Willstätter-Gymnasium Nürnberg Oberstufenjahrgang 2015/2017

S e m i n a r a r b e i t

im Seminar

Softwareentwickelung von Computerspielen

(Leitfach Informatik)

Lässt sich Extreme Programming bei der Entwicklung einer Software in Partnerarbeit anwenden? Erprobt anhand der Entwicklung des Spiels Minesweeper mit der IDE Greenfoot.

Verfasser : Alexander Wiedemann

Seminarleiter: OstR Seiler

Abgabetermine: 8.11.2016

Erzielte Punkte schriftlich: ..... in Worten: .....
(0 – 15, einfache Wertung)

Erzielte Punkte mündlich: ..... in Worten: .....
Präsentation / Prüfungsgespräch
(0 – 15, einfache Wertung)

.....
Unterschrift des Seminarleiters

Inhaltsverzeichnis

1 Einführung

Extreme Programming(XP), was ist das überhaupt? XP ist ein agiler Softwareentwicklungsprozess[1], der aus verschiedenen Werten, Prinzipien und Praktiken besteht.

Die Grundwerte auf denen XP beruht sind Kommunikation, Schlichtheit, Feedback und Mut[2]. Auf diesen Werten bauen dann die einzelnen Arbeitstechniken und Praktiken auf. Eine dieser Arbeitstechniken ist das Planungsspiel, also das Aufschreiben der Wünsche des Kunden auf sogenannte Storycards, was dazu dient die wichtigsten Features des Programms herauszufinden. Anschließend werden die Features nach Aufwand und Risiko der Umsetzung sortiert, sodass der Kunde im nächsten Schritt diejenigen Storycards auswählen kann, die er implementiert haben möchte[3].

Andere, für XP typische, Praktiken sind die Modultests, sowie die daraus folgende kontinuierliche Integration. Bei den Modultests werden kleine Code-Stücke in Test-Frameworks getestet. Bei der kontinuierlichen Integration werden dann alle Code-Stücke zusammengeführt und die aktuelle Version des gesamten Quelltextes auf Funktionalität geprüft[4].

Eine weitere Praktik von XP ist das Programmieren in Paaren. Hierbei arbeiten zwei Entwickler zusammen. Einer der beiden übernimmt den "aktiven" Part, er erzeugt also durch Kodierung den Quelltext. Der andere stellt Fragen, denkt mit, sucht nach Fehlern und überlegt ob es einfachere Lösungen für das derzeitige Problem gibt. Die Entwickler tauschen hierbei ständig die Rollen[5], was zu einer höheren Codequalität führt[6]. Um diese und andere partnerorientierte Praktiken von XP anwenden zu können wurde das Spiel Minesweeper in Zusammenarbeit mit Lixian Lao programmiert.

Es wurden allerings einige Aspekte von XP nicht angewandt. Die Modultests und die kontinuierliche Integration waren für das Programmieren von Minesweeper nicht nötig da der Quelltext nicht all zu lang und auch nicht sonderlich kompliziert ist.

1 Westphal, Frank: Extreme Programming, 26.08.2001, Z. 2
2 Sixtus, Mario: Gemeinsam auf die Spitze treiben, 22.12.2003, Z. 12 ff
3 Hubwieser, Peter: Informatik 4, Ernst Klett Verlag, Stuttgart 2009, S. 180
4 Hubwieser, S. 181
5 Hubwieser, S. 181
6 Westphal, Z. 87 f

Das bedeutet jedoch nicht, dass der Quelltext nicht getestet wurde, da der komplette Code nach jeder Implementierung einer neuen Methode überprüft und getestet wurde. Des weiteren haben wir auch keinen Kunden, also haben wir diese Rolle praktisch selbst übernommen, und sind damit quasi Entwickler und Kunde zugleich. Bei der Programmierung in Paaren wurde der Aspekt des ständigen Tauschens der Rollen nicht berücksichtigt, sodass Lixian Lao die meiste Zeit den Quelltext geschrieben hat. Andere hier nicht erwähnte Praktiken wurden auch nicht angewandt. Wir haben uns bei der Entwicklung an dem Originalspiel von Windows sowie an einem Greenfoot-Szenario orientiert.

Im Folgenden soll das Spiel Minesweeper erklärt werden und die bei der Entwicklung verwendeten Aspekte von Extreme Programming erläutert werden.

2 Das Spiel Minesweeper und seine Geschichte

2.1 Die Geschichte des Spiels

Minesweeper wurde ursprünglich von Microsoft im Jahre 1992 für Windows 3.1, als Teil der Standardinstallation, entwickelt. Das Spiel diente ursprünglich als Trainingsprogramm für die Bedienung der Maus. Man konnte gut sein eigenes Tempo der Maus herausfinden und an seiner Präzision arbeiten, schnell bestimmte Stellen anzuklicken[7].

Seit Windows 8 muss das Spiel als App aus dem Windows Store nachinstalliert werden[8].

2.2 Prinzip des Spiels

2.2.1 Schwierigkeitsstufen

Im Spiel Minesweeper gibt es insgesamt drei verschiedene Schwierigkeitsstufen: Anfänger, Fortgeschritten und Profi.

Bei der Anfängerstufe wird ein Spielfeld mit den Maßen 8x8 Felder erzeugt, in welchem sich insgesamt 10 Minen verbergen. In der Fortgeschrittenenstufe wird ein Spielfeld mit den Maßen 16x16 Felder erzeugt, in welchem sich insgesamt 40 Minen verbergen.

7 Horn, Dennis: Solitär, Minesweeper und Co. - und was wirklich hinter den Windows-Spielen steckt, 20.01.2016, Z. 21 ff
8 Horn, Z. 33 ff

Und im Profi Modus wird ein Feld mit den Maßen 30x16 erzeugt, in welchem sich insgesamt 99 Minen verbergen.

2.2.2 Anleitung zum Spiel

Am Anfang wird eine von drei Schwierigkeitsstufen gewählt, die die Anzahl der Minen und die Größe des Spielfeldes festlegt. Es werden nun alle Felder verdeckt und die Minen auf dem Spielfeld verteilt. Das Ziel des Spiels ist es jedes Feld aufzudecken, hinter welchem sich keine Mine verbirgt.

Durch Linksklicken kann der Spieler ein Feld aufdecken. Beim ersten Klick gibt es 3 mögliche Szenarien: Es wird eine einzelne Zahl frei; Es wird eine Mine frei, wodurch man das Spiel verliert; Es wird ein Feld mit weder einer Zahl noch einer Mine frei, wodurch sich automatisch die umliegenden Felder auch aufdecken, da sich dort keine Minen verbergen können.

Die Zahlen zeigen an, wie viele Minen sich um dieses Feld verbergen. Auf jedem Feld kann maximal eine Mine liegen und somit können sich um ein Feld herum nur maximal 8 Minen befinfden.

Durch Rechtsklicken kann der Spieler ein Feld als Mine markieren. Dort erscheint eine Flagge und macht das Feld nicht 'klickbar'. Durch einen erneuten Rechtsklick kann der Spieler das Feld als unsicher markieren. Dort erscheint nun ein Fragezeichen und das Feld bleibt nicht 'klickbar'.

Wenn man einen Doppelklick mit der linken Maustaste auf ein aufgedecktes Feld mit einer Zahl macht, um welches herum alle möglichen Minen markiert wurden, werden die anderen nicht markierte Felder um das Feld herum aufgedeckt.

Achtung: Die Anzahl der Flaggen muss der Zahl auf dem Feld entsprechen und bei falscher Markierung verliert der Spieler, weil er eine Mine aufgedeckt hat.

3 Verortung in der Computerspiel-Literatur: Jane McGonigal „Besser als die Wirklichkeit“

Nach Jane McGonigal liegen allen Spielen die folge vier Kernelementen zugrunde[9]:

Es gibt ein Ziel, welches die Aufmerksamkeit des Spielers fesselt und sie in eine geeignete Richtung lenkt[10].

9 McGonigal, Jane: Besser als die Wirklichkeit, Wilhelm Heyne Verlag, München 2012, S.32 ff
10 McGonigal, S. 33

Es gibt Regeln, die den Spieler bei seinem Versuch an das Ziel zu gelangen einschränken[11].

Es gibt in irgendeiner Form ein Feedbacksystem, welches den Spieler darüber informiert wie nah er dem Ziel ist[12].

Es herrscht das Prinzip der freiwilligen Teilnahme, welches voraussetzt, dass jeder Spieler das Ziel, die Regeln und das Feedbacksystem kennt und all diese Punkte anerkennt[13].

Auch das Spiel Minesweeper erfüllt diese Kernelemente:

Das Ziel ist es jedes Feld, unter dem sich keine Mine befindet, aufzudecken.

Die Regeln sind in Minesweeper recht simpel: Wenn mit linksklick auf ein Feld mit Bombe geklickt wird ist das Spiel vorbei.

Das Feedbacksystem ist hierbei ein Scoreboard, welches nach beenden des Spiels die Punktezahl anzeigt. Außerdem wurde noch ein Echtzeitfeedback über die Anzahl der markierten Felder implementiert, das zur Motivationssteigerung führen soll.

In Minesweeper gilt natürlich auch das Prinzip der freiwilligen Teilnahme, da man nicht gezwungen wird das Spiel zu spielen.

Außerdem nennt McGonigal Maßnahmen mit denen sich die Wirklichkeit, durch Computerspiele, verbessern lässt[14]. So sind Spiele unter anderem unnötige Hindernisse, da wir nach dem Prinzip der Freiwilligen Teilnahme diese ja nicht bewältigen müssen. Sich mit unnötige Hindernissen zu befassen bewirkt jedoch, dass die Selbstmotivation gesteigert wird, die Interesse und Kreativität geweckt wird und wir immer auf der Höhe unserer Fähigkeiten arbeiten[15].Darüber hinaus führen Spiele auch zur emotionalen Stimulation, indem sie, egal wie die Laune des Spielers im Momment ist, Glücksgefühle auslösen, da der Spieler eine optimistische Wahrnehmung des eigenen Leistungspotentials durch Bewältigung des Spiels erfährt[16]. Spiele bewirken auch das Empfinden von glückseliger Aktivität, indem sie uns klare und leicht verfolgbare Ziele und eindeutige Resultate liefern[17].

11 McGonigal, S. 33
12 McGonigal, S. 33
13 McGonigal, S. 33 f
14 McGonigal, S. 449 ff
15 McGonigal, S. 449 (Fix #1)
16 McGonigal, S. 449 (Fix #2)
17 McGonigal, S. 449 (Fix #3)

Die oben genannten Maßnahmen treffen auch auf Minesweeper zu, da das Spiel an sich ein unnötiges Hindernis darstellt welches man durch logisches denken bewältigen kann. Schafft man es also Minesweeper zu bewältigen führt das zu Glücksgefühlen, glückseliger Aktivität und zur optimistischen Wahrnehmung des eigenen Leistungspotentials.

4 Storycards

4.1 Alle Storycards

Story	Testszenario	Nr
Es gibt einen Startscreen der eine Schwierigkeitsstufenauswahl hat. Nachdem man eine Schwierigkeitsstufe gewählt hat wird ein Spielfeld mit bestimmten Maßenerschaffen, auf jedes Feld wird jeweils ein Knopf gesetzt und die Minen werden zufällig verteilt.	Starten des Spiels.	1
Unter einem Knopf gibt es höchstens eine Mine.	Nachschauen.	2
Der Spieler drückt auf einen Knopf, der entweder verschwindet oder explodiert.	Der Spieler klickt mit der linken Maustaste auf einen Knopf.	3
Auf den offenen Feldern gibt es Zahlen, die anzeigen wie viele Minen es um das jeweilige Feld gibt.	Der Spieler klickt mit der linken Maustaste auf einen Knopf, um den es keine Minen gibt.	4
Wenn der Spieler auf ein Feld drückt, um das es keine Minen gibt, öffnen sich umliegende Felder automatisch.	Der Spieler klickt mit der linken Maustaste auf einen Knopf, der nicht explodiert.	5
Der Spieler kann auf Feldern, zur Hilfe, eine Flagge oder ein Fragezeichen plazieren um eine Mine zu markieren	Der Spieler klickt mit der rechten Maustaste auf einen Knopf	6
Der Spieler kann auf eine Zahl, um die alle mögliche Minen markiert wurden doppelt linksklicken um alle nicht markierten Felder aufzudecken. Der Spieler verliert jedoch wenn eine der Markierungen falsch ist.	Der Spieler doppelklickt mit der linken Maustaste auf eine Zahl	7
Der Spieler verliert sobald er auf einen Knopf drückt, unter dem eine Mine liegt und es erscheint ein Scoreboard.	Der Spieler drückt auf einen Knopf, unter dem eine Mine liegt.	8
Der Spieler gewinnt sobald alle Knöpfe ohne Mine gedrückt sind und es erscheint ein Scoreboard.	Alle Knöpfe drücken, unter denen keine Mine ist.	9

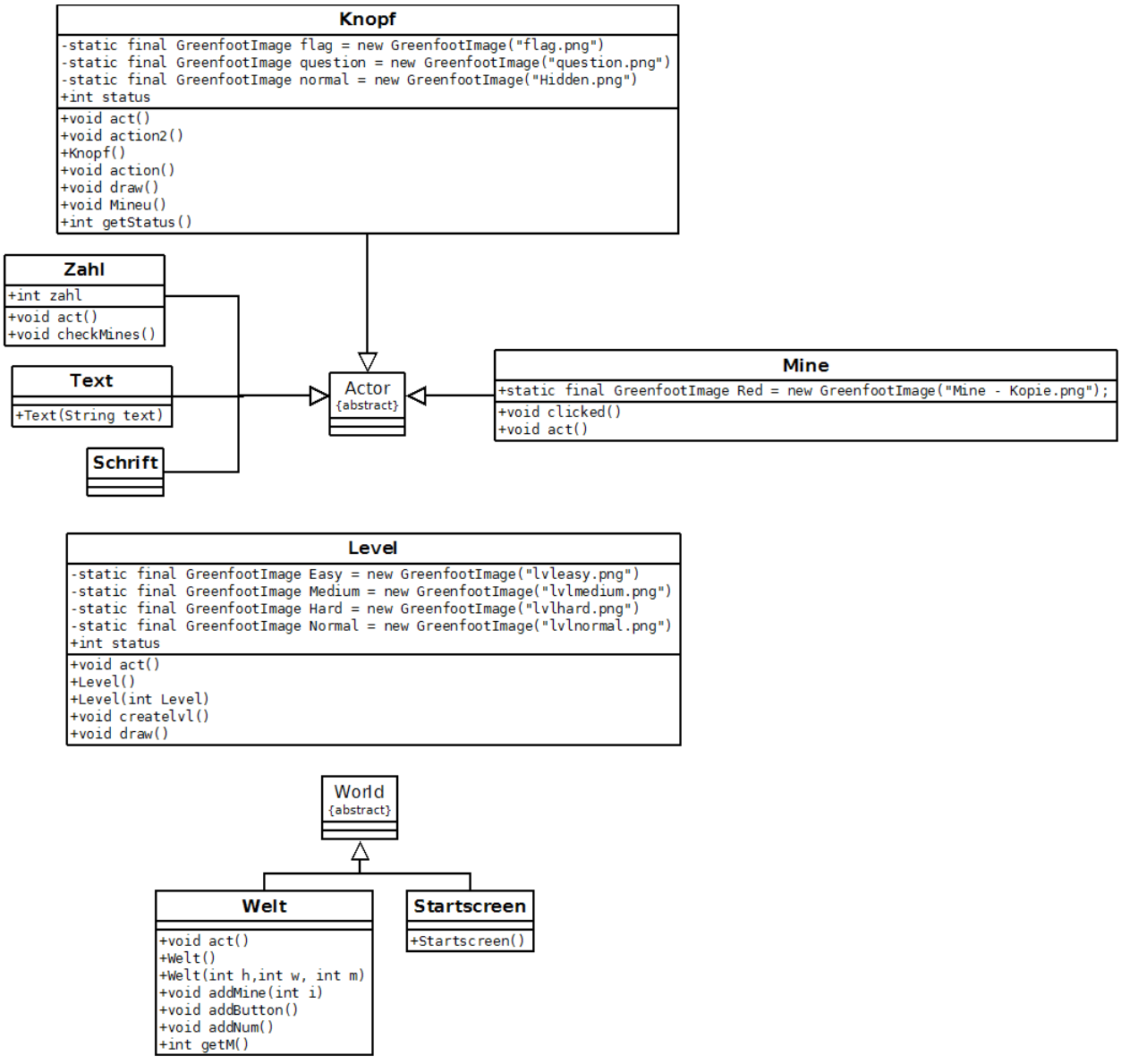
Wenn das Spiel gewonnen oder verloren wird gibt es ein Feuerwerk.	Das Spiel gewinnen oder verlieren.	10

4.2 Nähere Erläuterung von zwei exemplarisch ausgewählten Storycards

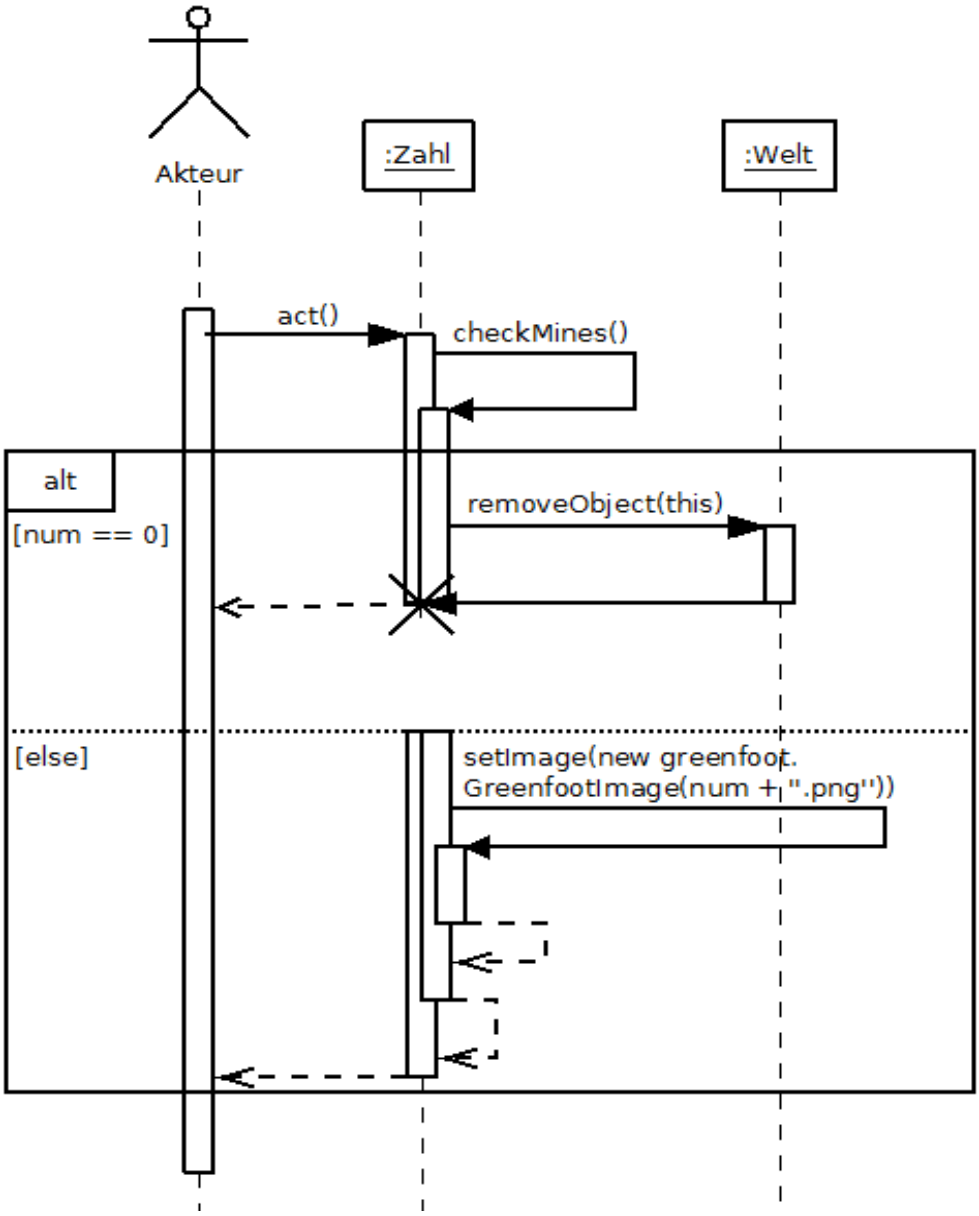
4.2.1 Erläuterung der Storycard Nummer 4

Da das Spiel nicht nur auf Glück basieren soll gibt es Zahlen, die dem Spieler helfen Minen ausfindig zu machen. Diese Storycard befasst sich genau mit diesem Thema. Für die Umsetzung muss zuerst eine neue Klasse 'Zahl' erstellt werden. Die Klasse benötigt eine Variable 'zahl' in der die Anzahl der umliegenden Minen gespeichert wird. Des Weiteren muss die Anzahl der umliegenden Minene ermittelt werden. Dies geschieht durch die Methode chekMines() in der Klasse 'zahl'.4.2.1.1

Klassendiagramm:



4.2.1.2 Sequenzdiagramm



Der Akteur ist in diesem Fall das Greenfoot Framework, da dieses die act() Methode aufruft.

4.2.1.3 Erklärung wichtiger Methoden

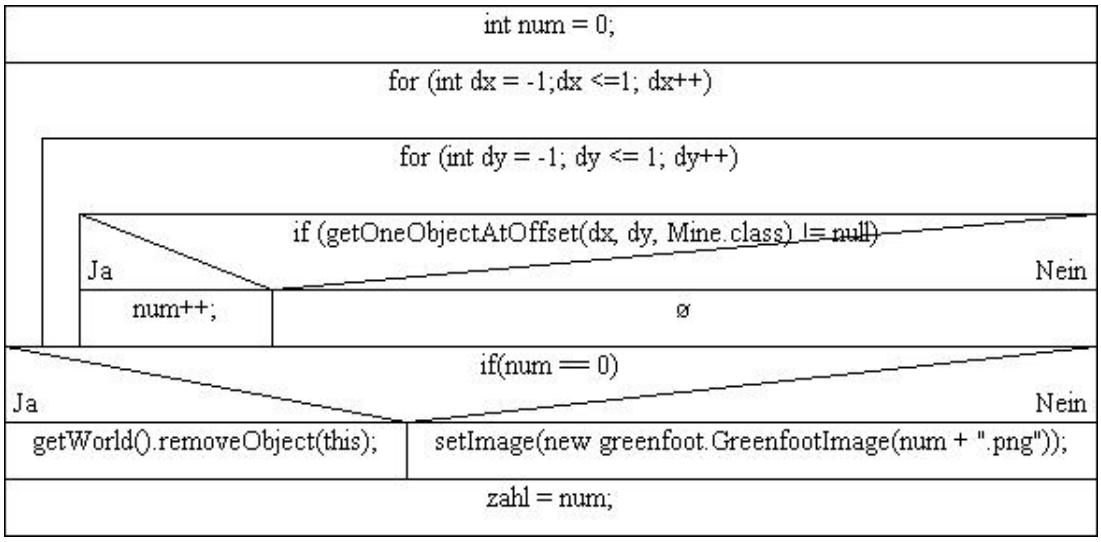
Die Methode act() in der Klasse 'Zahl':

Die Methode act() wird von der Klasse 'Actor' vererbt. Sie wird vom Greenfoot Framework immer wieder an jedem Objekt aufgerufen. In der Klasse 'Zahl' bedeutet das, dass jedes erzeugte Objekt der Klasse 'Zahl' andauernd die Anzahl der Minen um es herum überprüft.

Die Methode checkMines() in der Klasse 'Zahl':

Diese Methode such die umliegenden x und y Koordinaten nach einem Objekt der Klasse 'Mine' ab und speichert die Anzahl der Minen in einer lokalen Variablen (= num). Nachdem alle umliegende Felder untersucht wurden, wird entweder das Bild der Klasse 'Zahl' an den Wert der lokalen Variable 'num' angepasst, oder das jeweilige Objekt der Klasse Zahl wird entfernt und die Anzahl der Minen in der globalen Variable 'zahl' gespeichert.

Strucktogramm zu checkMines():



Der Quelltext zu checkMines():

```
void checkMines()
{
   int num = 0;
   for (int dx = -1; dx <= 1; dx++){
      for (int dy = -1; dy <= 1; dy++){
         if (getOneObjectAtOffset(dx, dy, Mine.class) != null){
            num++;
         }
      }
   }
   if (num == 0){
      getWorld().removeObject(this);
   }
   else{
      setImage(new greenfoot.GreenfootImage(num + ".png"));
   }
   zahl=num;
}
```

Die Methoden removeObjekt(), setImage() und getOneObjectAtOffset():

Die Methoden removeObjekt(), setImage() und getOneObjectAtOffset() sind durch Greenfoot bereitgestellte Methoden. Die Methode removeObjekt() entfernt das als Parameter mitgegebene Objekt, die Methode setImage() gibt dem Objekt ein neues Bild und die Methode getOneObjectAtOffset() gibt an, ob sich an einer, durch die Parameter, bestimmten Stelle ein bestimmtes Objekt befindet. Der Quelltext von durch Greenfoot bereitgestellte Methoden kann nicht eingesehen werden und es ist lediglich ihre Funktion in der Dokumentation beschrieben, was der Grund dafür ist, dass der Quelltext an dieser Stelle nicht angegeben ist.

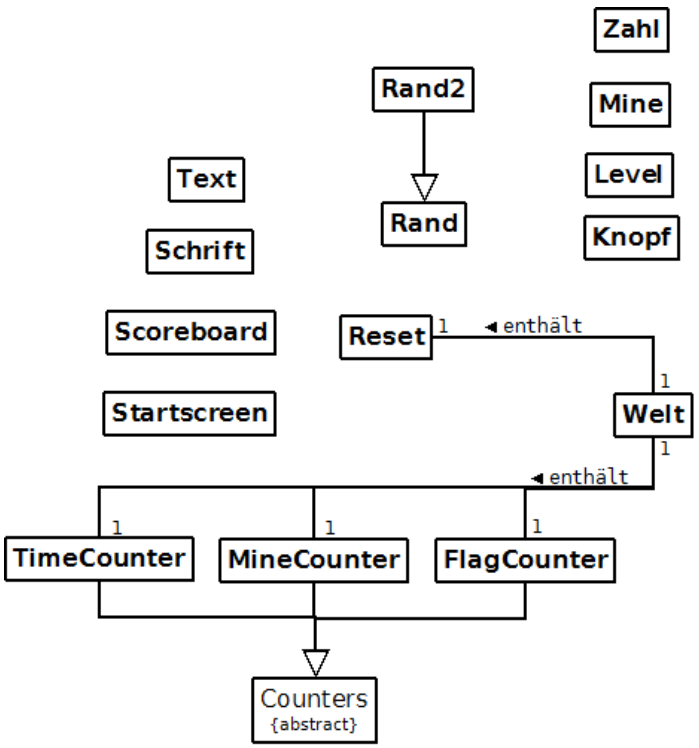
4.2.2 Erläuterung der Storycard Nummer 9

Damit man ein Spiel gewinnen kann muss es eine Siegesbedingung geben. Die Siegesbedingung in Minesweeper ist es jeden Knopf aufzedecken unter dem sich keine Mine befindet. Wenn dieses Ziel erreicht ist soll es außerdem, ganz nach Jane McGonigal, ein Feedback in Form eines Scoreboards geben, welches einem die benötigte Zeit anzeigt.

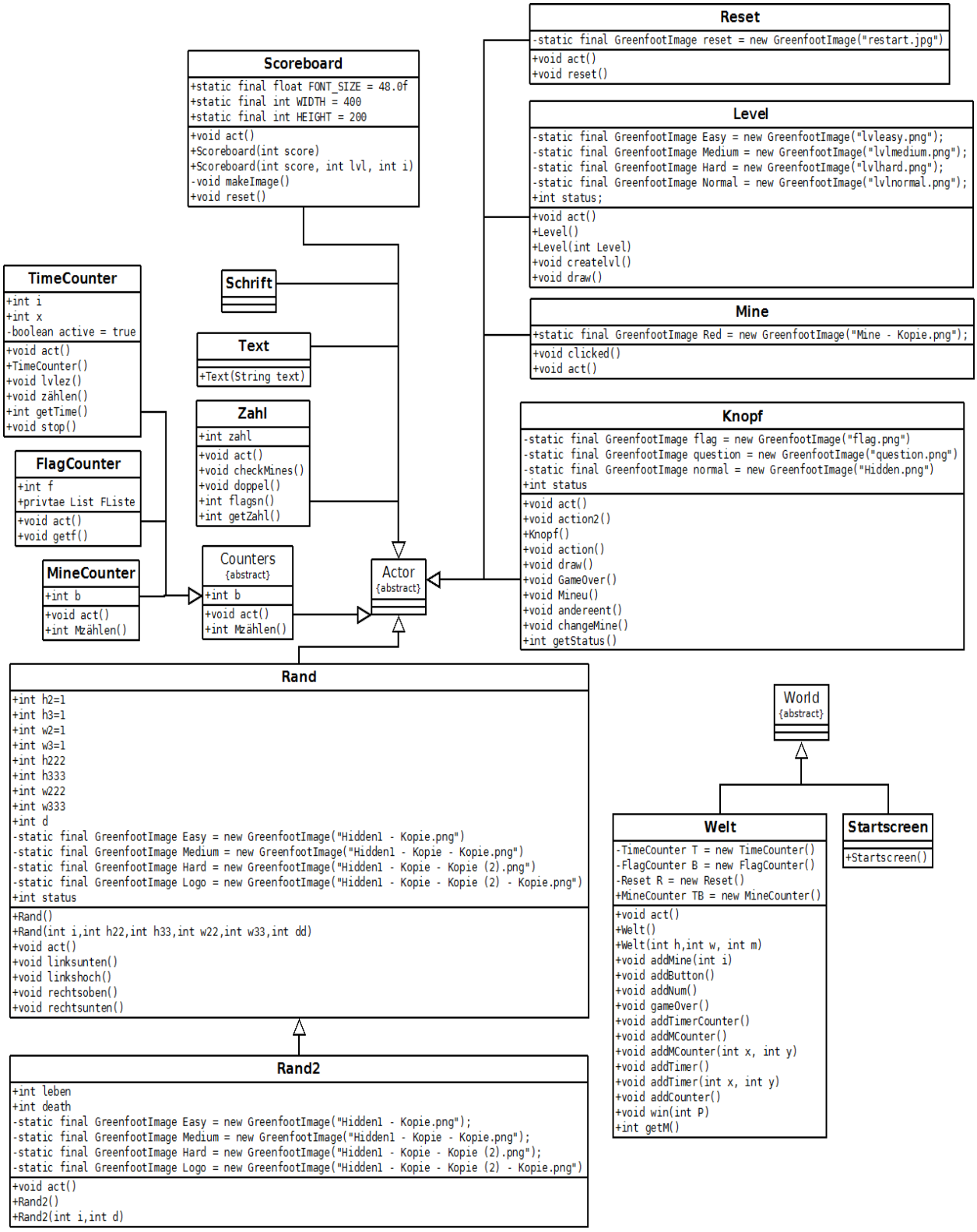
4.2.2.1 Klassendiagramme

Ein gesamtes Klassendiagramm wurde zur Übersichtlichkeit in zwei aufgeteilt.

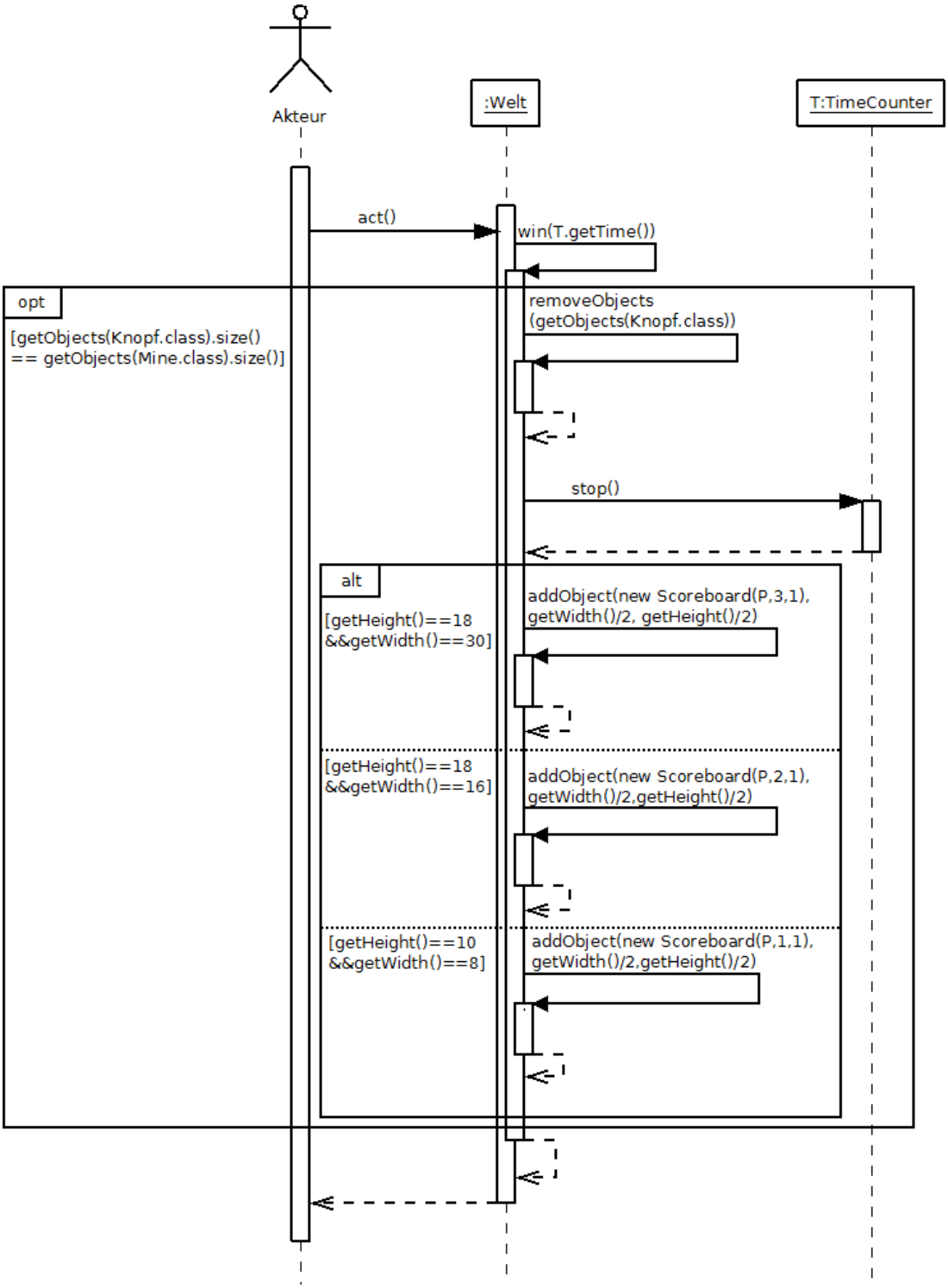
Klassendiagramm ohne die Klassen 'Actor' und 'World' und ohen Attribute und Methoden:



Klassendiagramm ohne Assoziationen:



4.2.2.2 Sequenzdiagramm



Der Akteur entspricht auch in diesem Sequenzdiagramm dem Greenfoot Framework.

4.2.2.3 Erklärung wichtiger Methoden

Die Methode act() in der Klasse 'Welt':

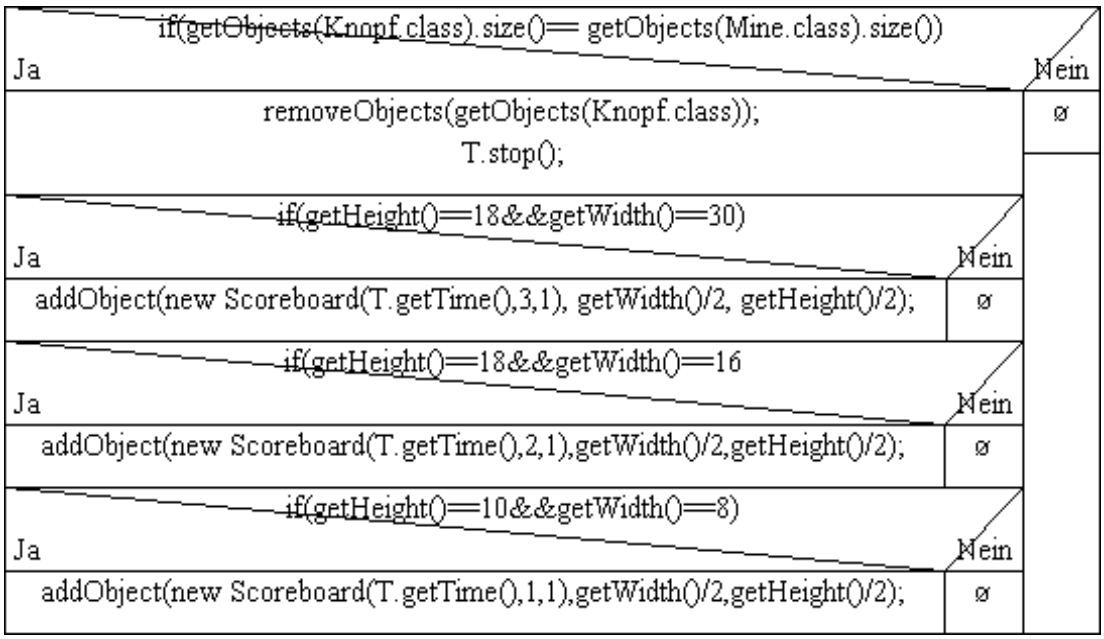
So wie alle Objekte in Greenfoot besitzen auch die Objekte der Klasse 'Welt' eine act()-Methode die durch das Greenfoot Framework aufgerufen wird.

In der Klasse 'Welt' führt dies dazu, dass die Methode win(int P) in der Klasse 'Welt' ständig aufgerufen wird. Hierbei wird der Parameter T.getTime() mitgegeben, welcher dazu dient die Zeit auf dem Scoreboard nach dem gewinnen des Spiels anzuzeigen.

Die Methode win(int P) in der Klasse 'Welt':

Die Methode stellt fest ob alle Knöpfe, unter denen keine Mine liegt, aufgedeckt wurden, also ob die Siegesbedingung erfüllt ist. Wenn dies der Fall ist werden alle übrig gebliebenen Objekte der Klasse 'Knopf' entfernt und somit die Positionen aller Minen offenbart. Daraufhin wird der Zeitzähler gestoppt. Anschließend wird das Scoreboard mit bestimmten Maßen je nach gespieltem Schwierigkeitsgrad erstellt. Hierbei wird die aus der act()-Methode mitgegebene Zeit als Parameter weitergegeben.

Strucktogramm zu win(int P):



Quelltext zu win(int P):

```
public void win(int P)
   {
      if(getObjects(Knopf.class).size()== getObjects(Mine.class).size())
      {
         removeObjects(getObjects(Knopf.class));
         T.stop();
         if(getHeight()==18&&getWidth()==30)
         {
            addObject(new Scoreboard(P,3,1), getWidth()/2, getHeight()/2);
         }
         if(getHeight()==18&&getWidth()==16)
         {
            addObject(new Scoreboard(P,2,1),getWidth()/2,getHeight()/2);
         }
         if(getHeight()==10&&getWidth()==8)
         {
            addObject(new Scoreboard(P,1,1),getWidth()/2,getHeight()/2);
         }
      }
   }
```

Die Methode stop() in der Klasse 'TimeCounter':

Diese Methode dient lediglich dazu den Zeitzähler anzuhalten. Sie erreicht das indem sie die globale Variable 'active' auf 'false' setzt und somit die act()-Methode in der Klasse 'TimeCounter' nichts mehr macht.

Quelltext zu stop():

```
public void stop()
   {
      active =false;
   }
```

Quelltext zu act() von der Klasse 'TimeCounter':

```
public void act(){
      if(active == false){}
      else{
         x++;
         zählen();
         setImage(new GreenfootImage ("Zeit : "
                        + i,28,Color.WHITE,Color.BLACK));
      }
   }
```

Die Klasse 'Scoreboard':

Die Klasse dient dazu am Ende des Spiels ein Scoreboard anzuzeigen. Da das Scoreboard schon in Storycard Nummer 8 eingeführt wurde musste die Klasse nur noch geringfügig erweitert werden.

Die Methode act() in der Klasse 'Scoreboard':

Sie dient dazu, dass man, nachdem das Scoreboard erstellt wurde, durch Rechtsklick ein neues Spiel starten kann.

Quelltext zu act() aus der Klasse 'Scoreboard':

```
public void act() {
      if(Greenfoot.mouseClicked(this)){
         if(Greenfoot.getMouseInfo().getButton()==3){
            Greenfoot.setWorld(new Startscreen());
         }
      }
   }
```

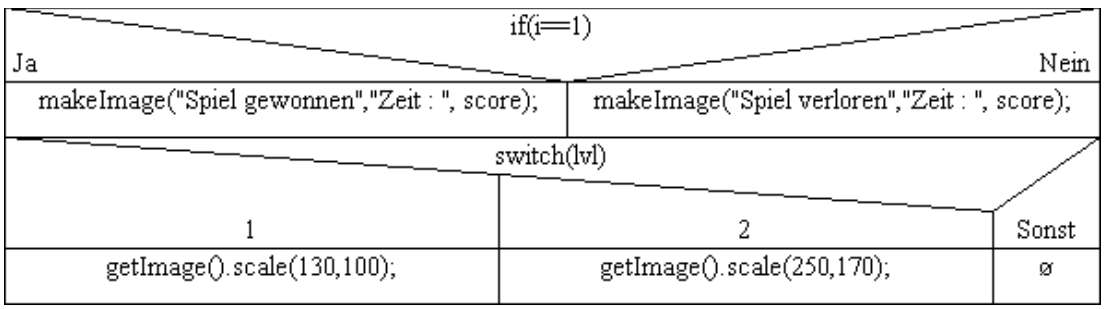
Der Konstruktor der Klasse 'Scoreboard':

Hier wird die Größe des Scoreboards sowie der Inhalt bestimmt.

Quelltext zum Konstruktor der Klasse 'Scoreboard':

```
public Scoreboard(int score, int lvl, int i){
      if(i==1){
         makeImage("Spiel gewonnen","Zeit : ", score);
      }
      else{
         makeImage("Spiel verloren","Zeit : ", score);
      }
      switch(lvl){
         case 1: getImage().scale(130,100);
         case 2: getImage().scale(250,170);
      }
   }
```

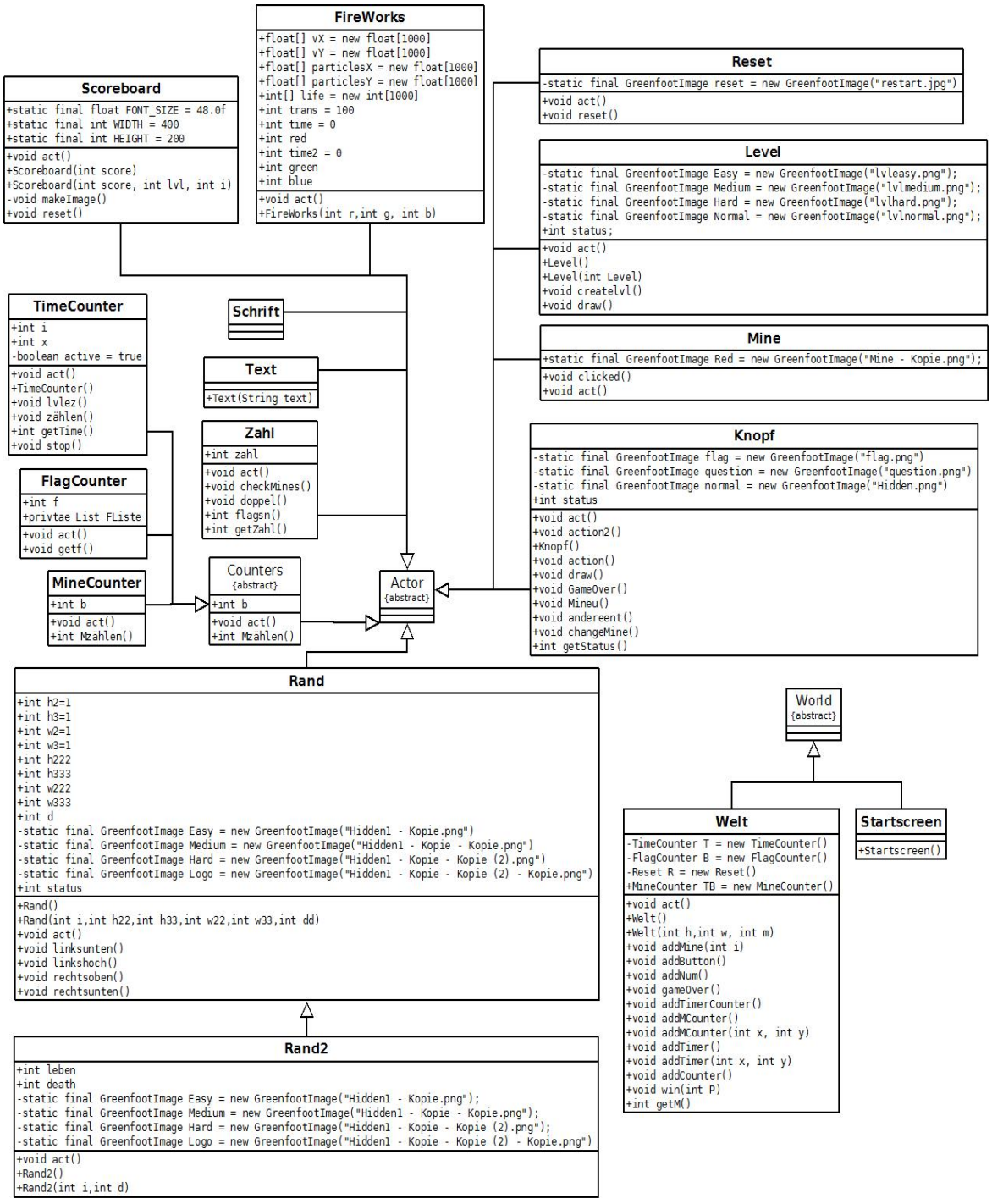
Strucktogramm zum Konstruktor der Klasse 'Scoreboard':



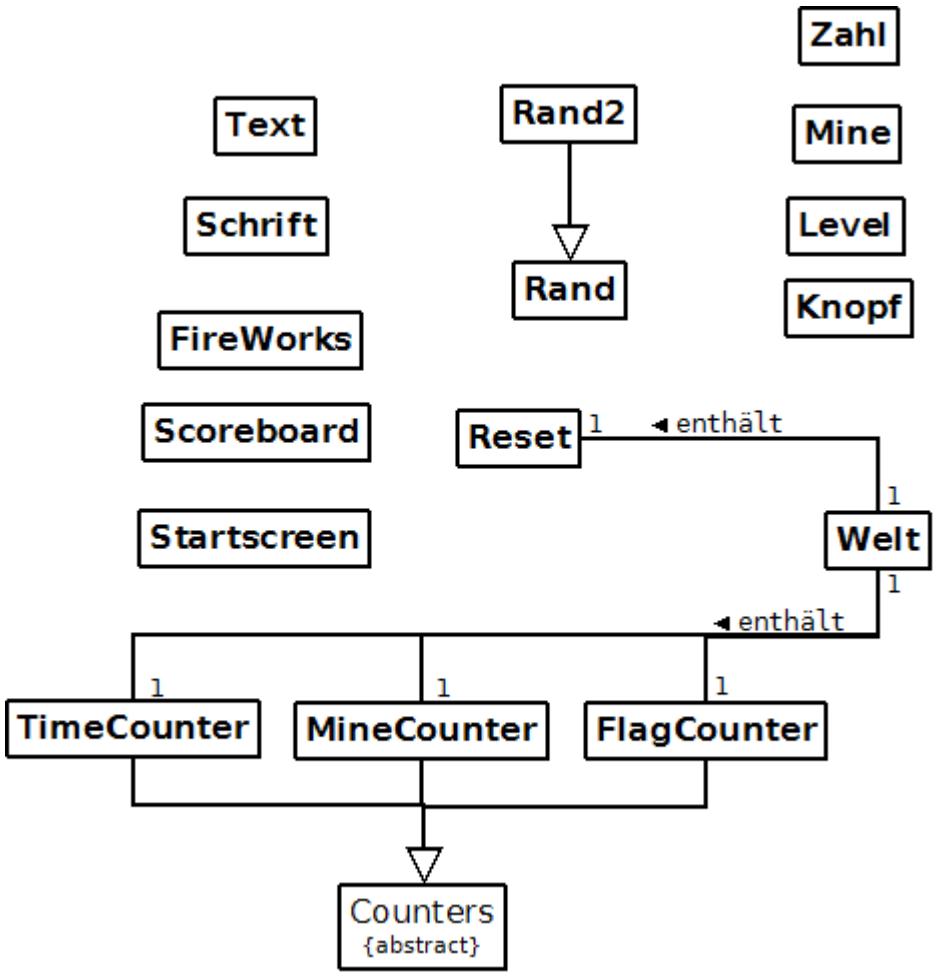
5 Klassendiagramme des fertiggestellten Spiels

Aus Gründen der Übersichtlichkeit wurde ein gesamtes Klassendiagramm auf zwei aufgeteilt. Der einzige Unterschied zwischen diesen Klassendigrammen und denen zu Storycard Nummer 9 ist, dass die Klasse 'FireWorks' hinzugefügt wurde.

Klassendiagramm ohne Assoziationen:



Klassendiagramm ohne die Klassen 'Actor' und 'World' und ohen Attribute und Methoden:



Alle Klassendiagramme sind auch als Bild und als Dia-Datei dem Anhang beigefügt.

6 Verwendete Aspekte von XP

Der erste und offensichtlichste Aspekt der verwendet wurde ist das Erstellen von Storycards. Dadurch konnten die benötigten Methoden bereits vor dem Programmieren entworfen werden, was zu einem gut struckturierten Quelltext geführt hat.

Des Weiteren ist das Spiel in Partnerarbeit umgesetzt worden. Lixian Lao hat den Quelltext geschrieben und ich habe ihn fortlaufend auf Fehler untersucht und Verbesserungsvorschläge eingebracht. Außerdem wurde während der Programmierung des Spiels immer wieder neue Features entworfen. So ist zum Beispiel das Scoreboard entstanden. Diese fortlaufende Entwicklung nennt sich Refactoring[18].

18 Westphal, Z. 100 ff

Hierbei wird das vorliegende System in kleinen Schritten so verbessert, dass alle bisherigen Funktionen erhalten bleiben und die neue Funktion ohne Fehler in das System integriert werden kann[19].

Den Aspekt der Programmierung in kleinen Schritten kann man deutlich daran erkennen, dass sich die Klassendiagramme von dem fertigen Spiel und die von Storycard Nummer 9 so ähnlich sind.

Natürlich mussten nicht alle Features des Spiels selbst entworfen werden, wegen der Orientierung an Minesweeper von Windows. Außerdem wurden die Methode checkMines()[20], die Methode andereent()[21] und die Klasse 'FireWorks'[22] aus Greenfoot-Szenarien übernommen, was viel Zeit gespart hat.

7 Abschließende Worte

7.1 Persönliche Bewertung der eingesetzten Aspekte von XP

Allgemein haben die von uns verwendeten Aspekte von XP gut dazu beigetragen das Spiel zu entwickeln.

Die Storycards haben gut dabei geholfen das Spiel Stück für Stück aufzubauen. Außerdem wusste man durch die Storycards ganz genau an welcher Stelle der Entwicklung man sich gerade befindet, indem sie einem veranschaulicht haben was bereits geschafft wurde und was noch vor einem lieg.

Das Programmieren in Paaren hat mir gut gefallen, da ich mich dadurch sehr gut darauf konzentrieren konnte die Fehler in unserem Spiel zu suchen.

XP bietet eine gute alternative zu herkömmlichen Programmierpraktiken wie zum Beispiel dem Wasserfallmodell. Beim Wasserfallmodell kann man während der Phase der Implementierung keine Features mehr hinzufügen wodurch sich die Entwicklung sehr unflexibel gestalltet[23], was hingegen in XP kein Problem darstellt, da das Programm ständig weiterentwickelt wird.

19 Westphal, Z. 100ff
20 FlyingRabidUnicornPig: Minesweeper, 30.06.2013, Z. 194 ff
21 FlyingRabidUnicornPig, Z. 371 ff
22 MatheMagician: FireWorks, 04.07.2012, Z. 729 ff
23 Hubwieser, S. 140ff

7.2 Die Rolle des Lehrers im Seminar

Herr Seiler hat seine Rolle als Seminarleiter sehr gut wahrgenommen. Er hat den Seminarteilnehmern die Aspekte von XP sehr gut nähergebracht, indem er mit ihenen XP an verschiedenen Beispiel durchgespielt hat. Außerdem hat er den Schülern gut erklärt wie man wissenschaftliche Arbeiten verfasst.

Das Auswerten von verschiedenen Texten im Untericht hat uns sehr viel aufwendige Arbeit zuhause erspart, da wir zum Schreiben der Seminararbeit nurnoch die verschiedenen Mindmaps anschauen mussten um den Inhalt der jeweiligen Texte gut zusammengefasst einsehen zu können.

7.3 Zukünftige Softwareentwicklung

Bei der nächsten Entwicklung einer Software werde ich definitiv wieder Aspekte von XP verwenden. Außerdem muss ich definitiv darauf achten mehr Diagramme vor dem eigentlichen Programmieren zu erstellen, da mir das etwas gefehlt hat. Obwohl dies sehr aufwädig ist, hilft es enorm, denn umso mehr Diagramme und Ideen wir zu einer bestimmten Funktion hatten desto einfacher war auch die daraus folgende Implementierung.

8 Literaturverzeichnis

McGonigal, Jane: Besser als die Wirklichkeit, Wilhelm Heyne Verlag, München 2012, S.32-34 & S.448ff

Hubwieser, Peter: Informatik 4, Ernst Klett Verlag, Stuttgart 2009, S.180ff

9 Internetquellenverzeichnis

Die Zeilenangaben zu den Internetquellen beziehen sich auf die Internetquellen als beigelegte Dokumente.

Westphal, Frank: Extreme Programming, 26.08.2001
unter: http://www.frankwestphal.de/ExtremeProgramming.html
[Stand 04.11.2016]

Horn, Dennis: Solitär, Minesweeper und Co. - und was wirklich hinter den Windows-Spielen steckt, 20.01.2016
unter: https://blog.wdr.de/digitalistan/microsoft-solitaer-minesweeper-windows/
[Stand 04.11.2016]

Sixtus, Mario: Gemeinsam auf die Spitze treiben, 22.12.2003
unter: http://www.zeit.de/2004/01/T-Extremprogrammierer/komplettansicht
[Stand 06.11.2016]

Greenfoot-Szenarien aus denen etwas übernommen wurde:

FlyingRabidUnicornPig: Minesweeper, 30.06.2013
unter: http://www.greenfoot.org/scenarios/8949
[Stand 06.11.2016]

MatheMagician: FireWorks, 04.07.2012
unter: http://www.greenfoot.org/scenarios/5678
[Stand 06.11.2016]

10 Verwendete Software

Greenfoot http://www.greenfoot.org/

OpenOffice https://www.openoffice.org

Dia http://dia-installer.de/index.html.en

XMind http://www.xmind.net/de/

Strucktogrammeditor http://www.whiledo.de/index.php?p=struktogrammeditor

11 Erklärung

Ich erkläre hiermit, dass ich die Seminararbeit selbstständig angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe.

…....., den …..... .....

Ort Datum Unterschrift des Schülers