1 Organisatorisches

1.1 Team

- Reinhard Penn, s1110306019
- Bernhard Selymes, s1110306024

1.2 Aufteilung

- Reinhard Penn
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Client, Slot, RemoteControl
  - Testen aller Klassen
- Bernhard Selymes
  - Planung
  - Klassendiagramm
  - Implementierung der Device und Command Klassen
  - Dokumentation

1.3 Zeitaufwand

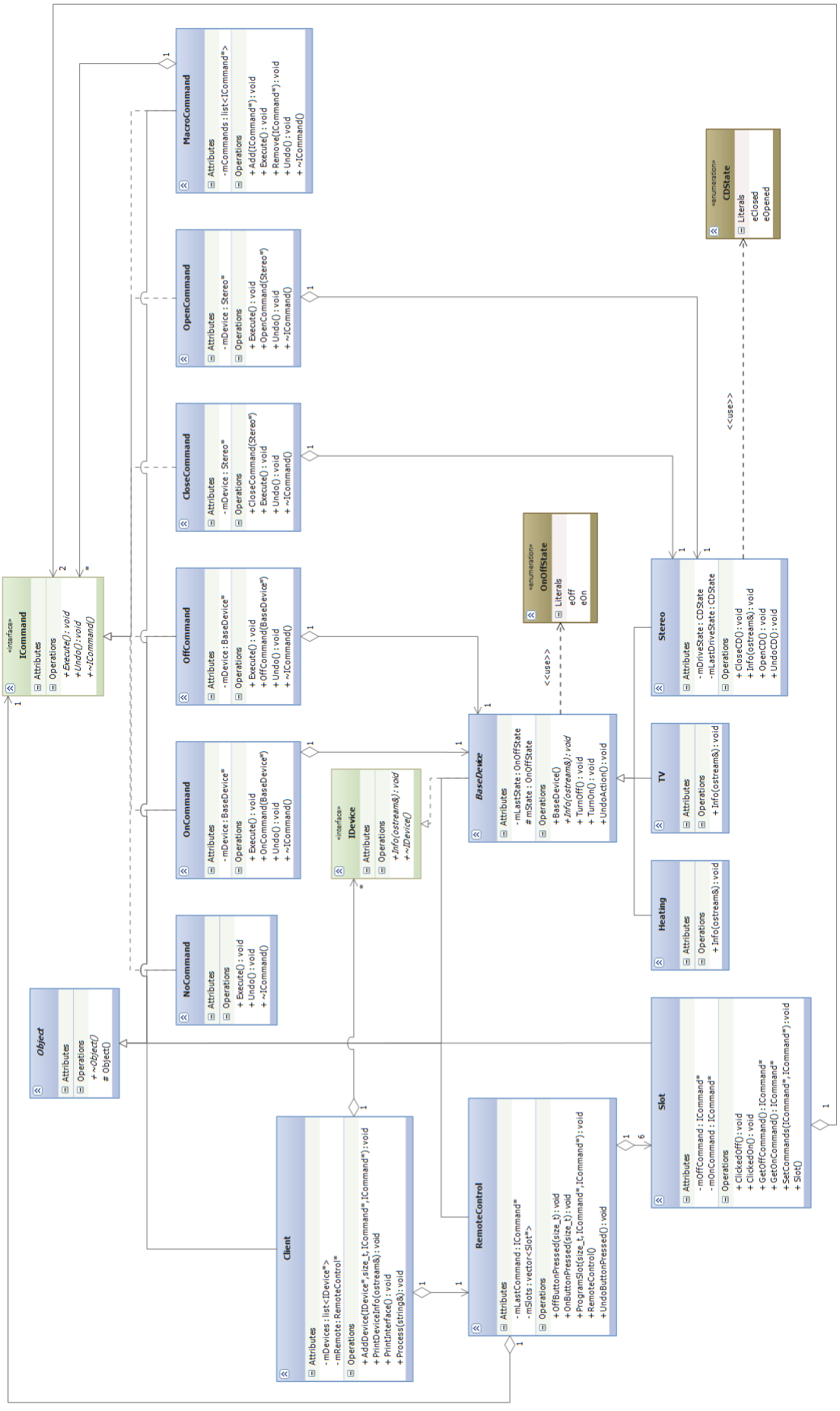
- geschätzte Mh: 15
- tatsächlich: Reinhard (6h), Bernhard (8h)

2 Systemspezifikation

Eine Software für eine programmierbare Fernsteuerung soll entworfen werden. Mit der Fernsteuerung können verschiedene Geräte ein- und ausgeschalten werden. Die Fernbedienung hat 6 Slots die aus je einer On und Off Taste bestehen. Die siebte Taste ist die Undo Taste mit der die letzte Eingabe zurückgenommen werden kann. TV-Geräte, Heizungen und Stereoanlagen können ferngesteuert werden. Alle können ein und ausgeschalten werden, die Stereoanlage zusätzlich geöffnet und geschlossen werden. Ein Kommandozeileninterface und die Geräteinformationen können ausgegeben werden.

3 Systementwurf

3.1 Klassendiagramm



3.2 Komponentenübersicht

- Klasse ”Object”:
  Basis aller Basisklassen.
- Klasse ”Client”:
  Verwaltet die Geräte und kann deren Informationen ausgeben und verarbeitet die Eingaben vom Benutzer.
- Klasse ”RemoteControl”:
  Verwaltet die Slots und kann die Slots programmieren.
- Klasse ”Slot”:
  Verwaltet die Kommandos eines Slots.
- Interface ”ICommand”:
  Schnittstellenbeschreibung für die Kommandos.
- Klassen ”OffCommand, OnCommand, CloseCommand und OpenCommand”:
  Konkrete Kommandoklassen.
- Klasse ”NoCommand”:
  Standard Kommando, das nur etwas ausgibt.
- Klasse ”MacroCommand”:
  Zusammenfassung mehrerer Kommandos.
- Interface ”IDevice”:
  Schnittstellenbeschreibung für die Geräte.
- Klasse ”BaseDevice”:
  Basisklasse für die Geräte.
- Klassen ”Heating, TV und Stereo”:
  Konkrete Geräteklassen.
- Enumeration ”CDState”:
  Status des CD-Laufwerks.
- Enumeration ”OnOffState”:
  Status ob ein- oder ausgeschalten.

4 Komponentenentwurf

4.1 Klasse ”Object”

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

4.2 Klasse ”Client”

Hat eine Liste die die Geräte verwaltet und einen Member der eine Referenz auf die Fernsteuerung speichert.

Methode ”AddDevice”:
Schnittstelle:
Parameter: IDevice*, size_t
Rückgabetyp: void
Fügt der Liste ein Gerät hinzu. Falsche Eingaben werden berücksichtigt. Ruft ProgramSlot von der Fernbedienung auf.

Methode ”PrintDeviceInfo”:
Schnittstelle:
Parameter: ostream&
Rückgabetyp: void
Gibt die Informationen der Geräte auf dem mitgegebenen Stream aus.

Methode ”PrintInterface”:
Schnittstelle:
Rückgabetyp: void
Gibt das Kommandozeilen-Interface auf der Konsole aus.

Methode ”Process”:
Schnittstelle:
Parameter: string&
Rückgabetyp: void
Verarbeitet den in der Konsole eingegebenen string und ruft die dazugehörigen Methoden der Fernsteuerung auf.

4.3 Klasse ”RemoteControl”

Hat einen Vektor der Referenzen auf die Slots speichert und einen Member der das letzte Kommando speichert.

Konstruktor ”RemoteControl”:
Erstellt die Slots und setzt die Kommandos auf NoCommand.

Methode ”OffButtonPressed”:
Schnittstelle:
Parameter: size_t
Rückgabetyp: void

Speichert das aktuelle Kommando im Member und ruft das Off-Kommando vom entsprechenden Slot auf.

Methode ”OnButtonPressed”:
Schnittstelle:
Parameter: size_t
Rückgabetyp: void
Speichert das aktuelle Kommando im Member und ruft das On-Kommando vom entsprechenden Slot auf.

Methode ”UndoButtonPressed”:
Schnittstelle:
Rückgabetyp: void
Ruft vom letzten Kommando die Methode Undo auf und setzt den Pointer auf 0, weil nur ein Mal zurückgesetzt werden kann.

Methode ”ProgramSlot”:
Schnittstelle:
Parameter: size_t, ICommand*, ICommand*
Rückgabetyp: void
Mit dieser Methode werden die Slots der Fernbedienung programmiert, das heißt die Kommandos werden zugewiesen.

4.4 Klasse ”Slot”

Speichert einen Pointer auf ein On- und ein Offkommando. Hat 2 Get-Methoden für diese und einen Destruktor der sie freigibt.

Konstruktor ”Slot”:
Weißt den Kommandozeigern 0 zu.

Methode ”ClickedOff”:
Schnittstelle:
Rückgabetyp: void
Überprüft den Pointer und ruft ”Execute” vom Off-Kommando auf.

Methode ”ClickedOn”:
Schnittstelle:
Rückgabetyp: void
Überprüft den Pointer und ruft ”Execute” vom On-Kommando auf.

Methode ”SetCommands”:
Schnittstelle:
Parameter: ICommand*, ICommand*
Rückgabetyp: void
Weist die Kommandos zu.

4.5 Interface ”ICommand”

Schnittstellendefiniton. Hat einen virtuellen Destruktor.

Methode ”Execute”:
Schnittstelle:
Rückgabetyp: void

Methode ”Undo”:
Schnittstelle:
Rückgabetyp: void

4.6 Klassen ”OffCommand, OnCommand, CloseCommand, OpenCommand und NoCommand”

Implementieren die Methoden Execute und Undo entsprechend der jeweiligen Klasse. Bei NoCommand wird einfach auf der Konsole ausgegeben, dass es sich um NoCommand handelt. Die anderen Kommandos rufen die entsprechenden Methoden in den Klassen, auf die sie eine Referenz haben, auf. Sie haben weiters einen Konstruktor dem diese Referenzen mitgegeben werden.

4.7 Klasse ”MacroCommand”

Hat eine Liste die die Referenzen auf mehrere Kommandos speichert. Der Liste können die Kommandos hinzugefügt werden, aber auch wieder entfernt werden. Es können nur maximal 2 Elemente in der Liste gespeichert werden.

4.8 Interface ”IDevice”

Schnittstellendefiniton. Hat einen virtuellen Destruktor.

Methode ”Info”:
Schnittstelle:
Parameter: ostream&
Rückgabetyp: void

4.9 Klasse ”BaseDevice”

Speichert den aktuellen und den letzten On-Off-Status. Der Konstruktor setzt beide Statusse auf Off. Die Methoden ”TurnOn” und ”TurnOff” speichern immer den aktuellen Status und weisen dann den neuen zu. Bei ”Undo” wird der letzte Status dem neuen zugewiesen.

4.10 Klassen ”Heating, TV und Stereo”

Die Methode ”Info” gibt die Informationen des jeweiligen Objektes entsprechend aus. Die Klasse Stereo definiert zusätzlich die für das CD-Laufwerk benötigten Member und Methoden.

5 Source Code

6 Testausgaben