Arbeitsblatt 1 - GKA - Klauck
Team: 08, Matthias Nitsche, Swaneet Sahoo

Aufgabenaufteilung:
Swaneet:
AIGraphImpl, Edge, Vertice, AIGraphTest

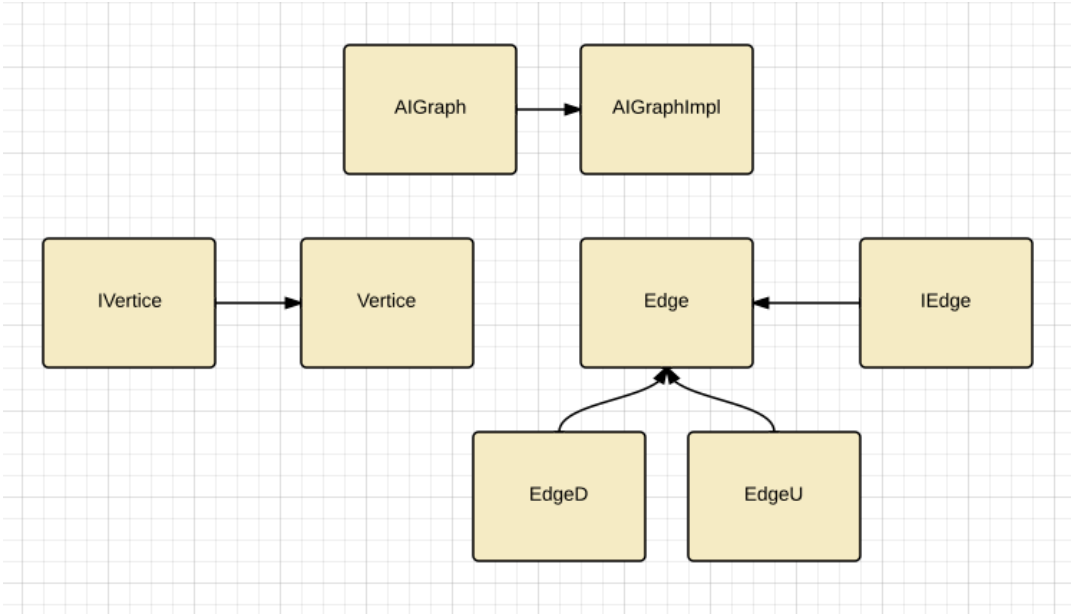
Matthias:
AIGraph, EdgeD, EdgeU, IVertice, IEdge, AIGraphImpl, Edge, Vertice, AIGraphTest

Es lief sehr ausgeglichen, wir saßen die meiste Zeit nebeneinander. Es ist nicht möglich auseinanderzuhalten wer was genau gemacht hat, es sind daher nur Anteile. Grob hat jeder aber alles implementiert.

Bearbeitungszeitraum:
Swaneet: 2-3 Stunden
Matthias: 2-3 Stunden
Zusammen: 5-6 Stunden
Insgesamt: 9-12 Stunden

Aktueller Stand:
ADT Interface ist fertig.
Test Abdeckung ca 90% (keine Negativen Tests (Exception-Testing).
GraphParser ist in der BetaPhase
GraphUtils fehlen

Skizze:



Das Vorgehen der Gruppe war schnell geklärt. Beide implementieren letztendlich jede Funktionalität und im Refactoring werden dann Verständnisfragen geklärt. Wir haben Agil mit kurzen Erklärungsrunden gearbeitet und viel Pair Programming. Teilweise Exploration zum schnellen Testen von neuen Methoden.

Die Skizze beschreibt relativ klar unsere Vererbung. Jede Entität, daher AIGraphImpl, Vertice, Edge haben ein eigenes Interface AIGraph, IVertice, IEdge mit Vorgaben. Edge besitzt außerdem zwei Unterklassen EdgeD und EdgeU. Diese sind nur für die Klarheit implementiert und besitzen als unterschied eine Hood Methode isDirected die Im Falle von EdgeD True und im Falle von EdgeU False zurückgibt. Es gibt also keine Probleme gemischte Graphen zu bauen. Der AIGraph hat die Aufgabe Edge und Vertice zu speichern (in diesem Falle in zwei HashMaps die ID und Objekt halten) und das vorgegebene Interface von den Vorlesungsfolien zu implementieren. Der AIGraph kennt jedoch nicht die Beziehung zwischen Edge und Vertice. Jeder Edge hält genau zwei Vertices.

Für den Zugriff ist dies sehr viel einfacher.

Die IDs werden intern vergeben und im AIGraphImpl gespeichert, bei jedem neuen Graphen werden ID's wieder von 0 hochgezählt. Der Nutzer hat damit nicht die Aufgabe ID's selbst zu setzen. Sowohl Vertice als auch Edges haben eine UniqueID dadurch.
Der User kann nur über den AIGraphen auf den Graphen zugreifen und die entsprechenden Interface-Methoden benutzen. Die unteren Implementationen bekommt dieser niemals zu Gesicht.

Quellen:
- Theoretische Annahmen und Grundlagen aus den Klauck Vorlesungsfolien
Begründung für Codeübernahme: -