Softwaretechnische Grundlagen
Programmierpraktikum

Florian Dittrich
Sommersemester 2016

Projekt #7

TDDT (Test Driven Development Trainer)

package konfig

de.hhu.propra16.avaders.konfig

class WerteTabelle implements IWerteTabelle

Zuständigkeit

Diese Klasse verwaltet Name-Wert-Paare in einer `HashMap` mit WerteEintrag. Auf die einzelnen Einträge kann nicht von außerhalb zugegriffen werden, um die Datenintegrität zu schützen.

Verwendung zur Verwaltung der Statuswerte
Da im Programm hüfig Werte aus anderen Klassen benötigt werden, müssten entweder alle diese Klassen als Parameter übergeben oder die Werte bereits vorher gesammelt werden. Durch die Wertetabelle sollen alle relevanten Werte an zentraler Stelle gesammelt und Verfügbar gemacht werden.

Verwendung zur Einstellungsverwaltung
Durch die Wertetabelle es möglich, unabhängig von den Klassen, die die Konfigurationsdatei verarbeiten, neue Einstellungen hinzuzufügen oder zu entfernen. Einzig die verwendeten Datentypen müssen hinterlegt sein. Als Grundformat werden Zeichenketten verwendet, die dann entsprechend konvertiert werden. Schlägt dieser Vorgang fehl, wird eine Exception geworfen und die aufrufende Funktion kann einen Standardwert setzen.

Methoden

Methoden, die zum Interface `IWerteTabelle` gehören, sind mit * markiert.

`public KonfigWerte()`

Diese Methode erstelle eine neue, leere Wertetabelle

`public boolSetzen(String pName, boolean pWert)` *

Speichert eine Einrag mit dem übergebenen Namen und Wert in der Wertetabelle. Falls bereits ein Eintrag mit diesem Namen vorhanden ist, wird sein Wert ersetzt.

Parameter:

`pName`	Der eindeutige Name des Eintrags, über den er verwaltet wird
`pWert`	Ein Wahrheitswert, mit dem der neue Eintrag initialisiert wird

`public intSetzen(String pName, int pWert)` *

Speichert eine Einrag mit dem übergebenen Namen und Wert in der Wertetabelle. Falls bereits ein Eintrag mit diesem Namen vorhanden ist, wird sein Wert ersetzt.

Parameter:

`pName`	Der eindeutige Name des Eintrags, über den er verwaltet wird
`pWert`	Ein Integer-Wert, mit dem der neue Eintrag initialisiert wird

`public stringSetzen(String pName, String pWert)` *

Speichert eine Einrag mit dem übergebenen Namen und Wert in der Wertetabelle. Falls bereits ein Eintrag mit diesem Namen vorhanden ist, wird sein Wert ersetzt.

Parameter:

`pName` Der eindeutige Name des Eintrags, über den er verwaltet wird

`pWert` Ein String, mit dem der neue Eintrag initialisiert wird

`public boolean boolAbfragen(String pName) throws EintragNichtGefunden)` *

Sucht in der Wertetabelle nach einem Eintrag mit dem übergebenen Namen und gibt seinen Wert als Wahrheitswert zurück.

Parameter:

`pName` Der Name des gesuchten Eintrags

Exceptions:

Wirft `EintragNichtGefunden`, falls kein Eintrag mit dem übergebenen Namen in der Wertetabelle gefunden werden konnte.

`public int intAbfragen(String pName) throws EintragNichtGefunden, ParseException)` *

Sucht in der Wertetabelle nach einem Eintrag mit dem übergebenen Namen und gibt seinen Wert als Integer zurück.

Parameter:

`pName` Der Name des gesuchten Eintrags

Exceptions:

Wirft `EintragNichtGefunden`, falls kein Eintrag mit dem übergebenen Namen in der Wertetabelle gefunden werden konnte.

Wirft `PharserException`, falls der Wert nicht in einen Integer umgewandelt werden konnte

`public String stringAbfragen(String pName) throws EintragNichtGefunden)` *

Sucht in der Wertetabelle nach einem Eintrag mit dem übergebenen Namen und gibt seinen Wert als String zurück.

Parameter:

`pName` Der Name des gesuchten Eintrags

Exceptions:

Wirft `EintragNichtGefunden`, falls kein Eintrag mit dem übergebenen Namen in der Wertetabelle gefunden werden konnte.

Abhängigkeiten von anderen Klassen und Interfaces

- class WerteEintrag

`class WerteEintrag implements IWerteEintrag`

Zuständigkeit

Ein WerteEintrag speichert einen einzelnen Wert in seinem originalen @link Datentypen und konvertiert ihn falls nötig in anderen Datentypen. Der Wert bleibt jedoch immer als Original erhalten und wir aus diesem konvertiert. Instanzen von WerteEintrag werden in einer WerteTabelle mit einem Namen versehen gespeichert und verwalten dort einzelne Einstellungen oder Statuswerte des Programms.

Methoden

Methoden, die zum Interface `IWerteEintrag` gehören, sind mit * markiert.

`public WerteEintrag(boolean pWert)`

Erstellt einen neuen WerteEintrag und initialisiert ihn mit dem übergebenen Wahrheitswert

Parameter:

`pWert` Wahrheitswert, mit dem dieser WerteEintrag initialisiert wird

`public WerteEintrag(int pWert)`

Erstellt einen neuen WerteEintrag und initialisiert ihn mit dem übergebenen Integer-Wert

Parameter:

`pWert` Integer, mit dem dieser WerteEintrag initialisiert wird

`public WerteEintrag(String pWert)`

Erstellt einen neuen WerteEintrag und initialisiert ihn mit dem übergebenen String

Parameter:

`pWert` String, mit dem dieser WerteEintrag initialisiert wird

`public void boolSetzen(boolean pWert)` *

Speichert den übergebenen Wahrheitswert als Wert dieser Instanz und ersetzt dabei, falls vorhanden, den alten Wert

Parameter:

`pWert` Ein Wahrheitswert, der als Wert gespeichert werden soll

`public void intSetzen(int pWert)` *

Speichert den übergebenen Integer als Wert dieser Instanz und ersetzt dabei, falls vorhanden, den alten Wert

Parameter:

`pWert` Ein Integer-Wert, der als Wert gespeichert werden soll

`public void stringSetzen(String pWert)` *

Speichert den übergebenen String als Wert dieser Instanz und ersetzt dabei, falls vorhanden, den alten Wert

Parameter:
`pWert` Ein String, der als Wert gespeichert werden soll

`public boolean boolAbfragen()` * Gibt den gespeicherten Wert als Boolean zurück. Falls der Wert im Original nicht als Boolean vorliegt, wird er entsprechend konvertiert.

Rückgabewert:
Der gespeichert Wert als String

`public int intAbfragen() throws ParseException` *

Gibt den gespeicherten Wert als Integer zurück. Falls der Wert im Original nicht als Integer vorliegt, wird er entsprechend konvertiert.

Rückgabewert:
Der gespeichert Wert als Integer **Exceptions:**
Wirft eine `PharserException`, falls der Wert nicht in einen Integer umgewandelt werden konnte

`public String stringAbfragen()` *

Gibt den gespeicherten Wert als String zurück. Falls der Wert im Original nicht als String vorliegt, wird er entsprechend konvertiert.

Rückgabewert:
Der gespeichert Wert als String

Abhängigkeiten von anderen Klassen und Interfaces

- enum Datentyp

enum Datentyp

Zuständigkeit

Diese Aufzhlung beinhaltet alle Datentypen, die in WerteEintrag gespeichert werden können und sollte die Datentypen beinhalten, die bezüglich den Programmeinstellungen und der Statuswerte verwendet werden.

Konstanten

`BOOLEAN`

Ein Wahrheitswert als primitiver Wahrheitswert. In einen Integer-Wert umgewandelt wird der Zustand „Wahr“ in 1 und der Zustand „Falsch“ in 0 konvertiert.

`INTEGER`

Ein ganzzahliger Wert als primitiver Datentyp. In einen Boolean-Wert umgewandelt entsprechen der Wert 0 dem Zustand „Falsch“ und alle anderen Werte dem Zustand „Falsch“.

`STRING`

Eine Zeichenkette, die ohne Konvertierungsprobleme alle anderen Datentypen speichern kann.

class KonfigAdapter

Zuständigkeit

Diese Klasse beinhaltet Funktionen, die der XML-Lader aufrufen soll, um Einstellungen zu setzen.

Methoden

`public KonfigAdapter(IKonfigWerte werte)`

Diese Methode erstelle eine neue Instanz von KonfigAdapter. Als Parameter wird die Werteliste übergeben, in die die Einträge geschrieben werden sollen.

`public void setzeATDD(boolean wert)`

Setzt die Einstellung `ATDD` auf den übergebenen Wert

`public void setzeRefactor2(boolean wert)`

Setzt die Einstellung `Refactor2` auf den übergebenen Wert

Abhängigkeiten von anderen Klassen und Interfaces

- interface KonfigEintrag
- interface KonfigWerte

Deprecated: class KonfigWerte implements IKonfigWerte

Zuständigkeit

Diese Klasse verwaltet Name-Wert-Paare mit den Einstellungen. So ist es möglich, unabhängig von den Klassen, die die Konfigurationsdatei verarbeiten, neue Einstellungen hinzuzufügen oder zu entfernen. Einzig die verwendeten Datentypen müssen hinterlegt sein. Als Grundformat werden Zeichenketten verwendet, die dann entsprechend konvertiert werden. Schlägt dieser Vorgang fehl, wird eine Exception geworfen und die aufrufende Funktion kann einen Standardwert setzen.

Methoden

`public KonfigWerte()`

Diese Methode erstelle eine neue Instanz von KonfigWerte.

`public void einstellungEintragen(String eigenschaft, IKonfigEintrag wert)`

Erstellt einen neuen Eintrag in der Werteliste mit dem übergebenen Namen und Wert oder setzt den Wert des Eintrags mit diesem Namen, falls er schon existiert

`public IKonfigEintrag einstellungAbfragen(String eigenschaft) throws Exception`

Sucht nach dem Eintrag und gibt falls mglich seinen Wert zurck, sonst wird eine Exception geworfen

Abhängigkeiten von anderen Klassen und Interfaces

Dieses Interface besitzt keine Abhängigkeiten von anderen Klassen oder Interfaces.

Deprecated: class KonfigEintrag implements IKonfigEintrag

Zuständigkeit

Diese Klasse verwaltet einen Eintrag aus `KonfigWerte`. Dieser besteht aus dem eigentlichen Wert, einer Zeichenkette, und Konvertierungsfunktionen. Beim Erstellen kann noch kein Wert übergeben werden, das dies mit einem Interface nicht möglich ist. Stattdessen wird gespeichert, ob der Wert gesetzt ist. Ist dies nicht der Fall, während eine Eigenschaft abgefragt wird, wird eine Exception geworfen. Kann der Wert nicht wie gefordert konvertiert werden, wird eine Exception geworfen und die aufrufende Funktion kann einen Standardwert setzen.

Methoden

`public KonfigEintrag()`

Diese Methode erstelle eine neue Instanz von KonfigEintrag.

`public void wertSetzen(String wert)`

Setzt den Wert des Eintrags. Kann nicht mit dem Konstruktor zusammengelegt werden, da dieser nicht im Interface gefordert werden kann

`public boolean BooleanAbfragen() throws Exception`

Gibt falls mglich den Wert als Boolean zurck, wirft sonst eine Exception

`public boolean IntegerAbfragen() throws Exception`

Gibt falls mglich den Wert als Integer zurck, wirft sonst eine Exception

`public boolean StringAbfragen() throws Exception`

Gibt falls mglich den Wert als String zurck, wirft sonst eine Exception

Abhängigkeiten von anderen Klassen und Interfaces

Dieses Interface besitzt keine Abhängigkeiten von anderen Klassen oder Interfaces.

Anhang

Geplante Änderungen

Der Adapter wird in nächster Zeit noch um einige Einstellungen erweitert werden. Im Zusammenhang mit der Möglichkeit, dass gesamte Projekt zu speichern, sollen Funktionen geschaffen werden, um ganze Wertetabellen aus einem Datenstrom zu lesen oder in einen Datenstrom zu schreiben.

Änderungsprotokoll

30.06.2016	• Das Dokument wurde erstellt
10.06.2016	• Die Klassen KonfigWerte und KonfigEintrag mit ihren Interfaces wurden durch allgemeinere Varianten ersetzt, da sie an mehreren Stellen im Programm eingesetzt werden sollen. Dabei wurden die beiden Klassen deutlich verbessert.