgt:

## 3.1 CarTests

Die Tests für die 'Car'-Klasse überprüfen:

* • Initialwerte wie Geschwindigkeit und Gang beim Erstellen eines Autos.
* • Die Validierung von Gängen zwischen 0 und 6 sowie die Ausnahmebehandlung bei ungültigen Werten.
* • Die Beschleunigungslogik durch Verwendung eines FakeDice, um die Geschwindigkeit auf Grundlage eines bekannten Wertes zu testen.

## 3.2 SectionTest

Dieser Test stellt sicher, dass die Klasse 'Section' die Attribute MaxSpeed und Length korrekt speichert und initialisiert.

# 4. Erweiterungsmöglichkeiten für Tests

Es gibt mehrere Möglichkeiten, die Tests weiter auszubauen und zusätzliche Szenarien zu prüfen:

* • Negative Tests für 'Accelerate', um ungültige Zustände wie negative Geschwindigkeit abzufangen.
* • Einhaltung der MaxSpeed-Einschränkungen in der Klasse 'Section', um zu testen, ob die Auto-Geschwindigkeit die Abschnittsgrenzen respektiert.
* • Überprüfung der Fehlermeldungen bei ungültigen Gangwerten, um sicherzustellen, dass die erwarteten Ausnahmen korrekt ausgelöst werden.

| **Schritt / Aspekt** | **Mit DefaultDice** | **Mit FakeDice** |
| --- | --- | --- |
| **Erstellen des Car-Objekts** | DefaultDice wird automatisch verwendet | FakeDice wird explizit übergeben |
| **Aufruf von Roll()** | Generiert eine zufällige Zahl für Dots | Setzt nur RollWasCalled auf true |
| **Berechnung von Speed** | Geschwindigkeit variiert, basierend auf Dots | Geschwindigkeit ist konstant (z. B. Dots = 2) |
| **Ergebnisse im Test** | Geschwindigkeit hängt von zufälligem Dots ab | Geschwindigkeit ist vorhersehbar, RollWasCalled überprüfbar |
| **Dots** | Zufälliger Wert zwischen 1 und 6 | Fester Wert für kontrollierte Tests |
| **Roll() Verhalten** | Generiert und setzt zufälligen Dots-Wert | Setzt RollWasCalled auf true, Dots bleibt unverändert |