

HTWK Leipzig
Fachbereich IMN
Sommersemester 2014

Traincard

Beleg im Smartcard Programmierung
bei Prof. Dr. rer. nat. Uwe Petermann

Kurt Junghanns, B.Sc.
Marcel Kirbst, B.Sc.
Michael Reher, B.Sc.

11. Juli 2014

Inhaltsverzeichnis

1	Einleitung	4
2	Anwendungsfall	5
2.1	Anwendungsfall aus Sicht des Trainierenden	5
2.2	Anwendungsfall aus Sicht des Trainers	5
3	Grundlagen	7
3.1	Grundlagen Smartcard	7
3.2	JCOP	7
3.3	APDU	8
4	Umsetzung	10
4.1	Kommunikation	10
4.2	Datenstruktur	12
4.3	On-Card Teil	13
4.4	Off-Card Teil	15
5	Zusammenfassung und Ausblick	25
6	Quellenverzeichnis	26

Abbildungsverzeichnis

1	APDU-Kommunikation schematisch dargestellt, Quelle: Autoren . . .	8
2	Selbstdefinierter Dateninhalt der APDUs	11
3	Klassendiagramm On-Card Teil	14
4	Hauptmenü	16
5	gesamter Trainingsplan	17
6	Tagesplan	18
7	Eingabemenü des Trainers	19
8	Eingabemenü des Trainers	20
9	Eingabemenü des Sportlers	21
10	Trainingsfortschritt ohne Einträge	22
11	Trainingsfortschritt mit Einträgen	23
12	Klassendiagramm Off-Card Teil	24

Tabellenverzeichnis

1 Einleitung

Diese Arbeit befasst sich mit der Vorstellung der Belegarbeit im Fach Smartcard Programmierung mit dem Thema “Traincard”. Ziel des Belegs ist die prototypische Implementierung eines Trainingssystems auf Basis JCOP-fähiger Smartcards.

Im ersten Kapitel [Anwendungsfall](#) wird der Anwendungsfall für die prototypische Implementierung beschrieben, sowohl unter [Anwendungsfall aus Sicht des Trainierenden](#) aus Sicht des Trainierenden, wie auch unter [Anwendungsfall aus Sicht des Trainers](#) aus Sicht des Trainers. Im Kapitel [Grundlagen](#) werden die grundlegenden Technologien und Standards vorgestellt. Kapitel [Umsetzung](#) erläutert detailliert die Systemumgebung, beschreibt den On-Card und den Off-Card Teil der Implementierung sowie deren Kommunikation. Abschließend wird im Kapitel [Zusammenfassung und Ausblick](#) eine Einschätzung des Belegs gegeben.

2 Anwendungsfall

In Fitnessstudios werden heute noch bei der Neuanmeldung Trainingspläne auf Papier ausgegeben. Der Trainierende führt den Trainingsplan über die Trainingsperiode ständig bei sich und verzeichnet den Trainingsfortschritt in diesem Dokument. Im Verlauf der Trainingsperiode kann der Trainer beurteilen, wie effektiv das Training beim Trainierenden ist.

Die Verwendung von Trainingspläne auf Papier hat jedoch einige Nachteile. Beispielsweise wird das Dokument vom Trainierenden manchmal vergessen, die betreffenden Trainingsfortschritte müssen also nachgetragen werden. Weiterhin sind Trainingspläne in Papierform nur bedingt resistent gegen Abnutzung und verschleiß mit fortdauernder Verwendung. Um den genannten Nachteilen zu begegnen soll im Rahmen dieser Belegarbeit der Einsatz von Trainingsplänen auf Basis so genannter Smartcards abgebildet werden. Smartcards können sehr leicht in einer Brieftasche mitgeführt werden und sind weniger anfällig für Abnutzung. Weiterhin bieten sie weitere Vorteile wie automatische elektronische und anonyme Datenerfassung zu den Sportlern, die Möglichkeit die Daten extern zu sichern und bei Bedarf auf eine beliebige Smartcard zu exportieren.

2.1 Anwendungsfall aus Sicht des Trainierenden

Aus Sicht des Trainierenden bietet die Verwendung der Smartcard einige der folgenden Vorteile:

- leichte Transportierbarkeit
- robuster als Trainingspläne aus Papier

2.2 Anwendungsfall aus Sicht des Trainers

Trainer profitieren beim Einsatz von Smartcards unter anderem von folgenden Vorteilen:

- leichte und lesbare Auswertung der vorgegebenen Trainingspläne
- zusätzliche Metainformationen wie beispielsweise zu welchem Zeitpunkt welcher Datensatz geschrieben wurde
- statistische Auswertungen lassen sich sehr leicht erstellen
- manipulationssicher

- es fallen zusätzlich Metadaten an die ausgewertet werden können (z.B.: wann wurde die Karte an welchem Gerät benutzt)

3 Grundlagen

In diesem Kapitel wird auf die Grundlagen der in diesem Beleg verwendeten Technologien eingegangen.

3.1 Grundlagen Smartcard

Die in diesem Beleg verwendete Smartcard basiert auf der Java Card Technologie. Die Java Card Technologie bietet eine Teilmenge der Java Programmiersprache, sowie eine hinsichtlich der Anforderungen an Smartcards optimierte Laufzeitumgebung.

Die Verwendung von Java-basierten Smartcards bietet einige Vorteile, von denen nachfolgend eine Auswahl beispielhaft genannt sei: [2]

plattformunabhängig: Java Card Applets, die der Java Card API entsprechen, lassen sich plattformunabhängig und herstellerübergreifend nutzen.

Multiapplikationsfähig: es können mehrere Applikationen gleichzeitig auf einer Smartcard ausgeführt werden

hohe Flexibilität: die Verwendung der objektorientierten Programmiersprache Java erlaubt die Erstellung komplexer Anwendungen für die Smartcard.

Post-Aktualisierbarkeit: die Möglichkeit, nachträglich Code auf der Smartcard zu modifizieren und auszutauschen erhöht die Flexibilität weiter

Standardkonformität: die Java Smartcards entsprechen dem ISO7816 Standard [3]

Eine Java Smartcard besteht im Wesentlichen aus den Bestandteilen Kommunikationsschnittstelle, Speicher und einem Prozessor zur Durchführung von Berechnungen. Bei Verwendung der Smartcard wird diese in ein Lesegerät eingelegt. Das Lesegerät wird in einschlägiger Literatur auch als Card Acceptance Device, abgekürzt CAD, bezeichnet. Der Speicher auf Java Smartcards besteht aus zwei Typen, RAM und EEPROM. Der RAM-Speicher ist flüchtig und kann beliebig oft beschrieben werden. Der EEPROM-Speicher ist nichtflüchtig und kann nur endlich oft beschrieben werden, je nach Hersteller und Modell bis zu 100.000 mal pro Speicherzelle.

3.2 JCOP

Die Java Card Open Plattform, abgekürzt JCOP, ist ein Smartcard Betriebssystem das initial von IBM entwickelt wurde, inzwischen jedoch von NXP Semiconductors

betreut wird. Der Titel leitet sich von den Namen der zu Grunde liegenden Spezifikationen für Java Card und Global Platform (früher bekannt unter Open Plattform) ab.

Bei der Softwareentwicklung kann auf reale sowie auf simulierte JCOP-Karten zurückgegriffen werden. Der Zugriff auf eine reale JCOP-Karte erfolgt mittels eines Terminal-Kartenlesers sowie spezieller Treiber. Da während der Erarbeitung des Belegs keine Hardware zur Verfügung stand, wurde die Implementierung an einer simulierten JCOP-Karte getestet. Auf die Entwicklungsumgebung wird im Kapitel [Umsetzung](#) näher eingegangen.

3.3 APDU

Die Kommunikation zwischen dem Kartenlesegerät und der Smartcard erfolgt reaktiv und paketweise. Reaktiv bedeutet, dass die Smartcard keine Kommunikation initiiert sondern nur auf Anfragen vom Lesegerät reagiert. Der bei der Kommunikation zwischen Smartcard und Lesegerät verwendete Kommunikationsmechanismus wird als Application Protocol Data Units, abgekürzt APDU, bezeichnet. Spezifiziert ist dies ebenfalls in ISO7816 Teil 4.[3]

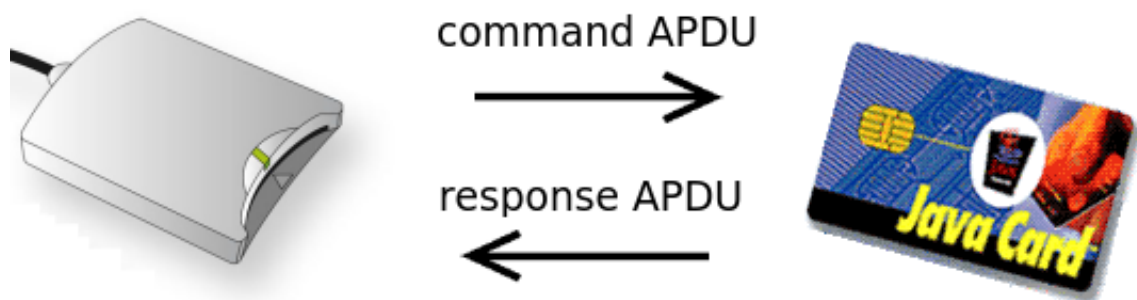


Abbildung 1: APDU-Kommunikation schematisch dargestellt, Quelle: Autoren

Die Kommunikation wird über ein so genanntes command APDU Paket initiiert, welches aus den folgenden Feldern besteht:

CLA: class, gibt die Klasse an, spezifiziert ob es sich um ein ISO7816-4 konformes Kommando handelt

INS: gibt die Instruktion an

P1: zusätzlicher Parameter

P2: zusätzlicher Parameter

Weiterhin können je nach Kommandotyp noch die folgenden, optionalen Felder an das command APDU Paket angehängt werden:

Lc: Length, gibt die Länge der Kommandodaten

Data: gibt die Kommandodaten an

Le: gibt die Länge der erwarteten Antwort an

Als Antwort erhält das Lesegerät von der Smartcard ein so genanntes response APDU Paket. Dieses Antwortpaket kann ein Datenfeld enthalten, dies ist jedoch nicht obligatorisch. Das response APU Paket ist wie folgt aufgebaut.

Data: optionales Datenfeld

Sw1: Statusword, erstes Byte

Sw2: Statusword, zweites Byte

4 Umsetzung

Dieses Kapitel legt die Umsetzung dar.

4.1 Kommunikation

Die Anwendung besteht aus einem On-Card und einem Off-Card Teil.

Entsprechend der JCOP Umgebung, ist der On-Card Teil durch ein Applet realisiert, welches auf der Smartcard installiert und gestartet wird. Als Smartcard kann eine physische oder eine emulierte zum Einsatz kommen. Im Rahmen dieses Projektes wird die Smartcard per JCOP Eclipse Umgebung emuliert und verwendet. Dabei ist in der gestarteten JCOP Shell der Befehl */close* auszuführen, wodurch die Smartcard für externe Zugriffe auf der lokalen IP des Emulator-Rechners auf dem Port 8090 erreichbar ist.

Der Off-Card Teil ist ebenfalls in Java geschrieben und kommuniziert mit Hilfe des OpenCard Frameworks mit der Smartcard.

Die Kommunikation zwischen On-Card und Off-Card auf verschiedenen Rechnern benötigt entsprechende Regeln der Firewalls der Rechner, lokale Ausführung beider Teile auf einem Rechner hingegen keine.

Der Inhalt der APDUs orientiert sich stark an der Norm ISO 7816-4. Der konkrete Aufbau ist in Abbildung 2 dargestellt.

Die Abkürzungen stehen dabei für folgendes:

CLA ein Byte, welches immer mit dem Wert 0 belegt wird, da die APDU nicht ISO 7816-4 konform ist

INS ein Byte, welches die Instruktion angibt

NOA ein Byte, welches die Anzahl an gesamt zu übertragenden APDUs zur vollen Abarbeitung der Instruktion angibt

LEN ein Byte, welches die Anzahl der folgenden Daten-Bytes aufzeigt

DATA beliebige Anzahl von Bytes, welche die Daten enthalten¹

SW1 ein Byte, welches den vorderen Teil der Statusrückgabe darstellt

SW2 ein Byte, welches den zweiten Teil der Statusrückgabe enthält

¹ Anzahl der Daten-Bytes sollte LEN entsprechen

Die Statusrückgabe wird automatisch durch die JCOP Umgebung an die APDU angehängen. Die Interpretation der Statusrückgabe ist anhand ISO 7816-4 durchzuführen.

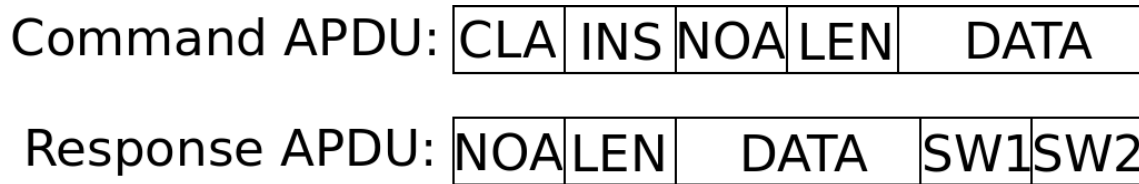


Abbildung 2: Selbstdefinierter Dateninhalt der APDUs

APDUs enthalten bis zu 256 Bytes, davon bis zu 252 Bytes für Daten. Es sind ganze Trainingspläne und andere Objekte im Byte Format zwischen On-Card und Off-Card Teil zu übertragen. Dazu existiert pro Datenmodell in der dazugehörigen Klasse je eine Funktion zur Umwandlung von Instanz zu Byte Array und zur Erzeugung einer Instanz aus einem Byte Array. Diese Umwandlung wurde eigens implementiert und orientiert sich am Aufbau von Ethernet Paketen. Ein Paket oder Byte Array besteht dabei immer aus einem Byte für einen eindeutigen Identifikator des Modells, zwei Bytes für die Anzahl der folgenden Bytes sowie Daten Bytes. Eine Instanz des Modells MyDate mit den Attributen Jahr², Monat und Tag wird in folgende Byte Repräsentation umgewandelt: 01 00 03 0e 07 0b

Bei der Realisierung des Trainingssystems werden keine Sicherheitsrelevanten Daten zwischen On-Card und Off-Card Teil übertragen. Auf Grund dessen wurde auf eine verschlüsselte Kommunikation mit asymmetrischer oder symmetrischer Verschlüsselung verzichtet. Einzig Rollen-spezifische Schreibvorgänge sind in beiden Card Teilen durch je ein Passwort geschützt. Der Trainer und der Sportler besitzen jeweils ein Passwort, welches mit der Java Bibliothek `java.security.MessageDigest` und der darin enthaltenen Hashfunktion SHA-256 zu einem 32 Byte langem Wert gehasht. Dieser Hash ist auf der Smartcard gespeichert und muss bei der Anmeldung zum anschließenden Vergleich des Trainers oder Sportlers übertragen werden. Der jeweilige Benutzer gibt in der grafischen Oberfläche sein Passwort im Klartext ein worauf die Programmlogik dieses hasht und es an die Smartcard überträgt. Initial ist die Karte mit je einem Hash für Trainer und Sportler zu füllen. Im Zuge der Vereinfachung sind diese Werte bereits statisch im Programmcode gefüllt. Um den Passworthash des Sportlers zu setzen ist das in Listing 1 dargestellte Kommando an die Smartcard zu senden.

² Bei einem Jahr werden nur die letzten zwei Ziffern hexadezimal abgespeichert.

```
/send 00 05 01 21 02 ... (Passworthash)
```

Listing 1: Setzen des Passwortes eines Sportlers in Form eines Kommandos

4.2 Datenstruktur

Die Übersicht der Modellklassen ist in ??? zu sehen.

Die Datenmodelle bilden die auf der Smartcard gespeicherten bzw. die vom Off-Card Teil verwendeten Datensätze wieder. Jede Modellklasse besitzt einen statischen Identifikator, einen Konstruktor, welcher alle Klassenvariablen setzt, und öffentliche lese- und Schreibmethoden der einzelnen Klassenvariablen. Weiterhin leitet jede Modellklasse von der abstrakten Klasse IModel ab, wodurch die Methoden toBytes und fromBytes überschrieben werden müssen und diese somit bei allen Klassen verfügbar sind.

Jeder Trainingsplan, hier Workoutplan, besitzt die folgenden Attribute:

- Erwärmungsphase
- Trainingsphase
- Abkühlungsphase
- Startdatum
- Enddatum

Jede Phase ist dabei ein Menge von Übungen bzw. Stages.

Eine Übung ist aus folgenden Attributen aufgebaut:

- Tagesnummer
- Geräteidentifikator
- Muskelgruppenidentifikator
- Übungsnummer
- Menge von Sätzen, hier Sets

Die Tagesnummer gibt an, welcher Trainingstag des wochenbezogenen Trainingsplanes die Übung zugeordnet ist. Da Zeichenketten nicht direkt auf der Smartcard gespeichert werden können, bzw. deren Byterepräsentation bei Beschreibungen die APDU Grenze von 256 Byte ohne Weiteres überschreiten können, werden zu Geräten und Muskelgruppen nicht deren Namen und Beschreibungen, sondern nur ein Identifikator gespeichert. Dieser Identifikator kann im Off-Card Teil zu einer Zeichenkette umgewandelt werden.

Einzelne Sätze beinhalten die Satznummer, das zu verwendende Gewicht und die Anzahl der Wiederholungen mit dem Gewicht.

Ein Datum wird durch die Klasse `MyDate` repräsentiert, welche je ein Byte für Jahr, Monat und Tag enthält.

Die Fortschrittsdarstellung erhebt deren Daten aus einer Menge von Fortschrittselementen mit den folgenden Attributen:

- Übungsnummer
- letzter Übungswert
- bester Übungswert
- schlechtester Übungswert

Ein Übungswert enthält dabei das verwendete Gewicht, die durchgeführten Wiederholungen und den Tag an welchem die Übung mit dem Gewicht und den Wiederholungen durchgeführt wurde.

4.3 On-Card Teil

Der On-Card Teil wird durch das Applet *Traincard* repräsentiert.

Im Klassendiagramm in der Abbildung 3 ist das Applet mit dessen Klassenvariablen und Methoden dargestellt. Das Applet befindet sich im Package `htwk.smartcard.traincard` und muss auf der verwendeten Smartcard installiert und gestartet werden.

Entsprechend der Oberklasse `javacard.framework.Applet`, existieren Methoden, welche die Installation, Ausführung und Deselektion behandeln. Die wichtigste Methode

ist dabei process, welche ankommende APDUs erhält, diese auswertet, die entsprechende Klassenmethode aufruft und deren Antwort-Bytes in Form einer Response APDU zurücksendet.

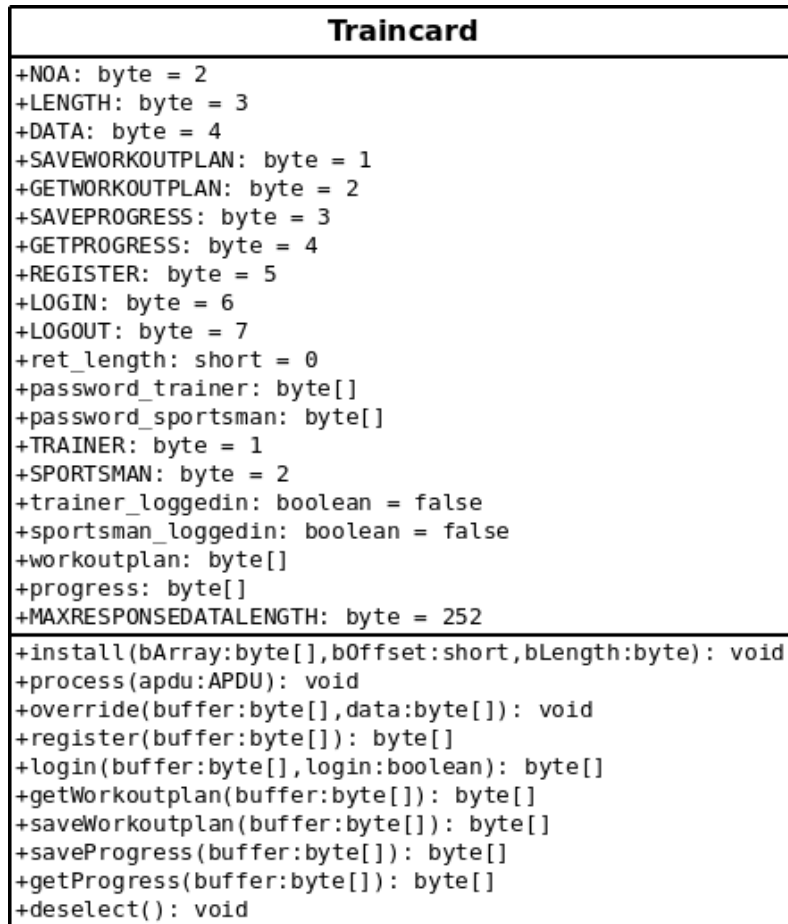


Abbildung 3: Klassendiagramm On-Card Teil

Das Instruktionsbyte wird mit den statischen Konstanten verglichen und anhand eines Treffers die entsprechende Methode aufgerufen. Jede dieser Methoden erhält den Puffer der APDU und gibt ein Byte Array für den Puffer der Response APDU zurück. Der Aufbau der Command und Response APDU ist in der Abbildung 2 dargestellt. Die darin dargestellten Data Bytes sind entsprechend der jeweiligen Instruktion gefüllt.

00 02 01 01 01 als APDU liest beispielsweise den Trainingsplan und sendet ihn zurück. Ist der Trainingsplan größer als MAXRESPONSEDATALENGTH Byte, muss mit 00 02 01 01 02 eine weitere APDU mit den nächsten Bytes des Trainingsplanes angefordert werden.

Lokale Byte Arrays der Methoden werden stets flüchtig mit der Methode make-

TransientByteArray der Klasse javacard.framework.JCSystem angelegt. Der Vergleich und das Kopieren von Byte Arrays erfolgt mit den Methoden arrayCompare und arrayCopy der Klasse javacard.framework.Util. Dadurch wird der EEPROM der Smartcard nicht durch Schreibvorgänge belastet und die Ausführungszeit verringert sich.

Der Trainingsplan und der Fortschritt wird in dynamischer Größe auf der Smartcard gespeichert und bei jedem Schreibvorgang neu erzeugt.

4.4 Off-Card Teil

Der Inhalt dieses Abschnittes bildet die Umsetzung des Off-Card-Teils. Dabei wird auf die graphische Oberfläche, die Logik zur Steuerung und zur Kommunikation mit der Smartcard eingegangen.

Wenn man die Off-Card-Anwendung startet, gelangt man in das Hauptmenü der Anwendung, welches in der Abbildung 5 dargestellt ist. Dieses Menü enthält fünf Buttons, welche jeweils zu einer Funktion des Programmes führt. Den gesamten Trainingsplan, den Tagesplan und den Trainingsfortschritt kann man sich ohne Eingabe eines Passwortes anschauen. Um die Trainingsergebnisse einzutragen oder den Trainingsplan zu ändern, ist ein Passwort nötig. Jede der Funktionen, sowie die Passwortabfrage, wird im folgenden genauer erläutert.

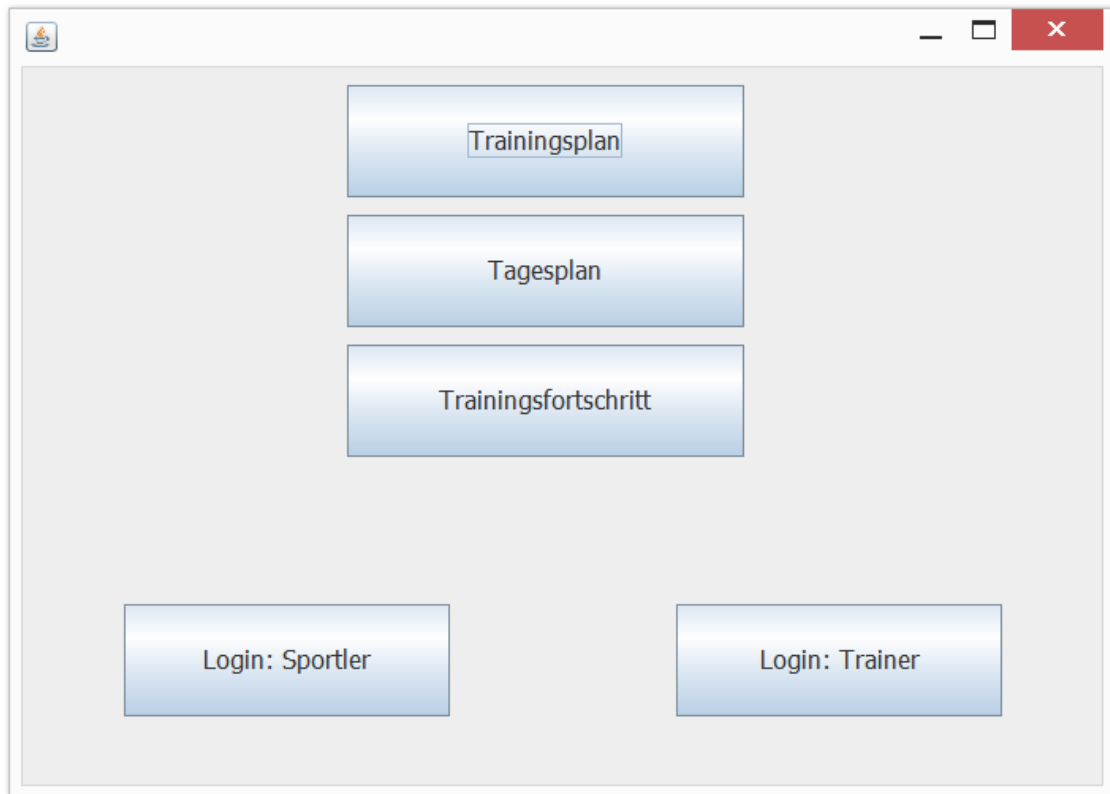
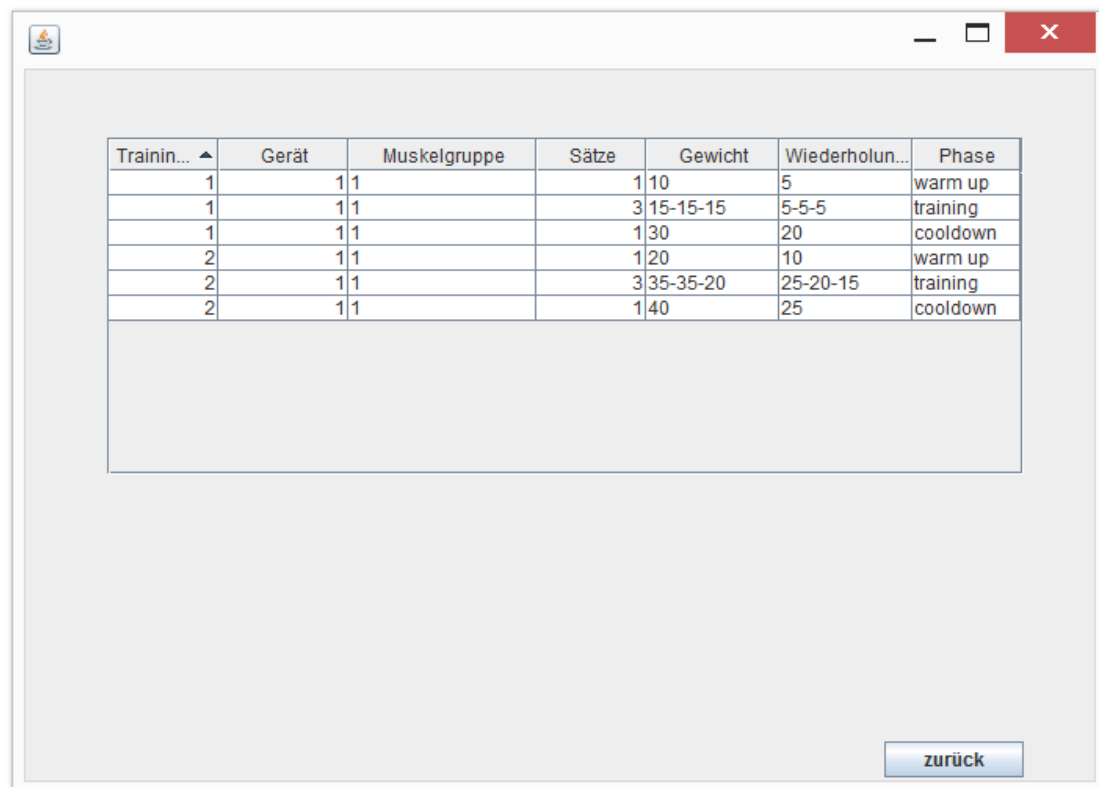


Abbildung 4: Hauptmenü

Trainingsplan

Über den Button Trainingsplan gelangt man in die nächste Ansicht, welche den gesamten aktuellen Trainingsplan darstellt (Abb. 5). Um den Trainingsplan zu laden, ruft das Programm die Funktion `CardInterface.getWorkoutplan()` aus dem `CardInterface` auf, welche als Rückgabewert den aktuellen Trainingsplan liefert. Dieser wird im Anschluss ausgelesen und in die Tabelle geschrieben. Wenn der Trainingsplan leer ist, ist folglich auch die Tabelle leer. Über den Zurück Button gelangt man, wie auch in allen anderen Funktion, zurück in das Hauptmenü.



Trainin...	Gerät	Muskelgruppe	Sätze	Gewicht	Wiederholun...	Phase
1	1	1	1	10	5	warm up
1	1	1	3	15-15-15	5-5-5	training
1	1	1	1	30	20	cooldown
2	1	1	1	20	10	warm up
2	1	1	3	35-35-20	25-20-15	training
2	1	1	1	40	25	cooldown

zurück

Abbildung 5: gesamter Trainingsplan

Tagesplan

Die Ansicht des Tagesplan ist ähnlich zur Anzeige des gesamten Trainingsplans, mit einer Einschränkung. Über die Combobox "Tag" kann der gewünschte Tag ausgewählt werden. Nach dieser Auswahl wird in der Tabelle, welche den Trainingsplan anzeigt, nur diejenigen Übungen eingetragen und dargestellt, welche an dem jeweiligen Tag ,laut Trainingsplan, stattfinden sollen. Dazu besitzt jede Stage in einem Workoutplan-Object ein Attribut day. Diese Ansicht ist in Abbildung 6 zu sehen.

Trainingstag	Muskelgruppe	Übung	Sätze	Gewicht	Wiederholun...	Phase
2	1	1	1	20	10	warm up
2	1	1	3	35-35-20	25-20-15	training
2	1	1	1	40	25	cooldown

Tag 2 ▼

zurück

Abbildung 6: Tagesplan

Login: Trainer

Um einen Trainingsplan mit den beiden Funktionen Trainingsplan und Tagesplan anzeigen zu können, ist es zuvor notwendig, einen Trainingsplan zu erzeugen. Dies ist möglich, wenn man sich als Trainer einloggt, wozu eine Passworteingabe nötig ist (Abb. 7).

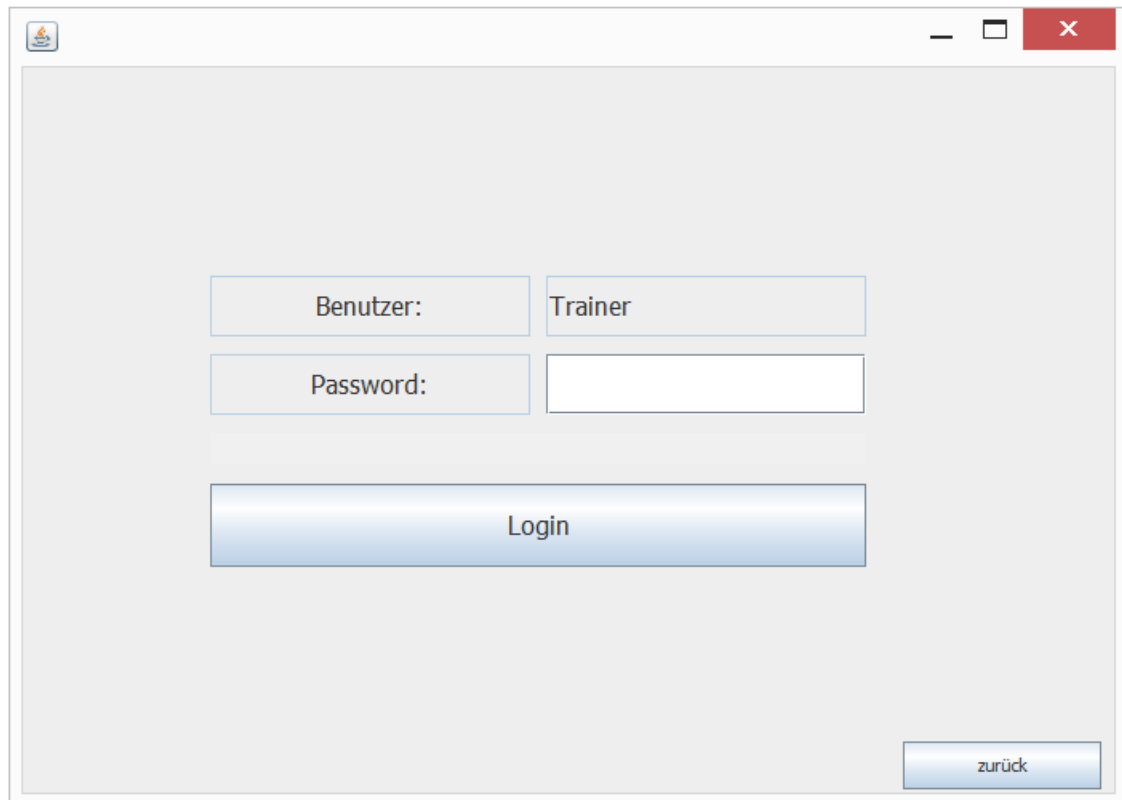


Abbildung 7: Eingabemenü des Trainers

Wenn man sein Passwort eingegeben hat und auf Login drückt, wird das Passwort gehashed und an die Smartcard mit dem Benutzertyp (Trainer oder Sportler) gesendet. Diese überprüft das Passwort auf Korrektheit und gibt als Antwort true oder false zurück. Wenn das Passwort korrekt ist, gelangt man in die Traineransicht und kann den aktuellen Trainingsplan verändern oder einen neuen anlegen. Die Ansicht ist in Abbildung 8 dargestellt. Über den Button neue Übung können weitere Zeilen zur Tabelle hinzugefügt werden. Zusätzlich zu den Übungen kann ein Start und Enddatum für diesen Trainingsplan festgelegt werden. Da auf der Karte keine Strings gespeichert werden können, muss man mit ID's arbeiten. Diese ID's kann man dann im Off-Card-Teil über eine Hashmap speziellen Strings zuweisen um beispielsweise die Muskelgruppe nicht, wie in der Abb. 8 zu sehen ist, als Zahl darstellen zu müssen. Beim speichern des Trainingsplans wird die Tabelle ausgelesen und die entsprechenden Werte einem neuen Workoutplan-Objekt zugewiesen, welches auf der Smartcard gespeichert wird. Zusätzlich dazu wird, auch ein neues Progressarray auf der Karte erzeugt, welche alle Stage-ID's für den neuen Plan enthält. Dies ist notwendig um den Fortschritt des Sportlers eintragen, speichern und auslesen zu können.

Trainingstag ▲	Gerät	Muskelgruppe	Sätze	Gewicht	Wiederholungen	Phase
1	1	1	1	10	5	warm up
1	1	1	3	15-15-15	5-5-5	training
1	1	1	1	30	20	cooldown
2	1	1	1	20	10	warm up
2	1	1	3	35-35-20	25-20-15	training
2	1	1	1	40	25	cooldown

neue Übung
 Startdatum: 1 1 2014
 Enddatum: 30 3 2014
 neuer Trainingsplan
 Trainingsplan speichern
 zurück

Abbildung 8: Eingabemenü des Trainers

Login: Sportler

Wenn nun ein Trainingsplan erzeugt und abgespeichert wurde, kann der Sportler seine Trainingsdaten bezüglich dieses Plans eintragen. Dazu ist es notwendig, dass er sich als Sportler einloggt. Wenn der Benutzer als Sportler angemeldet ist, gelangt er in die Sportleransicht, welche in Abb. 9 zu sehen ist. In dieser Ansicht werden alle Übungen, für den ausgewählten Tag, angezeigt. Die Auswahl geschieht nach dem gleichen Prinzip, welches auch in der Tagesplanansicht benutzt wird, siehe Abschnitt Tagesplan. Die Daten zu den jeweiligen Übungen, kann der Nutzer manuell eingeben, oder die vorgegebenen Trainingswerte über die Checkbox übernehmen lassen. Wenn die Daten eingetragen sind und gespeichert werden (Button: Trainingsdaten speichern), wird das Progressarray aktualisiert. Dazu wird es zunächst von der Karte gelesen, dann Werte die eingetragenen Werte überprüft, ob gegebenenfalls ein neuer bester oder schlechtester Wert erreicht wurde. Nach der Aktualisierung wird das Progressarray wieder auf der Karte abgelegt.

Gerät	Muskelgruppe	Sätze	heute	Gewicht	heute	Wiederholungen	heute	keine Abweichung
1	1	1	1	10	10	5	5	<input checked="" type="checkbox"/>
1	1	3	3	15-15-15	15-1...	5-5-5	5-5-5	<input checked="" type="checkbox"/>
1	1	1	1	30	30	20	20	<input checked="" type="checkbox"/>

Trainingsdaten speichern

1 ▼

zurück

Abbildung 9: Eingabemenü des Sportlers

Trainingsfortschritt

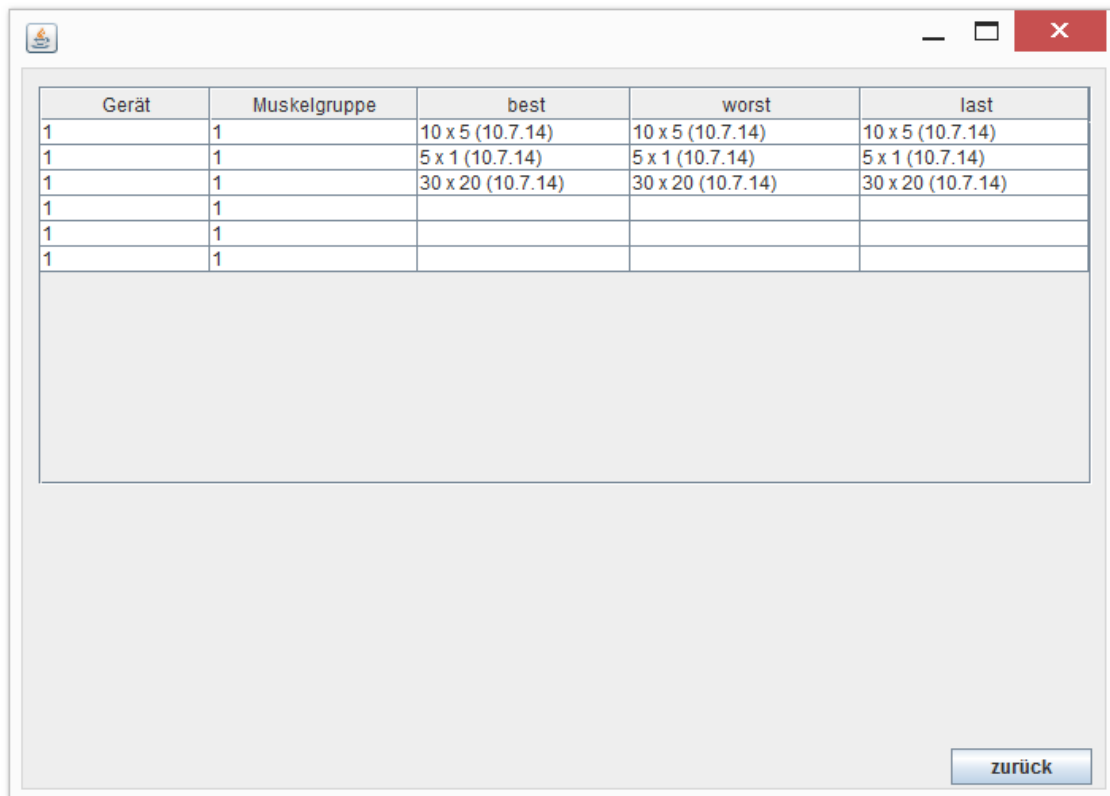
Über die Funktion Trainingsfortschritt, kann wie es der Name sagt, sich den aktuellen Trainingsfortschritt anzeigen lassen. Dazu wird von der Karte der aktuelle Trainingsplan und das Progressarray abgerufen. Dabei gibt es verschiedene Möglichkeiten. Zum einen kann noch kein Trainingsplan existieren. Wenn dieser Fall vorliegt, bleibt die Tabelle leer und das Progressarray existiert folglich auch nicht und muss nicht ausgelesen werden. Wenn ein Trainingsplan existiert, allerdings noch keine Trainingsdaten eingetragen wurden, wird nur der Trainingsplan abgebildet und die Felder der Trainingsdaten bleiben leer, siehe Abb. 10.

Gerät	Muskelgruppe	best	worst	last
1	1			
1	1			
1	1			
1	1			
1	1			
1	1			

zurück

Abbildung 10: Trainingsfortschritt ohne Einträge

Weiterhin existiert die Möglichkeit, dass im Progressarray zu einigen Übungen schon Trainingsdaten vorhanden sind, allerdings noch nicht zu allen. Dann werden, wie in der Abbildung 11 zu sehen ist, alle vorhandenen Werte eingetragen und die restlichen freigelassen. Die Spalte best bzw. worst gibt jeweils den besten bzw. schlechtesten Trainingswert an. Wenn ein neuer Trainingswert eingetragen wird, wird dieser immer in der Spalte last abgebildet. Um im späteren Verlauf nachvollziehen zu können, wann welcher Wert erreicht wurde, wird bezüglich der Daten noch ein Datum angegeben.



Gerät	Muskelgruppe	best	worst	last
1	1	10 x 5 (10.7.14)	10 x 5 (10.7.14)	10 x 5 (10.7.14)
1	1	5 x 1 (10.7.14)	5 x 1 (10.7.14)	5 x 1 (10.7.14)
1	1	30 x 20 (10.7.14)	30 x 20 (10.7.14)	30 x 20 (10.7.14)
1	1			
1	1			
1	1			

zurück

Abbildung 11: Trainingsfortschritt mit Einträgen

Klassendiagramm Off-Card Teil

In diesem Abschnitt wird das Klassendiagramm der Traingui beschrieben.

Die Traingui ist in die vier packages: model, main, gui und comm unterteilt. Im package main befindet sich der Programmstart, welche als erstes die MainView startet, um das Hauptmenü anzuzeigen. Von diesem aus, sind alle weiteren Funktionen erreichbar, welche in den vorrangegangenen Abschnitten erläutert wurden.

Die MainView befindet sich im package gui, welche alle Views (MainView, SportlerView, FortschrittView, TagesplanView, TrainerView und TrainingsplanView) und die Programmlogik (TableMethods und PasswordCheck) enthält. Die Logik dient zum Auslesen und Schreiben der Worktoutplan und Progress Elementen, welche in der Klasse TableMethods implementiert ist. Das Auslesen dient zum Befüllen der Tabellen. Vor dem schreiben neuer Elemente ist es notwendig, die Tabellen auszulesen und die entsprechenden Werte in das jeweilige Objekt zu übertragen. Die Zugriffsrechte auf die Smartcard werden auf Benutzerseite über die Klasse PasswordCheck geregelt, welche die eingegebenen Passwörter an die Smartcard überträgt und

diese dann die Eingabe aus korrektheit überprüft.

Jegliche Kommunikation der Traingui zur Smartcard, wird mit dem comm package realisiert. Dies beinhaltet dazu die Klassen CardInterface und CardHandler, welche alle benötigten Funktionsaufrufe, zum lesen und schreiben von und auf die Karte, enthalten.

Der Aufbau der Datentypen ist im package model deklariert. Dieses enthält ein Interface IModel und die jeweiligen Klassen für den einzelnen Elemente: Workoutplan, Progress, ProgressElement, MyDate, Stage, und Set. Die Klasse MyDate wird benötigt, da die Karte kein Date-Objekt speichern kann und für das Ablegen von einem Datum eine Byte-Kodierung benötigt.

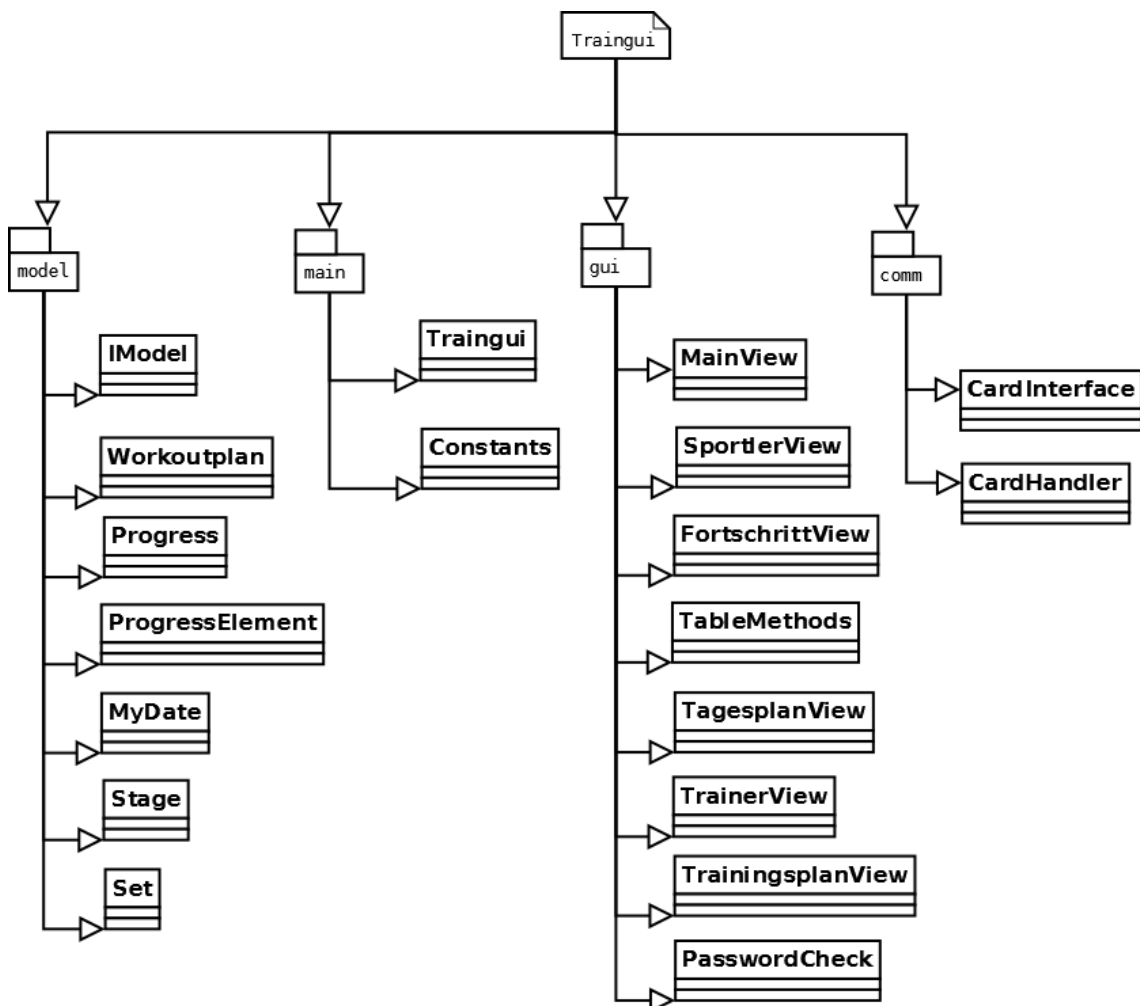


Abbildung 12: Klassendiagramm Off-Card Teil

5 Zusammenfassung und Ausblick

Der beschriebene Anwendungsfall wurde im Rahmen dieses Belegs analysiert und eine Lösung implementiert.

6 Quellenverzeichnis

Quellenverzeichnis

- [1] Prof. Dr. rer. nat. Petermann, Uwe: *Lehrmaterial zur Veranstaltung Smartcard-Programmierung im SS2014 an der HTWK-Leipzig*.
<https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/437649412/CourseNode/87246817112798>
intern abrufbar am 10.Juli 2014

- [2] Sun Microsystems, Inc.: *Java Card Applet Developer's Guide*
<http://www.oracle.com/technetwork/java/javacard/downloads/index.html>
abrufbar am 01.Juli 2014

- [3] International Organization for Standardization: *ISO 7816 - Identification cards – Integrated circuit cards* kostenpflichtig abrufbar unter <http://www.iso.org/>