Über gute Modularisierung und Kopplung sowie die Vorzüge und Grenzen des funktionalen Ansatzes am Beispiel eines Frameworks für Gesellschaftsspiele

Übersicht

- Referenzielle Transparenz
- Kategorienmodell der Bedeutung und OOP
- Kopplung und Modularisierung (Konzeptebene)
- Überblick über weitere Themen

Referenzielle Transparenz - Standarddefinition

- **Arbeitsdefinition für funktionale Programmierung**: Programmierstil, bei dem man sich (von Typdeklarationen abgesehen) auf die Definition reiner Funktionen beschränkt.
- Referenzielle Transparenz: In seiner Standarddefinition ein hierzu im Grunde äquivalentes Konzept.

Referenzielle Transparenz – Praktischer Nutzen?

- Auswertungsergebnis von Ausdrücken ist von der Ausführungshistorie des Algorithmus unabhängig.
- Aufgrund der Seiteneffektfreiheit ist das Auswertungsergebnis der einzige zu berücksichtigende Effekt.

  => bessere Verständlichkeit des Quellcodes

Referenzielle Transparenz – Mein Definitionsvorschlag

Ein Bezeichner ist refeferenziell transparent, wenn seine **Bedeutung** stabil ist gegenüber der Ausführungshistorie des Algorithmus.

(Ein Algorithmus ist referenziell transparent, wenn alle in ihm auftretenden Bezeichner referenziell transparent sind.)

Anmerkungen zum Konzept Bedeutung:

- Bedeutung vs. Referenz
- Internalistische (Computer-interne) vs. externalistische Semantik

=> Bedeutung hier: Reale Größe oder Entität, die ein Bezeichner repräsentiert.

Referenzielle Transparenz – Liberalisierte funktionale Programmierung

Reinheit ist hinreichend für referenzielle Transparenz, aber nicht mehr notwendig. Beispiele für unreine aber referenziell transparente Elemente:

- Zufallswerte
- Externe Steuervariablen
- Einlesen von der Konsole

Hinweis: Fregesches Kompositionalitätsprinzip

Referenzielle Transparenz – Imperative Programmierung

Arten von Softwareproblemen:

- Berechnungspobleme
- Dynamische Modellierungsprobleme

Für letztere: Die Veränderlichkeit des Modells selbst steht nicht im Widerspruch zur Gewährung referenzieller Transparenz, sondern bildet vielmehr deren Voraussetzung.

Problem allerdings: Synchronisationspunkte

Kategorienmodell der Bedeutung und OOP

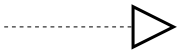
- Unterklassenbildung in der OOP ist das Analogon zur Unterbriffsbildung in der natürlichen Sprache.
- Bedeutung eines Begriffs wird durch notwendige und hinreichende Bedingungen konstituiert, die unter den Begriff fallenden Phänomene erfüllen müssen.
- Unterbegriffe erweitern diese Bedingungen.
- Da es sich bei diesen konstitutiven Bedingungen auch um Verhaltensdispositionen handeln kann, folgt aus dem Kategorienmodell das Liskovsche Substitutionsprinzip der OOP: Subtypen müssen mindestens das Verhalten des Basistyps zusichern, d.h., Subtypen dürfen die Schnittstelle nur erweitern.
- Bei der Unterklassenbildung geht die Erweiterung des Moduls mit der Spezialisierung des repräsentierten Konzepts Hand in Hand.

Kopplung und Modularisierung (Konzeptebene)

- Kopplung: Abhängigkeit eines Moduls von einem anderen Modul
- Unterklassenbildung = Schnittstellenvererbung + Implementierungsvererbung
- Schnittstellenvererbung auch separat möglich

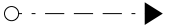
Kopplung und Modularisierung (Konzeptebene) - Kopplungsformen

: Vererbung zwischen Klassen, i.e.,
Vererbung der Schnittstelle und der Implementierung
(links erbt von rechts)



: Vererbung der Schnittstelle
(links erbt von rechts)

: Benutzung der Schnittstelle eines Konzepts
(links benutzt rechts)



: Verwendung des Ereignismusters =>
Einsparung einer (Rück-)Kopplung
Links: Sender, rechts: Empfänger

Kopplung und Modularisierung (Konzeptebene) – einige Kriterien für eine gutartige Architektur

- Wiederverwendbarkeit
  - Einzelner Komponenten
  - Der Architektur selbst (Flexibilität, Erweiterbarkeit)
- Lokalität von Änderungen
- Verständlichkeit
  - Übersichtlichkeit

Entwurfsprinzipien:

0. Zirkelfreiheit:
- Verständlichkeit
- Wiederverwendbarkeit
- Lokalität von Änderungen

1. Autarkieprinzip
- Voraussetzung für 2

2. Prinzip der thematischen / begrifflichen Zerfällung
- Übersichtlichkeit
- Wiederverwendbarkeit
- Voraussetzung für 3

3. Prinzip der allgemeinst-möglichen Abhängigkeit
- Wiederverwendbarkeit
- Lokalität von Änderungen

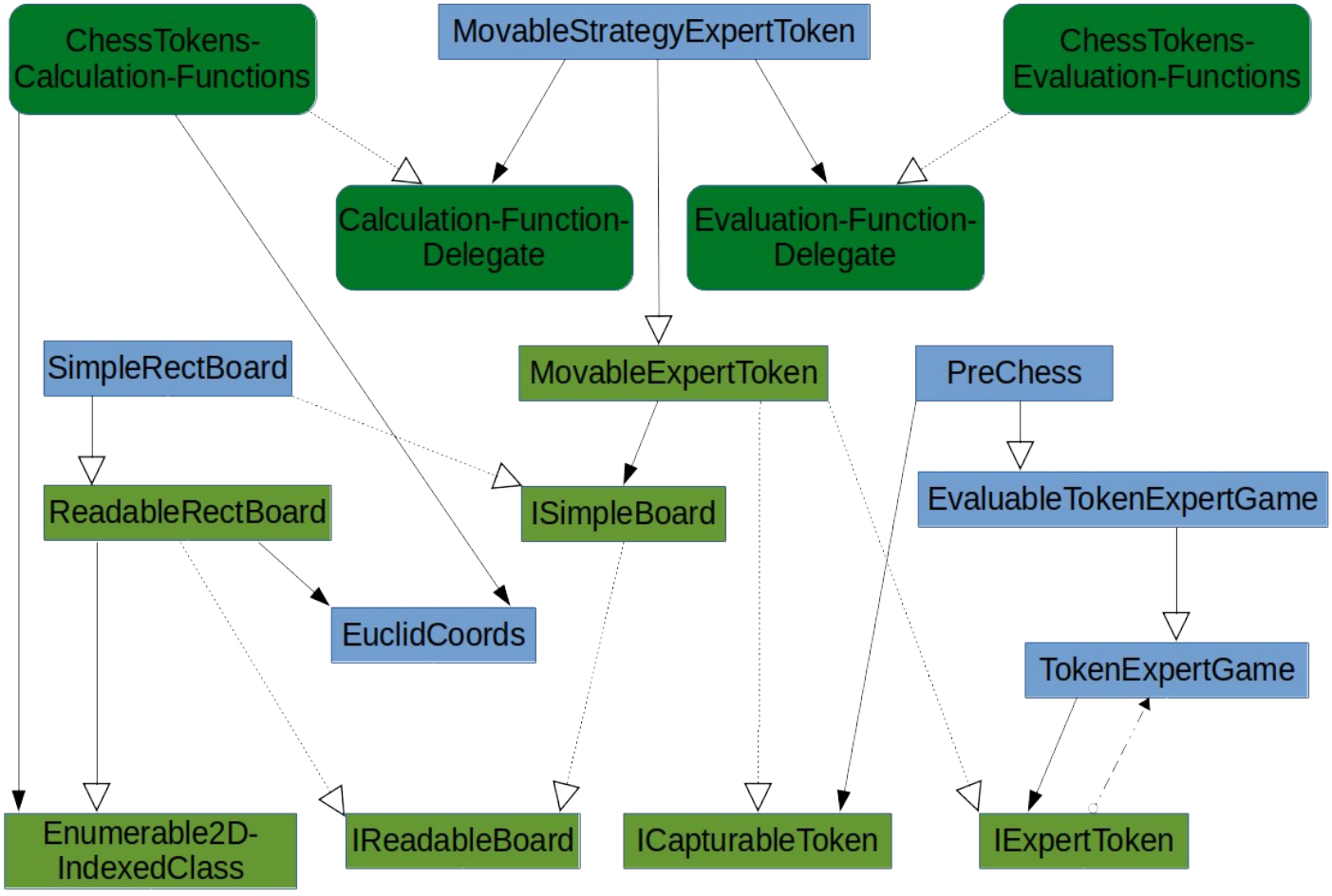


Abb 6 : PreChess

4. Prinzip der Übersprungsfreiheit
- ebenso

Überblick über weitere Themen

- Vereinbarkeit von funktionaler Programmierung und Objektorientierung
- Inversion of Control (neue Definition, Beziehung zum Open-closed-Prinzip)
- Implementierungsvererbung und tatsächliche / vermeintliche Aternativen