Nutzerhandbuch

Einleitung:

AADMS ist eine Anwendung, die dazu dienen soll, den Nutzer an die testgetriebene Entwicklung von Programmen (TDD) heranzuführen.

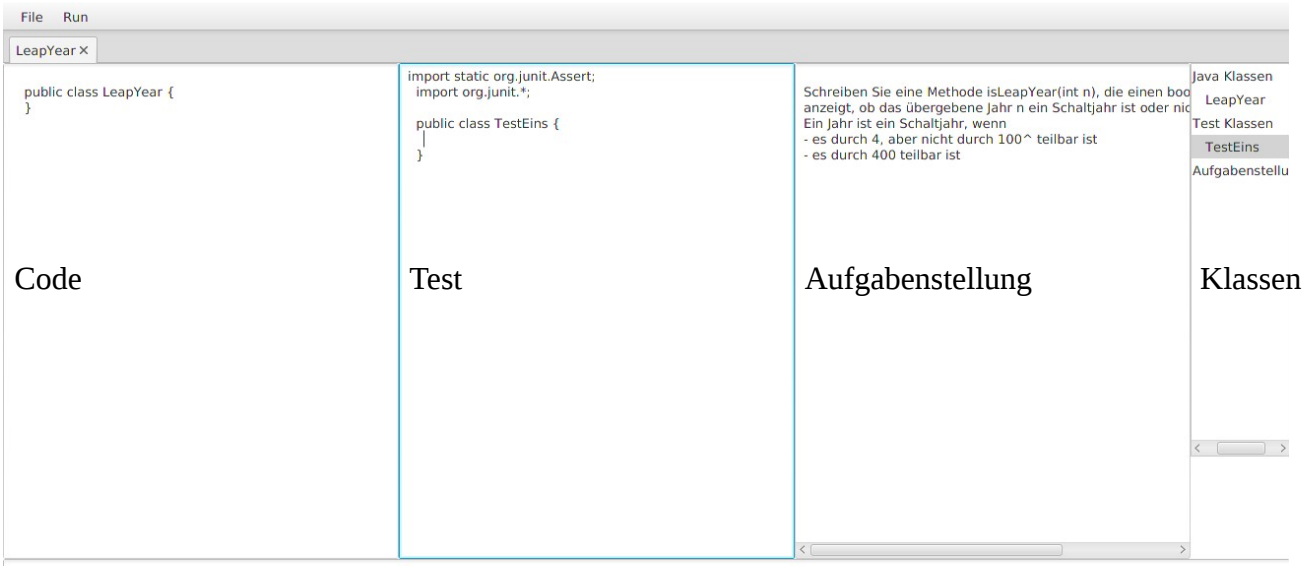
Dabei gibt es drei Phasen:

Phase RED: Schreiben eines fehlschlagenden Tests. (Test muss kompilieren)

Phase GREEN: Schreiben von Code, welcher den Test erfüllt. (Code muss kompilieren)

Phase REFACTOR: Hier darf der Nutzer seinen Code verbessern.

Oberfläche:



Schaltflächen:

File:

- *New File*: Öffnen von Aufgaben aus der Aufgabenbibliothek
- *refresh Projects*: lädt die Aufgabenbibliothek neu ein
- *zip Projec*t: erstellt ein Zip-Archiv des aktuellen Projekts

Run:

- *Phase RED: Compile Test*: prüft eingegebenen Test auf Fehlschlagen und Kompilierbarkeit.
- *Phase GREEN: Compile Test:* prüft eingegebenen Code auf Erfüllen des (der) Tests und Kompilierbarkeit.
- *Refactor Test:* Überprüft nach Überarbeiten des Codes erneut auf Erfüllen des (der) Tests und Kompilierbarkeit.

Tabs:

Es kann gleichzeitig an mehreren Aufgaben gearbeitet werden, zwischen denen der Nutzer über die Tabs wechseln kann.

Textfelder:

- *Code*: In Phase GREEN kann hier der Code der ausgewählten Klasse editiert werden.
- *Test:* In Phase RED kann hier der Test geschrieben werden.
- *Aufgabenstellung:* Hier findet sich die Aufgabenstellung der aktuell geöffneten Aufgabe
- *Konsole:* Hierüber werden Statusinformationen des Programms und Kompilierfehler an den Nutzer weitergeleitet.

Klassenfeld:

Dieses Auswahlfeld lässt den Nutzer die Klassen und Testklassen der aktuellen Aufgabe öffnen. Ein Wechsel der anzuzeigenden Klasse ist dabei immer möglich. Die Klassen werden dabei zwischengespeichert.

Programmablauf:

Zuerst muss eine Aufgabe geladen werden. Hierzu wird über die Schaltfläche *File → New File* die Auflistung der Aufgaben aufgerufen. Ist eine Aufgabe ausgewählt, werden die Aufgabenstellung, sowie die zugehörigen Klassen und Testklassen angezeigt. Die einzelnen Klassen können aufgerufen und je nach Phase bearbeitet werden. Zuerst befindet der Nutzer sich in Phase RED, es muss also ein fehlschlagender, kompilierender Test geschrieben werden. Durch Klicken auf *Run → Phase RED: Compile Test* prüft das Programm den eingegebenen Test auf diese Eigenschaften (die anderes Phasen sind dabei deaktiviert). War die Prüfung erfolgreich, befindet sich der Nutzer nun in

Phase GREEN. Nun muss eine Methode geschrieben werden, welche den zuvor fehlschlagenden Test erfüllt. Der Code muss auch hier kompilieren. Durch Klicken auf *Run → Phase GREEN: Compile Code* wird dies durch das Programm überprüft. War auch das erfolgreich befindet sich der Nutzer nun in der Phase REFACTOR. Der zuvor geschriebene Code darf nun verbessert werden. Über *Run → Refactor Test* wird erneut geprüft, ob der Code den Test erfüllt und kompiliert. Der Kreisprozess beginnt nun von vorne ausgehend vom Schreiben eines weiteren fehlschlagenden Tests (Phase RED).

Zusatzfunktionen:

Babysteps:

Je nach Aufgabe kann das Babysteps-Feature aktiv sein. Dabei gibt es ein Zeitlimit zur Erfüllung einer Phase durch den Nutzer. Läuft die Zeit ab, wird in der aktuellen Phase geschriebene Code gelöscht. Ziel dessen ist, dass der Nutzer eine kleinschrittige Programmierweise entwickelt.

Akzeptanztests:

Je nach Aufgabenstellung können Akzeptanztests vorhanden sein, die der Nutzer erfüllen muss und nicht löschen kann.