Software Engineering Praktikum

CAB-Train

Software-Entwicklungspraktikum (SEP)
Sommersemester 2012

F e i n e n t w u r f

Auftraggeber
Technische Universität Braunschweig
Peter L. Reichertz Institut für Medizinische Informatik
Prof. Dr. Reinhold Haux
Mühlenpfordstraße 23
38106 Braunschweig

Betreuer: Markus Wagner, Thomas Franken

Auftragnehmer

Name	E-Mail-Adresse
Arnaud Chevrollier	arnaud.chevrollier@tu-bs.de
Christoph Harburg	c.harburg@tu-bs.de
Svenja Molitor	svenja_moli@hotmail.de
Bianca Oppermann	BiancaOppermann@gmx.de
Aisha Shnati	aishash777@googlemail.com
Virnadita Sjahran	virnadita@yahoo.de

Braunschweig, 27.06.2012

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
v0.1	11.06.2012	s. Auftragnehmer	offen	Erste Version, Texte von Kapitel 1 und 2 verfasst und in vorgegebenes Layout eingefügt
v0.2	15.06.2012	s. Auftragnehmer	offen	In erste Version wurde Kapitel 4 eingefügt
v0.3	20.06.2012	s. Auftragnehmer	offen	Tabellen und Klassendiagramme für Kapitel 3 erstellt
v0.4	24.06.2012	s. Auftragnehmer	offen	Texte zu Kapitel 3 verfasst und zusammen mit v0.3 in Version zwei eingebunden
v0.5	27.06.2012	s. Auftragnehmer	offen	Endgültige Formatierung

Inhaltsverzeichnis

1 Einleitung **6**

2 Erfüllung der Kriterien **7**

2.1 Musskriterien 7
2.1.1 /M10/ Erkennen eines Sensors 7
2.1.2 /M20/ Herstellen einer Verbindung zu einem erkannten Sensor 7
2.1.3 /M30/ Accountverwaltung 8
2.1.4 /M40/ Auswählen einer Übung 8
2.1.5 /M50/ Erklären der Übung 9
2.1.6 /M60/ Starten einer Übung 9
2.1.7 /M70/ Anzeigen des Übungsablaufs 9
2.1.8 /M80/ Speichern der Daten 10
2.1.9 /M90/ Beenden einer Übung 10
2.1.10 /M100/ Auswertung der Übung 11
2.1.11 /M110/ Programmstatus ändern 11
2.1.12 /M120/ Framework 12
2.2 Wunschkriterien 12
2.2.1 /W10/ Auswählen des zu verwendenden Eingabegerätes 12
2.2.2 /W20/ Anzeigen von erfolgreichem/fehlgeschlagenem Verbindungsaufbau 12
2.2.3 /W30/ Erklären einer Übung als Bild 13
2.3 Abgrenzungskriterien 13
2.3.1 /A10/ Ausführen auf mobilen Endgeräten 13
2.3.2 /A20/ Verwendung für nicht medizinische Bewegungsübungen 14

3 Implementierungsentwurf **15**

3.1 Gesamtsystem 15
3.2 Implementierung von Komponente WBB 16
3.2.1 Paketdiagramm/Klassendiagramm 17
3.2.2 Erläuterung 17
3.3 Implementierung von Komponente Position 19
3.3.1 Paketdiagramm/Klassendiagramm 19
3.3.2 Erläuterung 19

3.4 Implementierung von Komponente Controller 21
3.4.1 Paketdiagramm/Klassendiagramm 22
3.4.2 Erläuterung 22
3.5 Implementierung von Komponente Kommunikation 24
3.5.1 Paketdiagramm/Klassendiagramm 24
3.5.2 Erläuterung 24
3.6 Implementierung von Komponente GUI 26
3.6.1 Paketdiagramm/Klassendiagramm 26
3.6.2 Erläuterung 27
3.7 Implementierung von Komponente Übung 32
3.7.1 Paketdiagramm/Klassendiagramm 32
3.7.2 Erläuterung 33
3.8 Implementierung von Komponente Auswertung 34
3.8.1 Paketdiagramm/Klassendiagramm 34
3.8.2 Erläuterung 35
3.9 Implementierung von Komponente Speicher 37
3.9.1 Paketdiagramm/Klassendiagramm 37
3.9.2 Erläuterung 37

4 Datenmodell **39**
4.1 Diagramm 39
4.2 Erläuterung 39

Abbildungsverzeichnis

3.1 Komponentendiagramm: Komponentenspezifikation 15
3.2 Klassendiagramm: WBB 17
3.3 Klassendiagramm: Position 19
3.4 Klassendiagramm: Controller 22
3.5 Klassendiagramm: Kommunikation 24
3.6 Klassendiagramm: GUI1 26
3.7 Klassendiagramm: GUI2 27
3.8 Klassendiagramm: Übung 32
3.9 Klassendiagramm: Auswertung 34
3.10 Klassendiagramm: Speicher 37

4.1 Datenmodell 39

1 Einleitung

Die Software „CAB-Train Framework für computerassistiertes Bewegungstraining“ soll dazu dienen, Patienten aus dem Rehabereich eine Möglichkeit zu bieten von zu Hause aus, ohne geschultes Personal, einfach Übungen zur Verbesserung oder dem Erhalt des Gesundheitszustandes richtig auszuführen. Außerdem soll diese eine Frameworkfunktion beinhalten, um von Informatikern dazu verwendet zu werden, die einzelnen Klassen einfach zu erweitern, um somit weitere Sensoren und Übungen leicht und ohne großen Aufwand hinzufügen zu können. Das folgende Dokument dient dazu die Implementierungsdetails der Software zu beschreiben. Aus diesem Grund werden in Kapitel 2 die im Pflichtenheft aufgeführten Muss-, Wunsch- und Abgrenzungskriterien konkretisiert. Darauffolgend werden in Kapitel 3 die Komponenten des Grobentwurfs genauer beschrieben und Ihre Beziehungen untereinander mit Hilfe von Klassendiagrammen dargestellt. Zum Schluss wird in Kapitel 4 die Speicherung der dauerhaft erzeugten Daten in einem Datenmodell beschrieben.

2 Erfüllung der Kriterien

Im Nachfolgenden Kapitel wird detailliert beschrieben, wie die einzelnen Kriterien des Pflichtenheftes in der Software umgesetzt werden. Die gesamten Fenster für die Benutzeroberfläche wurden mit Hilfe der Graphikbibliothek Swing in Java implementiert. Zudem bezieht sich dieser Feinentwurf auf ein Framework mit einer Beispielimplementierung für das Wii-Balance-Board, weswegen sich viele Punkte hauptsächlich auf dieses beziehen.

2.1 Musskriterien

Die folgenden Punkte sind für das Projekt unerlässlich und müssen durch das Produkt erfüllt werden:

2.1.1 /M10/ Erkennen eines Sensors

Durch dieses Musskriterium wird die Möglichkeit zum Erkennen verschiedener Sensoren durch das Programm beschrieben. In diesem konkreten Fall wird das Wii-Balance-Board über eine Bluetooth-Schnittstelle angesprochen. Hierfür müssen beide Geräte (PC, Wii-Balance-Board) eingeschaltet sein. Eine weitere Voraussetzung ist, dass der Computer, an dem das Programm ausgeführt wird, über ein integriertes Bluetooth-Modul verfügt oder ein kompatibler Bluetooth-Adapter angeschlossen ist. Außerdem muss die Jar-Datei „Bluecove“ in der JRE System Library eingebunden sein. Diese Aspekte müssen bei allen anderen Sensoren immer beachtet werden und je nach Datenübertragung, wie z.B. über ein Kabel, geändert werden.

2.1.2 /M20/ Herstellen einer Verbindung zu einem erkannten Sensor

Wird das Musskriterium /M10/ erfüllt, kann nun eine Verbindung zu dem gewünschten Sensor aufgebaut werden. Nach dem Starten des Programmes wird automatisch versucht eine Verbindung zu einem in /M10/ erkannten Sensoren herzustellen. Aus diesem Grund ist es wichtig, dass der Benutzer immer nur ein Eingabegerät einschaltet. Nach Aufgabenstellung ist es bisher nur möglich, eine Verbindung zum Wii-Balance-Board aufzubauen. Durch die implementierte Methode „WiiBoardDiscoverer“ wird dieses zunächst erkannt. Daraufhin wird auf der Konsole

„Discovered“ und die Nachricht „Press the syncro-button“ angezeigt. Um endgültig die Verbindung herstellen zu können, muss die Synchronisations-Taste am Wii-Balance-Board, die sich im Batteriefach befindet, gedrückt werden. Dadurch wird die Datenübertragung via Bluetooth gestartet. Durch die Umsetzung des Wunschkriteriums /W10/ wird die Kommunikation zwischen Benutzer und Rechner von der Konsole auf die Benutzeroberfläche übertragen. Um die Frameworkfunktion zu erfüllen, wird eine übergeordnete Methode in der zugehörigen Vaterklasse implementiert, um den Verbindungsaufbau zu anderen Sensoren zu ermöglichen.

2.1.3 /M30/ Accountverwaltung

Dieses Musskriterium beschreibt den Anmeldevorgang eines Patienten für die für ihn zugeschnittenen Übungen. Die Übungen werden im Voraus von einem Arzt festgelegt, welche im Framework von einem Informatiker zu implementieren sind. Zudem hat der Informatiker die Aufgabe, einen Namen und ein Passwort für den zu behandelnden Patienten einzurichten. Der genauere Ablauf der Anmeldung für den Patienten wird nun beschrieben. Nach erfolgreichem Verbindungsaufbau zum eingeschalteten Sensor wird dem Benutzer am Anfang ein Fenster angezeigt, welches zwei Felder zur Eingabe besitzt. In diese beiden Felder hat der Benutzer seinen Namen und Passwort einzugeben, welche vom Programm per If-Bedingung auf Korrektheit überprüft werden. Sollten die eingegebenen Benutzerdaten falsch („false“) sein, wird das Fenster neu geladen und gibt dem Anwender die Möglichkeit, seine Eingabe zu wiederholen. Nach korrekter Eingabe der Benutzerdaten, entscheidet das Programm durch Case-Anweisungen, welche Übungen mit Hilfe eines Übungswahlfensters, welches in /M40/ weiter beschrieben wird, dem Benutzer angezeigt wird.

2.1.4 /M40/ Auswählen einer Übung

Das Auswählen einer Übung kann nur erfolgen, wenn eine korrekte Eingabe der Benutzerdaten in /M30/ erfolgt ist. Daraufhin erscheint die für den Patienten individuell zusammengestellte Liste von Übungen in einem Fenster. Diese Übungen sind per Mausklick auf einen Button auszuwählen, die als JToggleButton implementiert wurden. So wird es dem Patienten ermöglicht zu sehen, welche Übungen bereits ausgewählt wurden. Unmittelbar nach der Auswahl durch Anklicken des Buttons der Übung öffnet sich ein weiteres Fenster,dessen Auswahl wiederum durch eine Case-Anweisung geregelt ist, welches dem Benutzer die Übung erklärt. Diese Erklärungsfenster werden in /M50/ ausführlicher beschrieben. Wählt der Patient jedoch eine Übung aus, die noch nicht implementiert worden ist, wird ein Default-Fall ausgelöst, wonach dem Anwender ein Fenster angezeigt und in diesem erklärt wird, dass die ausgewählte Übung noch nicht implementiert wurde und was an dieser Stelle erscheinen würde, wenn die Übung hingegen bereits vorhanden wäre.

2.1.5 /M50/ Erklären der Übung

Die Erklärung einer Übung erscheint in einem in Swing implementierten Fenster, sobald der Benutzer eine gewünschte Übung in /M40/ ausgewählt hat. In diesem Erklärungsfenster wird dem Anwender in Textform detailliert beschrieben, wie der Patient seine ausgewählte Übung auszuführen hat. Je nach Bereitschaft kann der Benutzer danach auf den Start-Button innerhalb des Fensters klicken und damit die ausgewählte Übung starten. Dieses wird in /M60/ näher beschrieben. Mit diesem Ereignis schließen sich das Erklärungsfenster und das Übungswahlfenster automatisch, indem die Sichtbarkeit der Fenster auf „false“ gesetzt wird. Drückt man jedoch auf einen Start-Button innerhalb eines Erklärungsfensters, dessen Übung noch nicht implementiert wurde, erscheint ein Warnhinweisfenster, welches dem Benutzer mitteilt, dass die Übung noch nicht implementiert wurde und deshalb nicht gestartet werden kann. Betätigt der Benutzer den Ok-Button innerhalb dieses Vordefinierten Fensters wird dieses geschlossen.

2.1.6 /M60/ Starten einer Übung

Um eine Übung zu starten, ist es nötig, dass der Ablauf in /M50/ vom Benutzer korrekt durchlaufen wurde. Nachdem dies vollzogen wurde, wird die Sichtbarkeit des Übungsablauffensters auf „true“ gesetzt und damit dem Benutzer auf dem Bildschirm angezeigt. In diesem Fenster wird die auszuführende Übung angezeigt und der Benutzer muss die Übung durch Betätigen des Sensors (in diesem Fall durch Betreten des Wii-Balance Boards) starten. Das Wii-Balance-Board beginnt erst die Übung, wenn sich das Gesamtgewicht, welches mathematisch im Programm errechnet wird, über 20 Kilo befindet. Dies wird durch eine If-Bedingung bewerkstelligt. Von nun an werden vom Sensor permanent Daten an den Rechner geschickt, welche für die spätere Auswertung dienen. Dadurch wird gewährleistet, dass der Benutzer die volle Übungszeit zur Verfügung hat und somit die Übung vollständig ausführt. Da man sich in diesem Fall auf das Wii-Balance-Board bezieht, kann das Übungsfenster für andere Sensoren sich von diesem Beispiel unterscheiden. Dies ist dann im Framework von einem Informatiker zu implementieren.

2.1.7 /M70/ Anzeigen des Übungsablaufs

Die auszuführende Übung wird im dafür erstellten Fenster angezeigt und nach den in /M60/ gegebenen Kriterien begonnen. Das Fenster besitzt in dem konkreten Fall des Wii-Balance-Boards ein Fadenkreuz, einen schwarzen Punkt, rote Kreise und eine Zeitangabe in einer Leiste. Der schwarze Punkt wird mit Hilfe der vier Sensordaten, die vom Wii-Balance-Board an den Rechner geschickt und mit einer mathematischen Formel zu x- und y-Werten umgerechnet werden, in dem begrenzten Fadenkreuz angezeigt, welches eine x- und y-Achse darstellt. Befindet sich der Anwender mittig und im Gleichgewicht auf dem Wii-Balance-Board, wird der schwarze Punkt genau in der Mitte, das heißt an dem Ort wo sich die x- und y-Achse schneiden, nämlich wenn

x und y gleich 0 sind, dargestellt. Durch Gleichgewichtsverlagerung ändern sich die x- und y-Werte, da die Gewichtsverlagerung auf die einzelnen Sensoren sich nun ändert, wodurch der schwarze Punkt mit repaint() seine Stelle im Fadenkreuz ändert. Dadurch ist es möglich, dass der schwarze Punkt sich während der Bewegungen des Patienten flüssig bewegt und damit die aktuelle Lage genau nachverfolgt werden kann. Die roten Kreise stellen in diesem Beispiel die Teilaufgaben dar, die der Benutzer mit dem schwarzen Punkt erreichen muss, indem dieser sein Gleichgewicht auf dem Wii-Balance-Board verlagert. Es wird dem Anwender aber lediglich ein Kreis angezeigt. Die Kreise haben eine vorher bestimmte Position im Koordinatensystem, in dem noch ein Bereich existiert, den der Benutzer zu erreichen hat, aber nicht sieht. Sobald der Benutzer den Bereich des angezeigten Kreises erreicht hat, leuchtet der schwarze Punkt kurz grün, der erreichte Kreis wird verborgen, der nächste Kreis angezeigt und der grüne Punkt wieder schwarz gesetzt. Während einer Übung wird dem Anwender in einer Zeitleiste in einem Fenster die Dauer der Übung angezeigt.

2.1.8 /M80/ Speichern der Daten

Dieses Musskriterium beschreibt den Vorgang der Speicherung der Trainingsdaten, die während und nach einer ausgeführten Übung empfangen werden. Die zu speichernden Daten sind die vom Programm errechneten x- und y-Koordinaten, die dazu dienen, dem Benutzer seine Schwerpunktsverlagerung während einer Übung anzuzeigen, sowie die Zeitangaben, die ein Benutzer benötigt, eine Teilaufgabe des ausgewählten Trainings zu erreichen. Die Speicherung dieser Daten erfolgt über einen Stream, in dem die erhaltenen Double-Werte eingelesen, zu String-Werten umgewandelt und dann durch eine Schreibfunktion in eine Textdatei geschrieben werden. Zudem wird nach der erfolgreichen Anmeldung in /M30/ der Name als String und die errechneten Werte der endgültigen Auswertung in eine Textdatei gespeichert. Der Name und die errechneten Werte der Auswertung erfolgen nicht über einen Stream, sondern werden direkt in die Textdatei geschrieben, da sie nicht so oft wie die Sensordaten in dem Programm vorkommen.

2.1.9 /M90/ Beenden einer Übung

Das Beenden einer Übung wird mit der vollständigen Durchführung oder einer Unterbrechung von mindestens zwei Minuten einer Übung realisiert. Falls der Anwender eine Übung vollständig durchgeführt hat, das heißt, dass die vorgeschriebene Zeit, in der die Übung durchgeführt werden soll, abgelaufen ist, wird das Fenster des Übungsablaufes geschlossen und die Zusendung der Daten vom Wii-BalanceBoard an den Rechner beendet. Mit der Beendigung der Zusendung der Daten, wird auch die Speicherung, die in /M80/ beschrieben wird, beendet.

2.1.10 /M100/ Auswertung der Übung

Dieses Musskriterium beschreibt unteranderem die Umrechnung der Trainingsdaten zu einer geeigneten Auswertung. Diese Umrechnung erfolgt nur dann, wenn die Übung wie in /M90/ beschrieben nach vollständiger Durchführung beendet wurde. Diese errechneten Auswertungsdaten geben letzten Endes die Leistung des Patienten graphisch als Ampel oder Tabelle aus. Im Folgenden wird detailliert beschrieben, wie die Umrechnung der Trainingsdaten stattfindet.

Während der Patient eine Übung ausführt, werden Daten, wie z.B. die Zeit, die der Patient braucht um jeden der vorgegeben Punkte zu erreichen, einzeln gespeichert. Diese Zeitangabe wird dem Benutzer später in einer Tabelle angezeigt, jedoch nicht auf der Benutzeroberfläche, sondern in einem dafür vorgesehenen Dokument. In dieser Tabelle wird aufgelistet, wie viel Zeit der Anwender für Teilaufgabe 1,2 usw. benötigt hat. Zusätzlich werden diese Zeiten zusammengerechnet und daraus ein einziger Mittelwert gebildet. Außerdem wird gespeichert, wie viele Teilaufgaben der Benutzer in der vorgegebenen Zeit ausgeführt hat. Wenn man den zuvor berechneten Mittelwert in Relation mit dem gerade angegeben Wert setzt, erhält man einen Wert, der dazu dient, im Ampelsystem aufzuzeigen, wie gut eine Übung ausgeführt wurde. Mit Hilfe dieser beiden Mittelwerte stellt man sicher, dass nicht nur die Geschwindigkeit der Ausführung der Teilaufgaben eine Rolle für die Auswertung spielt, sondern auch die Genauigkeit mit der die einzelnen Teilaufgaben durchgeführt wurden. Wurde eine Übung nicht durch das vollständige Ausführen einer Übung beendet, wird dem Benutzer trotzdem das Auswertungsfenster angezeigt. Jedoch wird in diesem Fenster keine Ampel angezeigt, sondern lediglich ein Hinweis an den Benutzer, dass keine Daten für die Auswertung zur Verfügung stehen und dass die Übung erneut ausgeführt werden muss.

2.1.11 /M110/ Programmstatus ändern

Sobald dem Benutzer das Auswertungsfenster /M100/ angezeigt wurde, besteht für diesen die Möglichkeit das Programm zu beenden oder es neu zu initialisieren. Dies wird dem Nutzer durch die Implementierung dreier Buttons ermöglicht. Betätigt der Anwender den Beenden-Button, wird das aktuelle Fenster geschlossen und das gesamte Programm beendet. Wichtig hierbei ist, dass das Programm an dieser Stelle nur durch das Klicken dieses Buttons beendet werden kann und nicht durch das Standard gegebene x am oberen Bildschirmrand, da dessen standardisierte Eigenschaft aufgehoben ist. Der zweite von drei Buttons mit der Beschriftung „Weiter zur Sensorwahl“ führt den Benutzer nach Betätigen des Buttons zurück zum ersten Anzeigefenster, also dem Sensorwahlmenü. Mit diesem Button wird das Programm erneut von vorne ausgeführt. Dies wird durch das Aufrufen des Controllers und damit das erneute Ausführen der Main-Methode implementiert. Der dritte und letzte Button- „Weiter zur Übungswahl“ ermöglicht es dem Benutzer direkt in das Übungswahlfenster zu wechseln, ohne vorher erneut einen Sensor auszuwählen, zu warten bis eine Verbindung aufgebaut wurde und seine Accountdaten anzugeben. Damit

wird es dem Anwender erleichtert, mehrere Aufgaben mit dem gleichen Sensor, nacheinander auszuführen.

2.1.12 /M120/ Framework

Da sich die Implementierung eines Frameworks auf das gesamte Projekt bezieht, kann man dies nicht unabhängig von den anderen Kriterien erläutern. Aus diesem Grund wurde in den zuvor beschrieben Kriterien teilweise schon auf die Umsetzung des Frameworks innerhalb des Produkts eingegangen. Da das Framework selbst nur die Strukturen und Hierarchien eines Softwareprodukts beschreibt, wird an dieser Stelle nicht weiter auf die Implementierung des Frameworks eingegangen. Eine gute Darstellung des Frameworks und seiner Implementierung erfolgt im Kapitel 3, bei den einzelnen Komponenten und den dazugehörigen Klassen.

2.2 Wunschkriterien

Die folgenden Punkte beschreiben die Wunschkriterien, die zur Verbesserung des Projekts umgesetzt wurden:

2.2.1 /W10/ Auswählen des zu verwendenden Eingabegerätes

Zur Ausarbeitung des Frameworks und zu Erleichterung der Anwendung des Programmes soll das Projekt über ein Auswahlfenster für die verschiedenen Sensoren verfügen. Dieses wird mit Hilfe der Komponente GUI auf dem Bildschirm angezeigt. Außerdem wird es durch die Grafikbibliothek Swing in Java implementiert. Die Oberfläche besteht aus einem einfachen Fenster, in dem sich eine JComboBox, mit der man die Sensoren auswählen kann, befindet. Innerhalb dieser JComboBox werden die auszuwählenden Sensoren, zur besseren Übersicht, als Bild angezeigt. Weiterhin verfügt das Fenster über einen JButton „Weiter“. Wird dieser angeklickt, folgt /M20/ und der Verbindungsaufbau wird automatisch gestartet.

2.2.2 /W20/ Anzeigen von erfolgreichem/fehlgeschlagenem Verbindungsaufbau

Nachdem der Benutzer wie in /W20/ beschrieben einen Sensorausgewählt hat und den Ok-Button betätigt hat wird versucht eine Verbindung zum gewählten Sensor herzustellen. In diesem Moment wird ein neues JFrame geöffnet mit einem Ladebalken, der solange angezeigt wird, bis der Verbindungsaufbau erfolgt ist. Dieser bewegt sich von links nach rechts. Der Ladebalken wird mithilfe der JProgressBar implementiert. Bei fehlgeschlagener Verbindung hält der Ladebalken an, ein Text „Die Verbindung konnte nicht hergestellt werden“ erscheint auf dem Bildschirm und der Benutzer erhält nun die Wahl einen von drei Buttons zu klicken. Entweder

kann der Anwender den JButton „Erneut versuchen“ klicken, woraufhin versucht wird, erneut eine Verbindung zum gewählten Sensor aufzubauen, oder er kann den Button „Zur Sensorwahl“ betätigen, woraufhin der Benutzer zurück zum Sensorwahlmenü gelangt und nun einen neuen/anderen Sensor wählen kann. Betätigt der Nutzer hingegen den letzten Button „Beenden“ wird das aktuelle Fenster geschlossen und das Programm beendet. Bei erfolgreichem Verbindungsaufbau erscheint im aktuellen Fenster der Text „Die Verbindung wurde hergestellt“ und der Anwender kann durch das Klicken des Weiter-Buttons in die Accountverwaltung gelangen. Außerdem wird das aktuelle Fenster daraufhin automatisch geschlossen.

2.2.3 /W30/ Erklären einer Übung als Bild

Wie in /M50/ genau beschrieben, wird die Übung zum einen textuell erklärt, zum anderen soll die Übung durch die Erfüllung dieses Wunschkriteriums auch mit Hilfe eines Bildes, welches durch ImageIcon(get.Class().getResource()), in das Erklärungsfenster eingefügt wurde, illustriert werden. Dieses wird innerhalb des in /M50/ beschriebenen Erklärungsfensters unterhalb des Erklärungstextes angezeigt und gibt dem Benutzer so eine Vorstellung über die durchzuführende Übung. Falls die ausgewählte Übung noch nicht implementiert wurde, wird unterhalb des Defaulttextes ein Defaultbild, in diesem Falle das Wii-Balance Board, angezeigt. Ist der Zugriff auf dieses Bild oder ein Anderes nicht möglich bleibt der Bereich unterhalb des Erklärungstextes leer.

2.3 Abgrenzungskriterien

Die folgenden Punkte beschreiben, warum diese Kriterien nicht umgesetzt werden können bzw. eine Umsetzung dieser Kriterien für das Produkt nicht sinnvoll ist:

2.3.1 /A10/ Ausführen auf mobilen Endgeräten

Das zu entwickelnde Programm soll lediglich mit einem Wii-BalanceBoard (und später weiteren Sensoren) und einem Rechner laufen. Es soll nicht als Applikation auf einem mobilen Endgerät, wie zum Beispiel einem Mobiltelefon, funktionieren. Es wurde sich dazu entschieden, dies nicht zu realisieren, da zum Einen das Programm für ältere Menschen gedacht ist, die meistens kein Handy besitzen, das ein solches Programm ausführen kann, und zum Anderen wäre es nötig, eine zusätzliche Software für mobile Endgeräte zu entwickeln. Außerdem ist für die Software ein Wii-BalanceBoard (oder ein anderer Sensor) nötig, welches nur bedingt für den mobilen Transport geeignet ist.

2.3.2 /A20/ Verwendung für nicht medizinische Bewegungsübungen

Dieses Produkt wurde ausschließlich für die Benutzung medizinischer Bewegungsübungen entwickelt. Die Struktur und der Ablauf der Software zeigen dies deutlich. So wurde darauf geachtet, eine leichte Benutzung des Produkts, gerade für ältere Menschen, zu gewährleisten. Außerdem wurde eine Accountverwaltung implementiert, damit der Patient nur die für ihn vorgesehenen und sinnvollen Übungen auswählen und somit ausführen kann. Die Implementierung dieser Übungen erfolgt zwar über einen Informatiker, aber nach der Absprache und den Anweisungen eines bzw. des behandelnden Arztes. Aus diesem Grund ist das Produkt nicht für andere Zwecke geeignet.

3 Implementierungsentwurf

Das folgende Kapitel wird in zwei Abschnitte unterteilt und dient dazu, die im Produkt verwendeten Klassen und Bibliotheken zu dokumentieren. In Abschnitt 3.1 wird zunächst einmal eine Übersicht über das Gesamtsystem mittels eines Komponentendiagrammes gegeben. Dies wird dazu verwendet, das Zusammenwirken der einzelnen Komponenten zu beschreiben. Folgend wird in Abschnitt 3.2 auf die einzelnen Komponenten eingegangen. Dabei wird die Implementierung dieser durch Klassendiagramme beschrieben, welche anschließend mit Hilfe von Tabellen näher erläutert werden. Klassen, die in mehreren Komponenten vorhanden sind, werden als Tabelle nur einmal im gesamten Implementierungsentwurf aufgeführt, um Redundanz zu vermeiden. Dies gilt insbesondere für die Klasse WBB.

3.1 Gesamtsystem

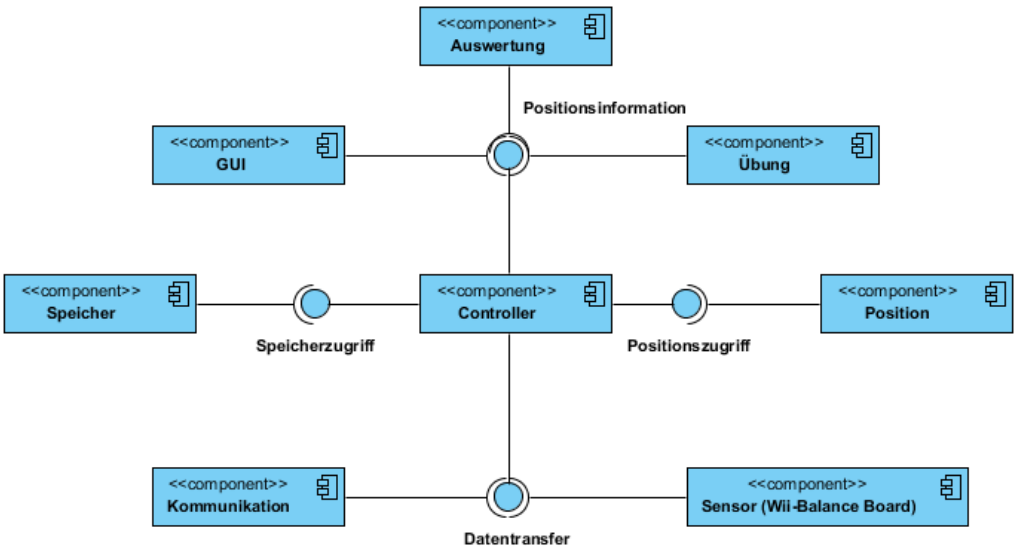

Abbildung 3.1: Komponentendiagramm: Komponentenspezifikation

Abbildung 3.1 zeigt die Komponenten und deren Interaktion untereinander. Die Implementierung der einzelnen Komponenten wird im Unterkapitel 3.2 beschrieben.

Die Komponente Controller stellt für alle anderen Komponenten Schnittstellen bereit und dient somit als Vermittler zwischen diesen. Die Komponenten GUI, Auswertung und Übung greifen indirekt über die Schnittstelle Positionsinformation via Controller auf die aktuelle Gewichtsverteilung des Benutzers zu. Aufgrund der dadurch erhaltenen Daten wird über die Verbindung zwischen GUI und Übung dem Benutzer seine Gleichgewichtsverlagerung visuell mitgeteilt. Außerdem kann durch die empfangenen Daten die Berechnung für die Auswertung stattfinden, die dem Benutzer wiederum über eine Verbindung zwischen GUI und Auswertung dargestellt wird. Über die Schnittstellen Speicherzugriff und Positionszugriff werden die Positionsdaten wiederum über den Controller beim Übungsablauf direkt an den Arbeitsspeicher übertragen. Danach werden über dieselben die Auswertungsdaten mit den zuvor temporär gespeicherten Daten an die Festplatte übergeben.

3.2 Implementierung von Komponente WBB

Die Implementierung der Komponente WBB erfolgt in Java. Dafür wird die Bibliothek WiiBoardSimple verwendet. In der Klasse WBB sind die wichtigsten Methoden, die gebraucht werden, um mit dem Wii-Balance-Board arbeiten zu können, vorgegeben. Daher ist diese eine abstrakte Klasse, sodass die weiteren Klassen die Methoden zwar benutzen, aber nicht verändern können.

3.2.1 Paketdiagramm/Klassendiagramm

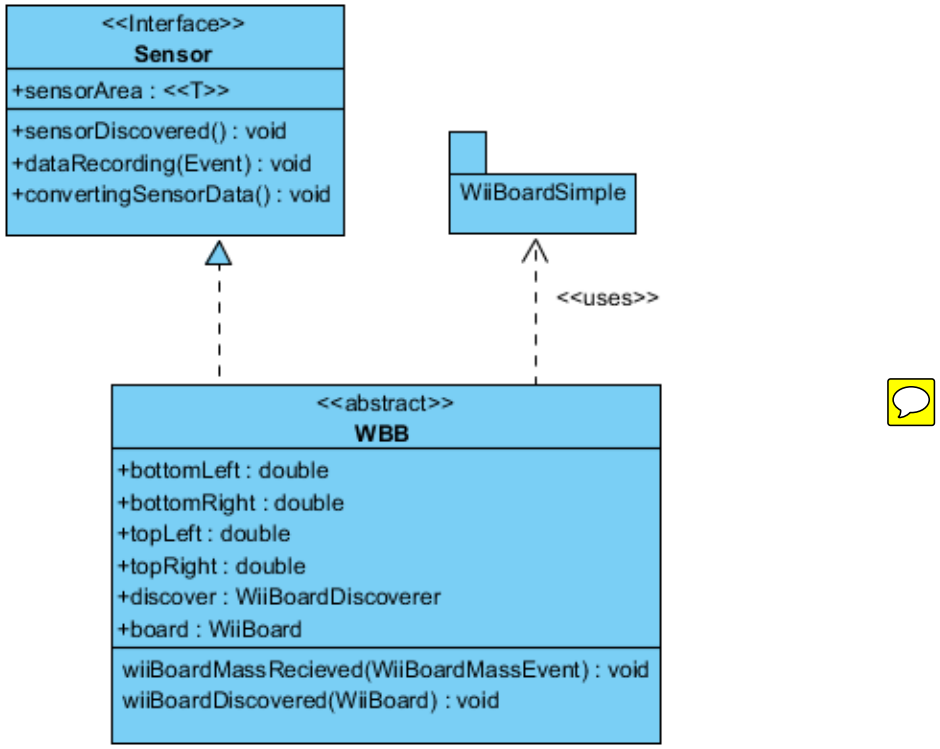

Abbildung 3.2: Klassendiagramm: WBB

3.2.2 Erläuterung

Es ist nicht erforderlich auf die zwei Methoden in der Klasse WBB näher einzugehen, da diese bereits in bestimmten Komponenten erläutert wurden. Um den Frameworkgedanken realisieren zu können, greift die Klasse WBB auf die Schnittstelle Sensor zu. Wenn ein Informatiker dieses Interface implementiert, besitzt er alle notwendigen Methoden, um eine einwandfreie Datenübertragung zu einem beliebigen Sensor zu ermöglichen. Dies wird durch die Methoden sensorDiscovered() und dataRecording() realisiert. Die Methode convertingSensorData() ist für eine generische Datenumwandlung zuständig. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- sensor: bezeichnet den Sensor mit dem eine Verbindung hergestellt wurde
- sensorArea: enthält die verschiedenen Sensoren zu denen eine Verbindung aufgebaut werden kann als Array
- bottomLeft: enthält die vom unteren linken Sensor aufgenommen Daten als DOUBLE-Werte

- bottomRight: enthält die vom unteren rechten Sensor aufgenommen Daten als DOUBLE-Werte
- topLeft: enthält die vom oberen linken Sensor aufgenommen Daten als DOUBLE-Werte
- topRight: enthält die vom oberen rechten Sensor aufgenommen Daten als DOUBLE-Werte
- discover: bezeichnet die gefundenen Sensoren
- board: bezeichnet das Wii-Balance-Board mit dem eine Verbindung hergestellt wurde

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Sensor	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***		***Name der Klasse***	*dauerhaft*
	1	sensor-Discovered()	herstellen der Verbindung zum gefundenen Sensor			
	2	data-Recording()	aufnehmen der empfangenen Sensordaten			
	3	converting-SensorData()	umwandeln der empfangenen Sensordaten in generische Werte			

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
WBB	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	bottomLeft: double topRight: double bottomRight: double topLeft: double board: WiiBoard	***Name der Klasse***	*dauerhaft*
	1	wiiBoard-Discovered()	herstellen der Verbindung zum gefundenen Wii-Balance-Board		Sensor	nein
	2	wiiBoard-Mass Received()	umwandeln der empfangenen Sensordaten in DOUBLE-Werte			

3.3 Implementierung von Komponente Position

Die Implementierung der Komponente erfolgt in Java. Für die Implementierung dieser Komponente wird die Wii-Bibliothek WiiBoardSimple verwendet. Die Komponente Position interagiert mit der Komponente WBB, um Sensordaten empfangen zu können. Nach der Umwandlung der Daten in geeignete Werte, werden diese an die Komponente Übung weitergeleitet, in der sie genutzt werden, um dem Benutzer seine Bewegungen bzw. allgemein Sensordaten anzuzeigen. Zusätzlich werden die umgewandelten Sensordaten an die Komponente Auswertung weitergeleitet, um zusammen mit den Daten aus der Komponente Übung, ausgewertet werden zu können.

3.3.1 Paketdiagramm/Klassendiagramm

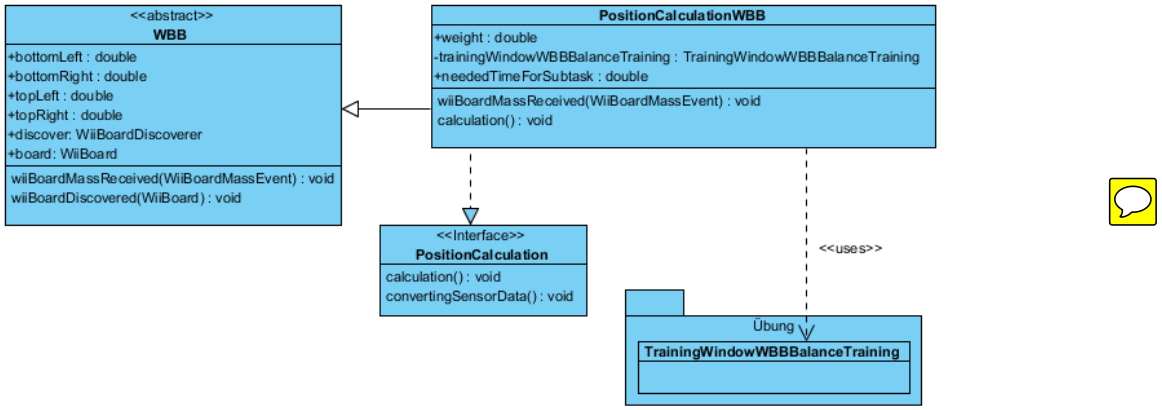

Abbildung 3.3: Klassendiagramm: Position

3.3.2 Erläuterung

Nachdem eine Verbindung zum Wii-Balance-Board aufgebaut wurde, der Benutzer eine Übung gestartet und sich auf das Wii-Balance-Board gestellt hat, werden vom Board 24Byte Daten an die Klasse PositionCalculationWBB übertragen. Diese 24Byte Daten sind aufgeteilt, in vier 6Byte Daten, die jeweils von einem der 4 Sensoren auf dem Wii-Balance-Board aufgenommen werden. Die Klasse PositionCalculationWBB wandelt nun für jeden der 4 Sensoren, diese 6 Byte Daten in double-Werte um. Nach der Umwandlung können diese Werte dann an die Komponenten, die sie benötigen, weitergeleitet werden. Die Klasse PositionCalculationWBB implementiert vom Interface PositionCalculation die Methoden, die notwendig sind, um die Sensordaten zu empfangen und weiter zu verarbeiten. Das Interface PostionCalculation dient dazu Informatikern zu zeigen, welche Methoden bei verschiedenen Sensoren implementiert werden müssen, um spezifische Sensordaten aufnehmen zu können und zu verarbeiten. Dieses Interface ist also für die Umsetzung des Frameworks notwendig. Wichtig ist, dass PositionCalculationWBB gleichzeitig

von der abstrakten Klasse WBB erbt. In dieser ist nämlich die Methode zum Aufnehmen bzw. Verarbeiten der 24Byte Daten, die das Wii-Balance-Board übermittelt, bereits implementiert. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- bottomLeft: enthält die vom unteren linken Sensor aufgenommen Daten als DOUBLE-Werte
- bottomRight: enthält die vom unteren rechten Sensor aufgenommen Daten als DOUBLE-Werte
- topLeft: enthält die vom oberen linken Sensor aufgenommen Daten als DOUBLE-Werte
- topRight: enthält die vom oberen rechten Sensor aufgenommen Daten als DOUBLE-Werte
- weight: bezeichnet das Gesamtgewicht, dass sich auf allen 4 Sensoren zugleich befindet
- neededTimeForSubtask: bezeichnet die Zeit, die für eine Teilaufgabe benötigt wird
- discover: bezeichnet die gefundenen Sensoren
- board: bezeichnet das Wii-Balance-Board mit dem eine Verbindung hergestellt wurde

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Position-Calculation	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***		***Name der Klasse***	*dauerhaft*
	1	converting-SensorData()	umwandeln der empfangenen Sensordaten in generische Werte		Sensor	ja
	2	calculation()	umrechnen der Daten in benötigte Werte für weitere Verwendung			

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Position-Calculation-WBB</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>wiiBoard-Mass Received()</td><td>umwandeln der empfangenen Sensordaten in DOUBLE-Werte</td></tr>
<tr><td>2</td><td>calculation()</td><td>umrechnen der Daten in X- und Y-Werte</td></tr>
</table>
</td>
<td>bottomLeft: double
bottomRight: double
topLeft: double
topRight: double</td>
<td>
<table>
<tr><th>Name der Klasse</th><td>dauerhaft</td></tr>
<tr><td>WBB</td><td>ja</td></tr>
</table>
</td>
</tr>
</table>

3.4 Implementierung von Komponente Controller

Die Implementierung der Komponente erfolgt in Java. Für die Implementierung dieser Komponente werden keine zusätzlichen Bibliotheken verwendet. Die Komponente Controller interagiert mit der Komponente GUI, um die Verarbeitung der Accountinformationen auf der Benutzeroberfläche anzuzeigen. Außerdem arbeitet der Controller mit allen anderen Komponenten zusammen, da er das Bindeglied zwischen allen Komponenten darstellt und für die Steuerung dieser zuständig ist. Das heißt, der Controller gibt an, welche Komponente zu welchem Zeitpunkt ausgeführt werden soll.

3.4.1 Paketdiagramm/Klassendiagramm

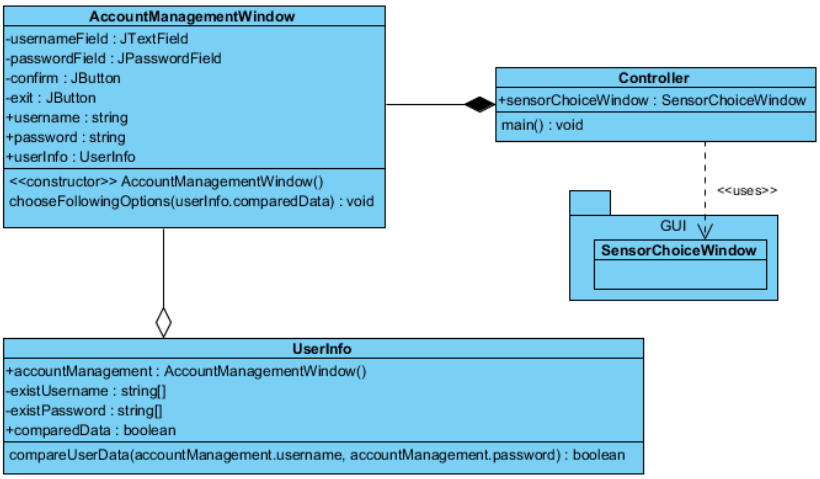

Abbildung 3.4: Klassendiagramm: Controller

3.4.2 Erläuterung

Sobald der Benutzer einen Sensor ausgewählt hat und eine Verbindung zu diesem hergestellt wurde, muss der Benutzer seine Nutzerdaten eingeben, um die für ihn bestimmten Übungen auswählen zu können. Die Klasse AccountManagmentWindow, die vom Interface Visualization abgeleitet ist, dient dazu, die Accountverwaltung auf der Benutzeroberfläche sichtbar zu machen. Innerhalb dieses Fensters befinden sich zwei Textfelder. In eines muss der Benutzer seinen Namen eingeben, in das andere das dazugehörige Passwort. Diese Eingaben werden innerhalb der Klasse AccountManagementWindow als Strings abgespeichert und an die Klasse UserInfo weitergeleitet. Innerhalb der Klasse UserInfo, sind die existierenden Benutzernamen und die dazu passenden Passwörter als Arrays gespeichert. Die Speicherung als Arrays ist für dieses Projekt von großem Nutzen, da so der Frameworkgedanke ein weiteres Mal umgesetzt wurde. Durch die Implementierung der vorhandenen Benutzerdaten als Strings, können leicht weitere Benutzer hinzugefügt werden. Die Klasse UserInfo überprüft mittels der Methode compareUserData(accountManagement.username, accountManagement.password), ob die zuvor eingegebenen Daten vom Benutzer korrekt sind und in dieser Kombination existieren. Je nachdem, welchen Rückgabewert diese Methode hat, überprüft chooseFollowingOptions(userInfo.comparedData) welche Aktion als nächstes ausgeführt werden soll. Die Accountverwaltung ist eine Klasse der Komponente Controller, da über diese kontrolliert wird, ob für einen Benutzer Übungen vorhanden sind und wenn ja, um welche Übungen es sich handelt. Damit steuert die Accountverwaltung indirekt den gesamten Folgeablauf des Programmes.

Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- username: bezeichnet den vom Benutzer, in das dafür vorgesehene Textfeld, eingegebenen Benutzernamen
- password: : bezeichnet das vom Benutzer, in das dafür vorgesehene Textfeld, eingegebenen Passwort
- userInfo: Attribut der Klasse UserInfo
- comparedData: bezeichnet den boolean-Wert, der zurückgegeben wird, anhand der eingegebenen Benutzerdaten

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Controller	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	sensorChoice-Window: SensorChoice-Window	***Name der Klasse***	*dauerhaft*
	1	main()	führt nacheinander die Elemente des Projekts aus		SensorChoice-Window	ja

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Account-Managment-Window	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	username: String password: String userInfo: UserInfo	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		UserInfo	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion		Visualization	nein
					Controller	ja
	3	choose-Following-Options()	feststellen, welche Folgefunktion ausgelöst werden soll		LinkConnection-SuccededWindow	ja

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>UserInfo</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>compareUser-Data()</td><td>vergleicht die eingegebenen Benutzerdaten mit den vorher Gespeicherten</td></tr>
</table>
</td>
<td>account-Management: Account-Management-Window

comparedData: boolean</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>Account-Management-Window</td><td>nein</td></tr>
</table>
</td>
</tr>
</table>

3.5 Implementierung von Komponente Kommunikation

Die Komponente wurde in Java implementiert. Sie nutzt die Klassenbibliothek WiiBoardSimple und die Vaterklasse Sensor aus der Komponente WBB, um den Verbindungsaufbau zwischen dem Sensor und dem Rechner zu ermöglichen.

3.5.1 Paketdiagramm/Klassendiagramm

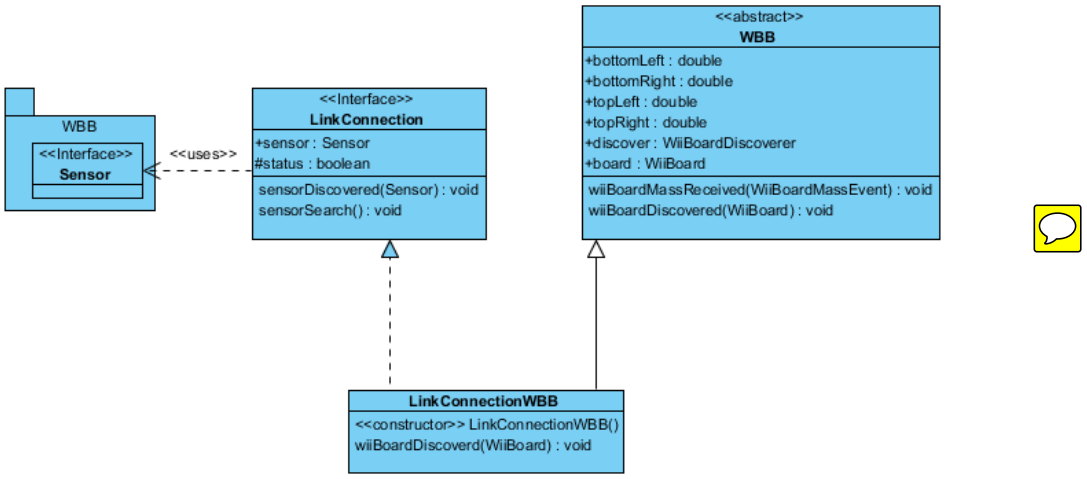

Abbildung 3.5: Klassendiagramm: Kommunikation

3.5.2 Erläuterung

Die Klasse LinkConnectionWBB sucht nach einem eingeschalteten Wii-Balance-Board und versucht mittels der Methode wiiBoardDiscovered(WiiBoard) eine Verbindung zu dem gefundenenen Board herzustellen. Außerdem erbt LinkConnectionWBB diese Methode von seiner ab-

strakten Vaterklasse WBB. Zusätzlich implementiert diese das Interface LinkConnection. Damit werden die Methoden angezeigt, die zum Suchen eines Sensors und für den Verbindungsaufbau nötig sind. Auf diese Weise können Informatiker später direkt erkennen, welche Methoden für den Verbindungsaufbau mit verschiedenen Sensoren gebraucht werden. Dieses Interface ist also für die Umsetzung des Frameworks notwendig. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- sensor: bezeichnet den Sensor mit dem eine Verbindung hergestellt wurde
- bottomLeft: enthält die vom unteren linken Sensor aufgenommen Daten als DOUBLE-Werte
- bottomRight: enthält die vom unteren rechten Sensor aufgenommen Daten als DOUBLE-Werte
- topLeft: enthält die vom oberen linken Sensor aufgenommen Daten als DOUBLE-Werte
- topRight: enthält die vom oberen rechten Sensor aufgenommen Daten als DOUBLE-Werte
- discover: bezeichnet die gefundenen Sensoren
- board: bezeichnet das Wii-Balance-Board mit dem eine Verbindung hergestellt wurde

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Link-Connection</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>sensor-Discovered()</td><td>herstellen einer Verbindung zum gefundenen Sensor</td></tr>
<tr><td>2</td><td>sensor-Search()</td><td>suchen von verfügbaren Sensoren</td></tr>
</table>
</td>
<td>sensor: Sensor</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>Sensor</td><td>ja</td></tr>
</table>
</td>
</tr>
</table>

Beteiligte Klasse	*Aufgabe*			*Notwendige Attribute*	*Notwendige Kommunikationspartner*	
Link-Connection-WBB	*Aufgaben-ID*	*Name der Aufgabe*	*Beschreibung*	board: WiiBoard	*Name der Klasse*	*dauerhaft*
	1	wiiBoard-Discovered()	herstellen der Verbindung zum gefundenen Wii-Balance-Board		WBB	ja

3.6 Implementierung von Komponente GUI

Die Komponente wurde in Java implementiert. Sie nutzt dabei verschiedene Interfaces der Graphikbibliothek Swing(Java). Dazu gehören z.B. JFrame, JLabel oder JPanel. GUI wird benötigt, um diverse Fenster mit den dazugehörigen Funktionen für das Projekt anzuzeigen. Das GUI arbeitet somit mit allen weiteren Komponenten zusammen.

3.6.1 Paketdiagramm/Klassendiagramm

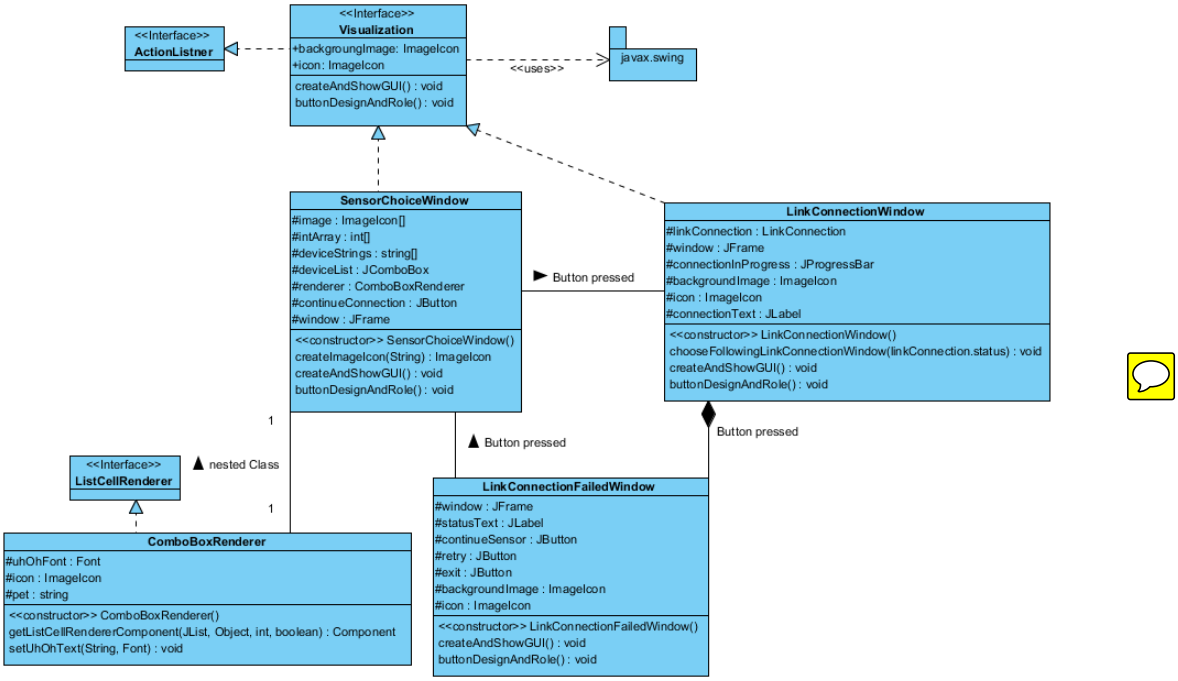

Abbildung 3.6: Klassendiagramm: GUI1

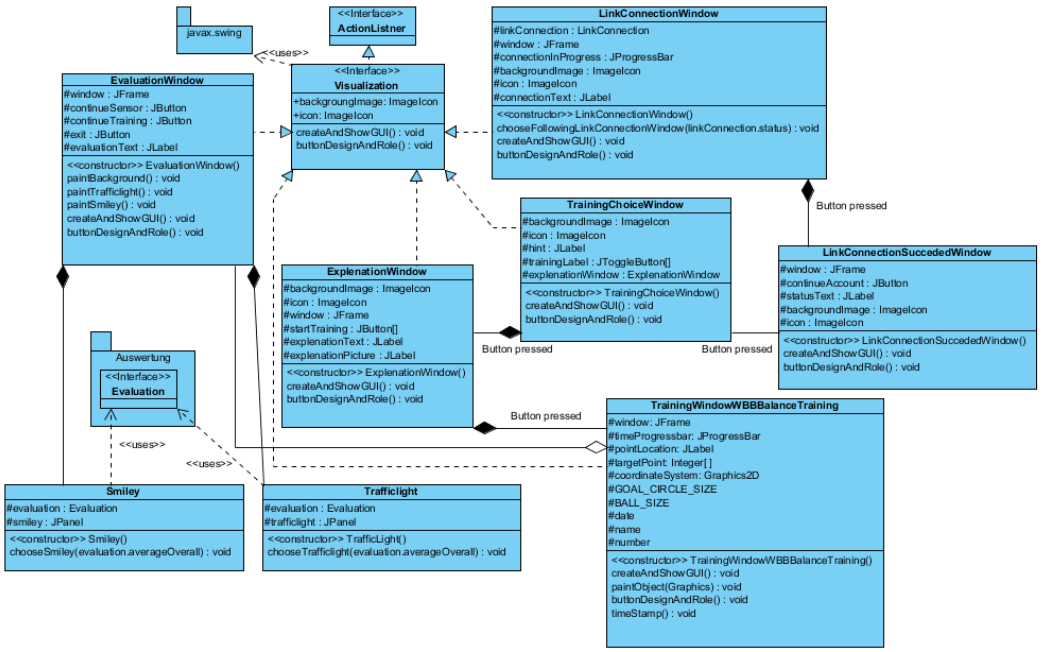

Abbildung 3.7: Klassendiagramm: GUI2

3.6.2 Erläuterung

An oberster Stelle steht hierbei das Interface Visualization, das die Bibliothek Swing(java) importiert und dadurch alles, was benötigt wird, aus dieser übernehmen kann. Da Visualization Vaterklasse von allen anderen Klassen in der Komponente GUI ist, haben diese auch Zugriff auf die Bibliothek. Auf diese Weise können beispielsweise Fenster (JFrame) hergestellt und Buttons (JButton) hinzufügt werden. Zusätzlich wird das Interface ActionListener benötigt, da dieser dafür zuständig ist, dass eine bestimmte Funktion ausgeführt wird, wenn man auf einen beliebigen Button drückt. Die Klasse SensorChoiceWindow dient zum Auswählen der Sensoren. Diese besitzt eine innere Klasse ComboBoxRenderer, die wiederum das Interface ListCellRenderer benötigt, damit die Auswahl der Sensoren durch ein Ausklappmenü angezeigt werden kann. Sobald man sich für einen Sensor entschieden hat, betätigt man den Weiter-Button, sodass sich das LinkConnectionWindow für den Verbindnugsaufbau öffnet. Hier muss ebenso auf ein Button gedrückt werden, um eine Verbindung herzustellen. War der Verbindungsaufbau erfolglos, schließt das Fenster der Klasse LinkConnectionWindow automatisch und es erscheint eine neue Anzeige der Klasse LinkConnectionFailedWindow mit drei integrierten Buttons. War der Verbindungsaufbau jedoch erfolgreich, öffnet sich das Fenster der Klasse LinkConnectionSucceededWindow mit einem Button. Durch das Betätigen des Buttons gelangt der Benutzer zum Übungswahlfenster. Innerhalb der Klasse TrainingChoiceWindow wird es dem Benutzer ermöglicht eine beliebige Übung auswählen, sodass das Erklärungsfenster mit zugehörigem Erklärungstext erscheint. Nach dem Betätigen des Buttons im ExplanationWindow kann die Übung beginnen und ausgeführt werden, die mittels der Klasse TrainingWindowWBBBalanceTraining

dargestellt wird. Direkt nach der Beendigung des Trainings taucht das EvaluationWindow für die Auswertung der Übung auf. Diese enthält in ihrem Fenster einen Smiley und ein Ampelsystem, welche durch die Klassen Smiley und TrafficLight realisiert werden. Mit Hilfe der Methoden chooseTrafficLight(evaluation.averageOverall) und chooseSmiley(evaluation.averageOverall) werden, in Abhängigkeit der Übungsausführung, die Graphiken gezeichnet.

Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- backgroundImage: Attribut der Klasse ImageIcon
- icon: enthält die Graphik, die später als Hintergrundbild verwendet wird

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Visualization	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	backgroundImage: ImageIcon icon: ImageIcon	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		ActionListener	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion			

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
SensorChoice-Window	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	intArray: Integer[] images: ImageIcon renderer: ComboBox-Renderer	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		ComboBox-Renderer	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion		Visualization	nein
					Controller	ja
	3	createImage-Icon()	Bilder werden den zugehörigen Sensoren zugewiesen			

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Link-Connection-Window	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	backgroundImage: ImageIcon icon: ImageIcon linkConnection: LinkConnection	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		LinkConnection	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion		Visualization	nein
					SensorChoice-Window	nein
	3	choose-FollowingLink-Connection-Window()	feststellen, welche Folgefunktion ausgelöst werden soll			

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Link-Connection-Failed-Window	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	backgroundImage: ImageIcon icon: ImageIcon	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		LinkConnection-Window	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion		Visualization	nein

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
Link-Connection-Succeded-Window	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	backgroundImage: ImageIcon icon: ImageIcon	***Name der Klasse***	*dauerhaft*
	1	createAnd-ShowGUI()	erzeugen des Anzeigefensters		LinkConnection-Window	nein
	2	buttonDesign-AndRole()	gestalten der Button und deren Funktion		Visualization	nein

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Training-Choice-Window</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>createAnd-ShowGUI()</td><td>erzeugen des Anzeigefensters</td></tr>
<tr><td>2</td><td>buttonDesign-AndRole()</td><td>gestalten der Button und deren Funktion</td></tr>
</table>
</td>
<td>backgroundImage: ImageIcon

icon: ImageIcon</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>Accont-Management</td><td>nein</td></tr>
<tr><td>Visualization</td><td>nein</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Explanation-Window</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>createAnd-ShowGUI()</td><td>erzeugen des Anzeigefensters</td></tr>
<tr><td>2</td><td>buttonDesign-AndRole()</td><td>gestalten der Button und deren Funktion</td></tr>
</table>
</td>
<td>backgroundImage: ImageIcon

icon: ImageIcon</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>TrainingChoice-Window</td><td>ja</td></tr>
<tr><td>Visualization</td><td>nein</td></tr>
</table>
</td>
</tr>
</table>

Beteiligte Klasse	***Aufgabe***	***Notwendige Attribute***	***Notwendige Kommunikationspartner***
Evaluation-Window	***Aufgaben-ID*** / ***Name der Aufgabe*** / ***Beschreibung*** 1 / createAnd-ShowGUI() / erzeugen des Anzeigefensters 2 / buttonDesign-AndRole() / gestalten der Button und deren Funktion 3 / paint-Backround() / festlegen der Hintergrund-farbe je nach Bewertung 4 / paintTraffic-Light() / modellieren der Ampel 5 / paintSmiley() / modellieren des Smileys	evaluation: Evaluation	***Name der Klasse*** / *dauerhaft* TrainingWindow-WBBBalance-Training / nein Visualization / nein

Beteiligte Klasse	***Aufgabe***	***Notwendige Attribute***	***Notwendige Kommunikationspartner***
TrafficLight	***Aufgaben-ID*** / ***Name der Aufgabe*** / ***Beschreibung*** 1 / chooseTraffic-Light() / je nach Bewertung wird die passende Ampel ausgewählt	evaluation: Evalution	***Name der Klasse*** / *dauerhaft* Evaluation / ja Visualization / nein

Beteiligte Klasse	***Aufgabe***	***Notwendige Attribute***	***Notwendige Kommunikationspartner***
Smiley	***Aufgaben-ID*** / ***Name der Aufgabe*** / ***Beschreibung*** 1 / chooseSmiley() / je nach Bewertung wird der passende Smiley ausgewählt	evaluation: Evalution	***Name der Klasse*** / *dauerhaft* Evaluation / ja Visualization / nein

3.7 Implementierung von Komponente Übung

Die Implementierung der Komponente Übung erfolgt in Java. Dazu wird die Graphikbibliothek Swing(Java) verwendet. In diesem Fall wurde ein Fenster für eine Gleichgewichtsübung für das Wii-Balance-Board implementiert. Hierzu interagiert die Komponente Übung mit den Komponenten GUI und Position. Außerdem ist eine Interaktion mit der Komponente Speicher für die darauffolgende Auswertung erforderlich, weil Daten während der Übung gespeichert werden.

3.7.1 Paketdiagramm/Klassendiagramm

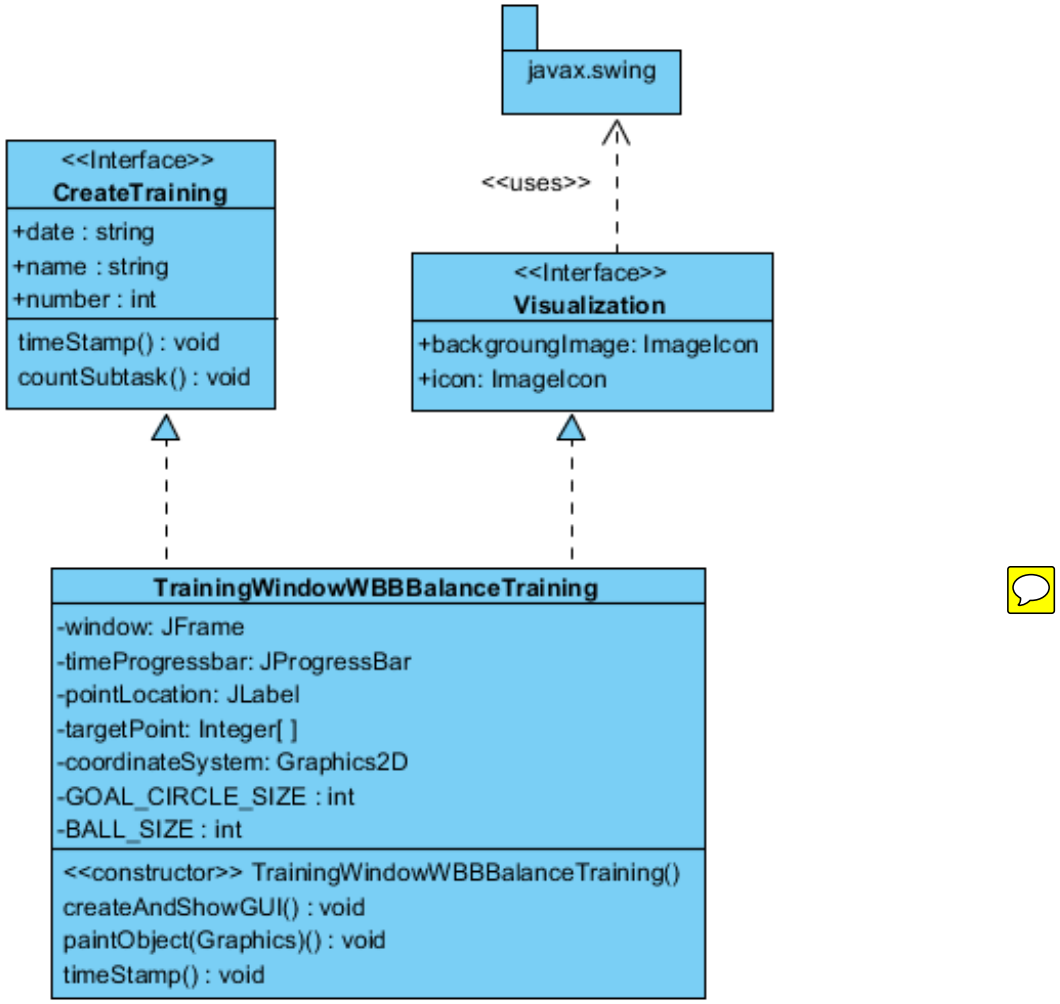

Abbildung 3.8: Klassendiagramm: Übung

3.7.2 Erläuterung

Bei dieser Übung wird das Interface Visualization implementiert, um mit der Methode createAndShowGUI() ein Fenster, in dem sich die benötigten Elemente für die Gleichgewichtsübung befinden, die mit der Methode paintObject(Graphics) erzeugt werden, an die Benutzeroberfläche zu übergeben und dem Benutzer anzuzeigen. In dem Fenster befinden sich eine Zeitanzeige, ein Fadenkreuz, ein Punkt, der die aktuelle Gleichgewichtsverteilung anzeigt, und rote Kreise, die die nacheinander auftretenden Teilziele anzeigen. Es wird das Interface CreateTraining implementiert, damit jede von Informatikern hinzugefügte Übung entweder über die Zeit oder einen Zähler identifiziert und bewertet werden kann. Mit der Methode timeStamp() wird die Zeit zwischen den einzelnen erreichten Teilzielen berechnet, das heißt, dass die fortlaufende Zeit beim Erreichen eines Teilzieles zurückgesetzt und für die Auswertung gespeichert wird. Die Methode countSubtask() zählt die in der vorgegebenen Gesamtzeit der Übung erreichten Teilziele. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- date: enthält das aktuelle Datum als String
- name: enthält den Namen der aktuellen Übung als String
- number: enthält die Identifikationsnummer der aktuellen Übung
- backgroundImage: Attribut der Klasse ImageIcon
- icon: enthält die Graphik, die später als Hintergrundbild verwendet wird

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr><td>Create-Training</td><td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>timeStamp()</td><td>Zeitzähler für eine Teilaufgabe</td></tr>
<tr><td>2</td><td>countSubtask()</td><td>Zähler für Teilaufgaben</td></tr>
</table>
</td><td></td><td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr><td>Training-WindowWBB-Balance-Training</td><td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>createAnd-ShowGUI()</td><td>erzeugen des Anzeigefensters</td></tr>
<tr><td>2</td><td>paintObject()</td><td>darstellen eines Koordinaten-systems, der zu erreichenden Punkte und des Positions-punktes</td></tr>
</table>
</td><td></td><td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>CreateTraining</td><td>ja</td></tr>
<tr><td>Visualization</td><td>ja</td></tr>
<tr><td>Explanation-Window</td><td>nein</td></tr>
</table>
</td></tr>
</table>

3.8 Implementierung von Komponente Auswertung

Diese Implementierung erfolgt ebenfalls in Java. Sie interagiert mit den Komponenten WBB und Speicher, da sich die benötigten Daten für die Auswertung im Speicher befinden. Die Klasse EvaluationWBB wird speziell für die Berechnung der Auswertung des Wii-Balance-Boards genutzt, die die Methoden aus dem Interface Evaluation implementiert.

3.8.1 Paketdiagramm/Klassendiagramm

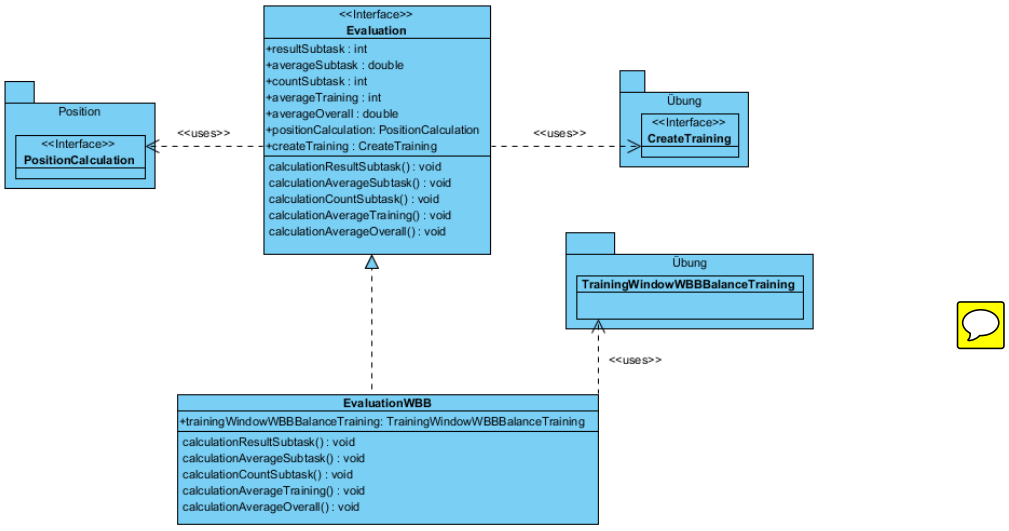

Abbildung 3.9: Klassendiagramm: Auswertung

3.8.2 Erläuterung

Das Interface Evaluation ist für die Berechnung der Auswertung zuständig. Durch die Methode calculationCountSubtask() im Interface werden die erreichten Teilziele gezählt und die gesamten Teilziele, die in der vorgegebenen Übungszeit erreicht wurden, zu einem Mittelwert für die, für den Anwender angezeigte Auswertung, berechnet. Zudem wird mit der Methode calculationAverageTraining() ein Durchschnitt der komplett durchgeführten gleichen Übungen berechnet. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- resultSubask: enthält Bewertung für die einzelnen Teilaufgaben
- averageSubtask: enthält die durchschnittliche Bewertung der Teilaufgaben
- countSubtask: enthält die Anzahl der ausgeführten Teilaufgaben
- averageTraining: enthält die zeitabhängige Bewertung
- averageOverall: bezeichnet die Gesamtbewertung der Übung
- positionCalculation: Attribut der Klasse PositionCalculation
- createTraining: Attribut der Klasse CreateTraining
- trainingWindowWBBBalanceTraining: Attribut der Klasse TrainingWindowWBBBalanceTraining
- bottomLeft: enthält die vom unteren linken Sensor aufgenommen Daten als DOUBLE-Werte
- bottomRight: enthält die vom unteren rechten Sensor aufgenommen Daten als DOUBLE-Werte
- topLeft: enthält die vom oberen linken Sensor aufgenommen Daten als DOUBLE-Werte
- topRight: enthält die vom oberen rechten Sensor aufgenommen Daten als DOUBLE-Werte
- discover: bezeichnet die gefundenen Sensoren
- board: bezeichnet das Wii-Balance-Board mit dem eine Verbindung hergestellt wurde

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Evaluation</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>calculation-Result-Subtask()</td><td>berechnen der Bewertung für die erste Teilaufgabe</td></tr>
<tr><td>2</td><td>calculation-Average-Subtask()</td><td>berechnen der durchschnitt-lichen Bewertung der Teilaufgaben</td></tr>
<tr><td>3</td><td>calculation-Count-Subtask()</td><td>zählen der ausgeführten Teilaufgaben</td></tr>
<tr><td>4</td><td>calculation-Average-Training()</td><td>berechnen der zeitabhängigen Bewertung</td></tr>
<tr><td>5</td><td>calculation-Average-Overall()</td><td>berechnen der Gesamtbe-wertung</td></tr>
</table>
</td>
<td>position-Calculation: Position-Calculation

createTraining: CreateTraining:</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>Position-Calculation</td><td>ja</td></tr>
<tr><td>CreateTraining</td><td>nein</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><th>Beteiligte Klasse</th><th>Aufgabe</th><th>Notwendige Attribute</th><th>Notwendige Kommunikationspartner</th></tr>
<tr>
<td>Evaluation-WBB</td>
<td>
<table>
<tr><th>Aufgaben-ID</th><th>Name der Aufgabe</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>calculation-Result-Subtask()</td><td>berechnen der Bewertung für die erste Teilaufgabe</td></tr>
<tr><td>2</td><td>calculation-Average-Subtask()</td><td>berechnen der durchschnitt-lichen Bewertung der Teilaufgaben</td></tr>
<tr><td>3</td><td>calculation-Count-Subtask()</td><td>zählen der ausgeführten Teilaufgaben</td></tr>
<tr><td>4</td><td>calculation-Average-Training()</td><td>berechnen der zeitabhängigen Bewertung</td></tr>
<tr><td>5</td><td>calculation-Average-Overall()</td><td>berechnen der Gesamtbe-wertung</td></tr>
</table>
</td>
<td>position-CalculationWBB: Position-CalculationWBB

trainingWindow-WBBBalance-Training: TrainingWindow-WBBBalance-Training</td>
<td>
<table>
<tr><th>Name der Klasse</th><th>dauerhaft</th></tr>
<tr><td>Position-CalculationWBB</td><td>ja</td></tr>
<tr><td>TrainingWindow-WBBBalance-Training</td><td>nein</td></tr>
</table>
</td>
</tr>
</table>

3.9 Implementierung von Komponente Speicher

Die Komponente Speicher wurde mit Java implementiert. Sie nutzt verschiedene Interfaces der Bibliothek io (Java). Dazu gehören z.B. BufferedWriter, File und FileWriter. Der Speicher interagiert mit den Komponenten Übung, Auswertung und Controller, um die aus diesen Komponenten hervorgehenden Daten zu speichern.

3.9.1 Paketdiagramm/Klassendiagramm

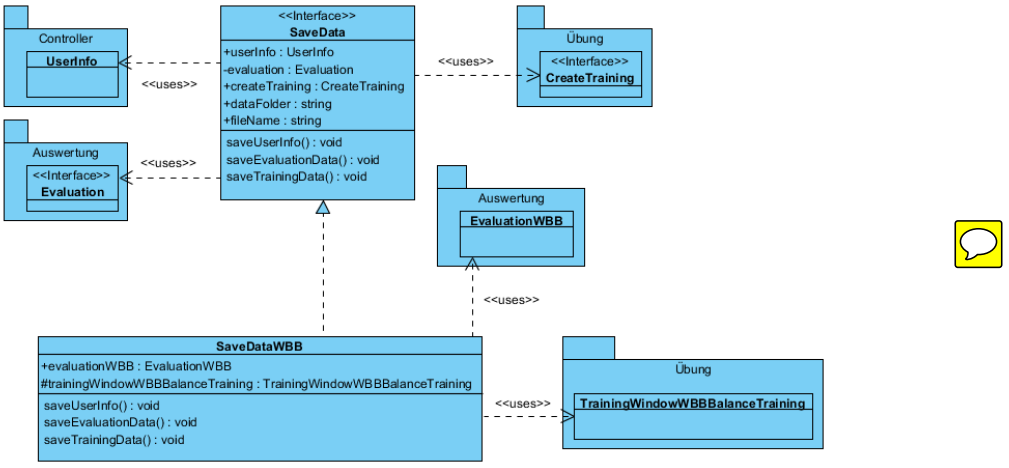

Abbildung 3.10: Klassendiagramm: Speicher

3.9.2 Erläuterung

Das Interface SaveData speichert sowohl die benötigten Userdaten, als auch die Trainingsdaten in Verbindung mit den Auswertungsdaten, die der Arzt für die Einschätzung der Qualität der durchgeführten Übungen braucht. Diese werden mittels der Methoden saveUserInfo(), saveEvaluationData() und saveTrainingData() in einem vordefinierten Layout innerhalb einem dafür vorgesehenen Ordner gespeichert. Um das Speichern für weitere Sensoren generisch zu halten, wird zusätzlich zum spezifischen Speichern SaveDataWBB das Interface SaveData implementiert. Die Attributnamen sind meist selbsterklärend gewählt, jedoch werden im Folgenden die public Attribute zusätzlich erläutert.

- userInfo: Attribut der Klasse UserInfo
- createTraining: Attribut der Klasse CreateTraining
- dataFolder: enthält den Dateipfad zu dem Ordner in dem Daten gespeichert werden sollen
- filename: enthält Namen des Ordners in dem Daten gespeichert werden sollen

- evaluationWBB: Attribut der Klasse EvaluationWBB

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
SaveData	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	userInfo : UserInfo evaluation: Evaluation createTraining: CreateTraining	***Name der Klasse***	*dauerhaft*
	1	saveUser-Info()	abspeichern der aktuellen Benutzerdaten		UserInfo	nein
	2	save-Evaluation-Data()	abspeichern der berechneten Bewertungs-daten		Evaluation	ja
	3	saveTraining-Data()	abspeichern der Übungsbezeich-nung, ÜbungsID und des Datums		CreateTraining	nein

Beteiligte Klasse	***Aufgabe***			***Notwendige Attribute***	***Notwendige Kommunikationspartner***	
SaveData-WBB	***Aufgaben-ID***	***Name der Aufgabe***	***Beschreibung***	userInfo : UserInfo evaluationWBB: EvaluationWBB trainingWindow-WBBBalance-Training: TrainingWindow-WBBBalance-Training	***Name der Klasse***	*dauerhaft*
	1	saveUser-Info()	abspeichern der aktuellen Benutzerdaten		UserInfo	nein
	2	save-Evaluation-Data()	abspeichern der berechneten Bewertungs-daten		Evaluation	ja
	3	saveTraining-Data()	abspeichern der Übungsbezeich-nung, ÜbungsID und des Datums		CreateTraining	nein

4 Datenmodell

In diesem Abschnitt wird mit Hilfe des Klassendiagramms Abbildung 4.1 die dauerhafte Speicherung der Accountdaten und Auswertungsdaten beschrieben. Dabei wird sich auf die permanente Speicherung der Daten beim Ausführen einer Übung mit dem Wii-Balance-Board bezogen.

4.1 Diagramm

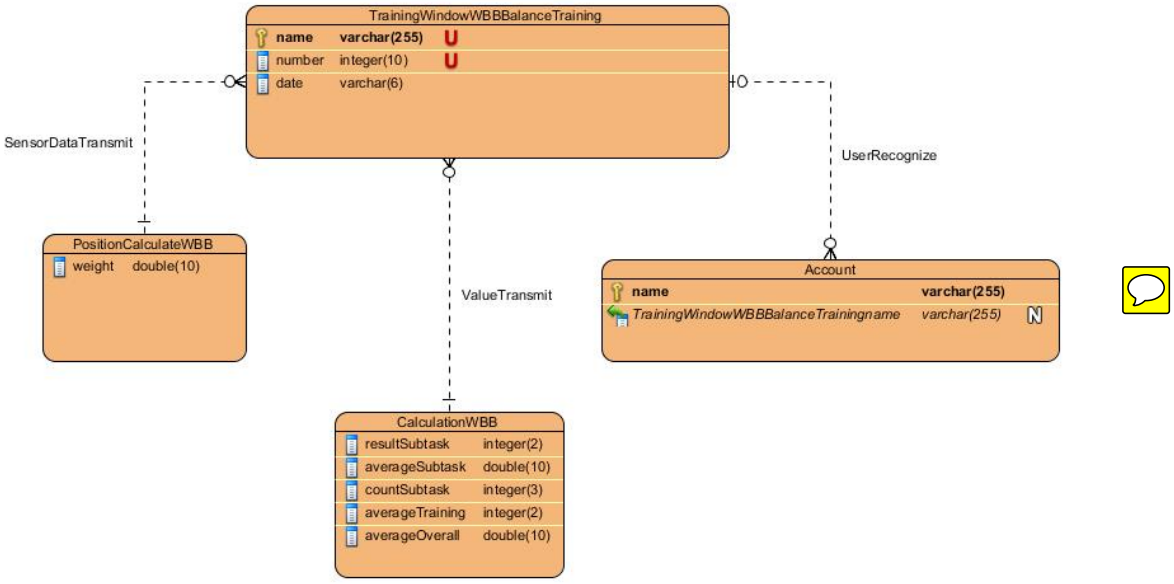

Abbildung 4.1: Datenmodell

4.2 Erläuterung

Jedes TrainingWindow des Wii-Balance-Board enthält seinen Namen als String und eine spezifische Nummer, damit die gespeicherten Daten, insbesondere die Auswertungsdaten, einer Übung zugewiesen werden können. Aus diesem Grund werden auch die Accountdaten, also der Name des Benutzers gespeichert. Über die Klasse PositionCalculationWBB, die mit dem TrainingWindowWBB verknüpft ist, wird zusätzlich das Gesamtgewicht des Patienten gespeichert. Die

wichtigste Verknüpfung ist die zwischen TrainingWindowWBBBalanceTraining und CalculationWBB. Mit Hilfe dieser Klasse werden die wichtigsten Daten dauerhaft gespeichert. Diese Daten sind die Auswertungsdaten, mit deren Hilfe der Arzt später nachvollziehen kann, in wie weit, der Benutzer Übungen ausgeführt hat und welche Fortschritte innerhalb einer Übung zu erkennen sind.