1 Organisatorisches

1.1 Team

- Reinhard Penn, s1110306019
- Bernhard Selymes, s1110306024

1.2 Aufteilung

- Reinhard Penn
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Milometer, Speedometer, Observer
  - Testen aller Klassen
- Bernhard Selymes
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Subject, PKW, RevolutionCounter
  - Dokumentation

1.3 Zeitaufwand

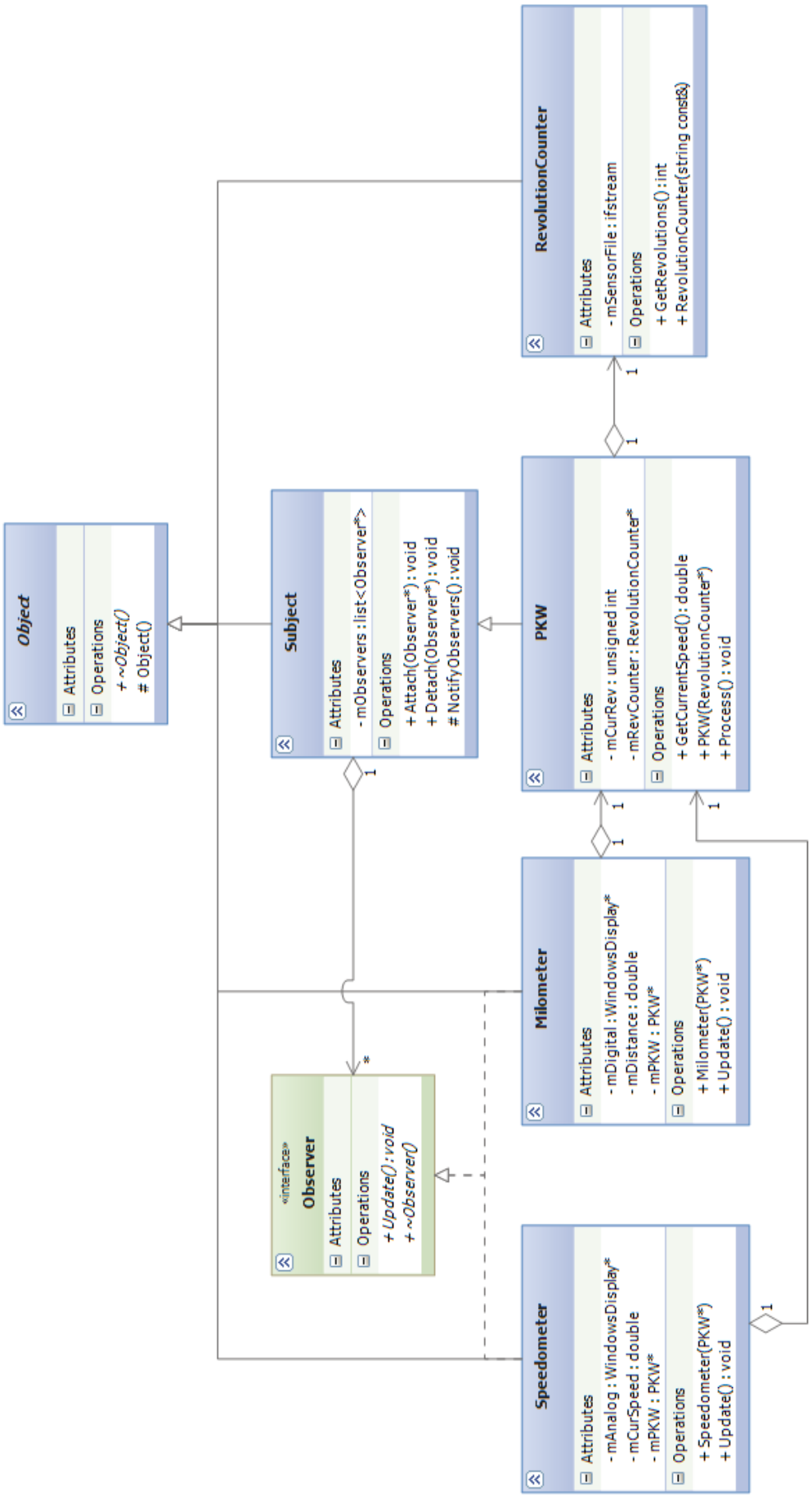
- geschätzte Mh: 10
- tatsächlich: Reinhard (5h), Bernhard (5h)

2 Systemspezifikation

Eine Software für einen PKW-Prüfstand ist zu entwickeln. Die Software soll Raddrehzahlen alle 500ms aus einer Datei einlesen und diese verarbeiten. Sie soll die Momentangeschwindigkeit und die gefahrenen Kilometer berechnen. Die Momentangeschwindigkeit wird auf einem analogen Display, die gefahrenen Kilometer auf einem digitalen Display angezeigt.

3 Systementwurf

3.1 Klassendiagramm



3.2 Komponentenübersicht

- Klasse ”Object”:
  Basis aller Basisklassen.
- Klasse ”Subject”:
  Basisklasse für Subjects, die überwacht werden.
- Klasse ”Observer”:
  Basisklasse für Observer.
- Klasse ”Speedometer”:
  Konkreter Observer, der die Momentangeschwindigkeit ermittelt. Überwacht PKW.
- Klasse ”Milometer”:
  Konkreter Observer, der die gefahrenen Kilometer ermittelt. Überwacht PKW.
- Klasse ”PKW”:
  Konkretes Subject, das überwacht wird.
- Klasse ”RevolutionCounter”:
  Klasse die die Umdrehungen pro Minute zur Verfügung stellt.

4 Komponentenentwurf

4.1 Klasse ”Object”

Abstrakte Basisklasse aller Klassen. Von ihr werden alle anderen Klassen abgeleitet. Beinhaltet einen virtuellen Destruktor.

4.2 Klasse ”Subject”

Hat eine Liste mit Zeigern auf Observer. Observer können via Methoden hinzugefügt, entfernt oder benachrichtigt.

Methode ”Attach”:
Schnittstelle:
Parameter: Observer*
Rückgabetyp: void.
Fügt einen noch nicht vorhandenen Observer hinzu.

Methode ”Detach”:
Schnittstelle:
Parameter: Observer*
Rückgabetyp: void.
Entfernt einen Observer, wenn er vorhanden ist.

Methode ”NotifyObservers”:
Schnittstelle:
Rückgabetyp: void.
Ruft von allen Observern Update auf.

4.3 Interface ”Observer”

Bietet die Schnittstelle für einen Observer. Hat einen virtuellen Destruktor.

Methode ”Update”:
Schnittstelle:
Rückgabetyp: void.
Pure virtual function.

4.4 Klasse ”Speedometer”

Gibt die aktuelle Geschwindigkeit auf einem analogen Display aus.

Konstruktor ”Speedometer”:
Schnittstelle:
Parameter: PKW*
Fügt den PKW hinzu und fügt dem PKW sich selbst zu. Fordert Speicher für ein neues Display an.

Methode ”Update”:
Schnittstelle:
Rückgabetyp: void.
Ruft die Funktion GetCurrentSpeed von PKW auf und gibt den Wert dem analogen Display weiter.

4.5 Klasse ”Milometer”

Gibt die gefahrenen Kilometer auf einem Display aus.

Konstruktor ”Milometer”:
Schnittstelle:
Parameter: PKW*
Fügt den PKW hinzu und fügt dem PKW sich selbst zu. Setzt die Kilometer auf 0. Fordert Speicher für ein neues Display an.

Methode ”Update”:
Schnittstelle:
Rückgabetyp: void.
Ruft die Funktion GetCurrentSpeed von PKW auf, rechnet daraus die in den letzten 500ms gefahrenen Kilometer aus, addiert sie zu den gesamt gefahrenen Kilometern und gibt das Ergebnis dem analogen Display weiter.

4.6 Klasse ”PKW”

Stellt einen konkreten PKW dar. Hat einen Member, der die aktuellen Radumdrehungen speichert und einen der eine Referenz auf einen Radumdrehungszähler hat.

Konstruktor ”PKW”:
Schnittstelle:
Parameter: RevolutionCounter*
Fügt dem PKW einen Radumdrehungszähler hinzu.

Methode ”GetCurrentSpeed”:
Schnittstelle:
Rückgabetyp: double
Berechnet die aktuelle Geschwindigkeit.

Methode ”Process”:
Schnittstelle:
Rückgabetyp: void
Aktualisiert die aktuellen Radumdrehungen und benachrichtigt dann die Observer.

4.7 Klasse ”RevolutionCounter”

Klasse für die Radumdrehungen. Hat einen Member der den Filestream mit den Daten speichert. Im Konstruktor wird dieser Stream geöffnet, im Destruktor geschlossen.

Konstruktor ”RevolutionCounter”:
Schnittstelle:
Parameter: std::string const& filename
Öffnet die gegebene Datei.

Methode ”GetRevolutions”:
Schnittstelle:
Rückgabetyp: int
Liest die Daten aus der Daten aus und gibt sie zurück.

5 Source Code

6 Testausgaben