Hausarbeit Zelluläre Automaten

I143 Praxis der Softwareentwicklung

GITLAB: HAUSARBEIT_I143_GAMRADT_CC

von Christian Apsel (9090), Fabian Forthmann (9104), Georg Mezlaw (8881),
Rane Petersen (9132)

Inhaltsverzeichnis

1. Einleitung

Diese Dokumentation wurde im Rahmen des Moduls I143 Praxis der Softwareentwicklung erstellt. Es geht um zelluläre Automaten. Dabei haben wir die festgelegten Regelsätze aus der Aufgabenstellung umgesetzt:

Parity Modell

„Eine Zelle ist genau dann lebendig, wenn in ihrer von-Neumann-Nachbarschaft eine ungerade Anzahl lebendiger Zellen existiert, andernfalls ist sie tot.“

- Startkonfiguration 1: Grid-Size: 400x400, alle Zellen werden mit "tot" initialisiert. Lebendig sind lediglich 4 Zellen in der Mitte.
- Starkonfiguration 2: Grid-Size: 100x100, gerade Zeilen werden abwechselnd mit "tot" oder "lebendig" initialisiert. Ungerade Zeilen werden abwechselnd mit "lebendig" oder "tot" initialisiert.

Game of Life Modell

„Eine lebendige Zelle bleibt lebendig, wenn Sie genau 2 oder 3 lebendige Nachbarn besitzt. Andernfalls stirbt sie. Eine tote Zelle bleibt tot bis sie exakt 3 lebendige Nachbarn hat.“

- Startkonfiguration 1: Grid-Size: 40x41, Nachbarschaft: Moore, mittig platziert
- Startkonfiguration 2: Grid-Size: 100x100, gerade Zeilen werden abwechselnd mit "tot" oder "lebendig" initialisiert. Ungerade Zeilen werden abwechselnd mit "lebendig" oder "tot" initialisiert. Nachbarschaft: Moore
- Startkonfiguration 3: Grid-Size: 300x300, alle Zellen sind "tot", Nachbarschaft: Moore

Entwickelt wurde mit der IntelliJ IDE. Zur Versionsverwaltung wurde das Gitlab der Nordakademie genutzt. Während des Gitlab-Ausfalls waren wir gezwungen, auf ein privates Github-Repository auszuweichen. Alle Commits des Repos wurden durch das Wechseln des Remotes eins zu eins in das Gitlab übernommen (24.10-25.10). Auf Anfrage wird selbstverständlich ein Zugang zu dem privaten Repository gewährt. Es wurde überwiegend mit der Arbeitstechnik des Pair Programming gearbeitet. Zweier-Gruppen haben im Wechsel gemeinsam am Quellcode gearbeitet. Dabei hat eine Person programmiert, während die zweite über die Problemstellung nachgedacht, den geschriebenen Code kontrolliert und Auffälligkeiten angesprochen hat. Es wurde eine grobe Aufteilung der Programmierung vorgenommen: Herr Forthmann und Herr Petersen haben überwiegend an der Spielelogik (Paket *gamelogic*), an dem *ConfigReader* und an dem *PropertyValidator* gearbeitet; Herr Apsel und Herr Mezlaw haben überwiegend an dem Spielfeld (Paket *grid*), dem *Logger* und der *Main* gearbeitet. Gegenseitige Unterstützung war dabei unerlässlich. Es wurde stets Test-First entwickelt.

Um die individuellen Leistungen kenntlich zu machen, wurden die Verantwortlichen mithilfe der JavaDocs kenntlich gemacht. Die Dokumentation wurde von Herrn Mezlaw verfasst.

2. UML – OO Struktur

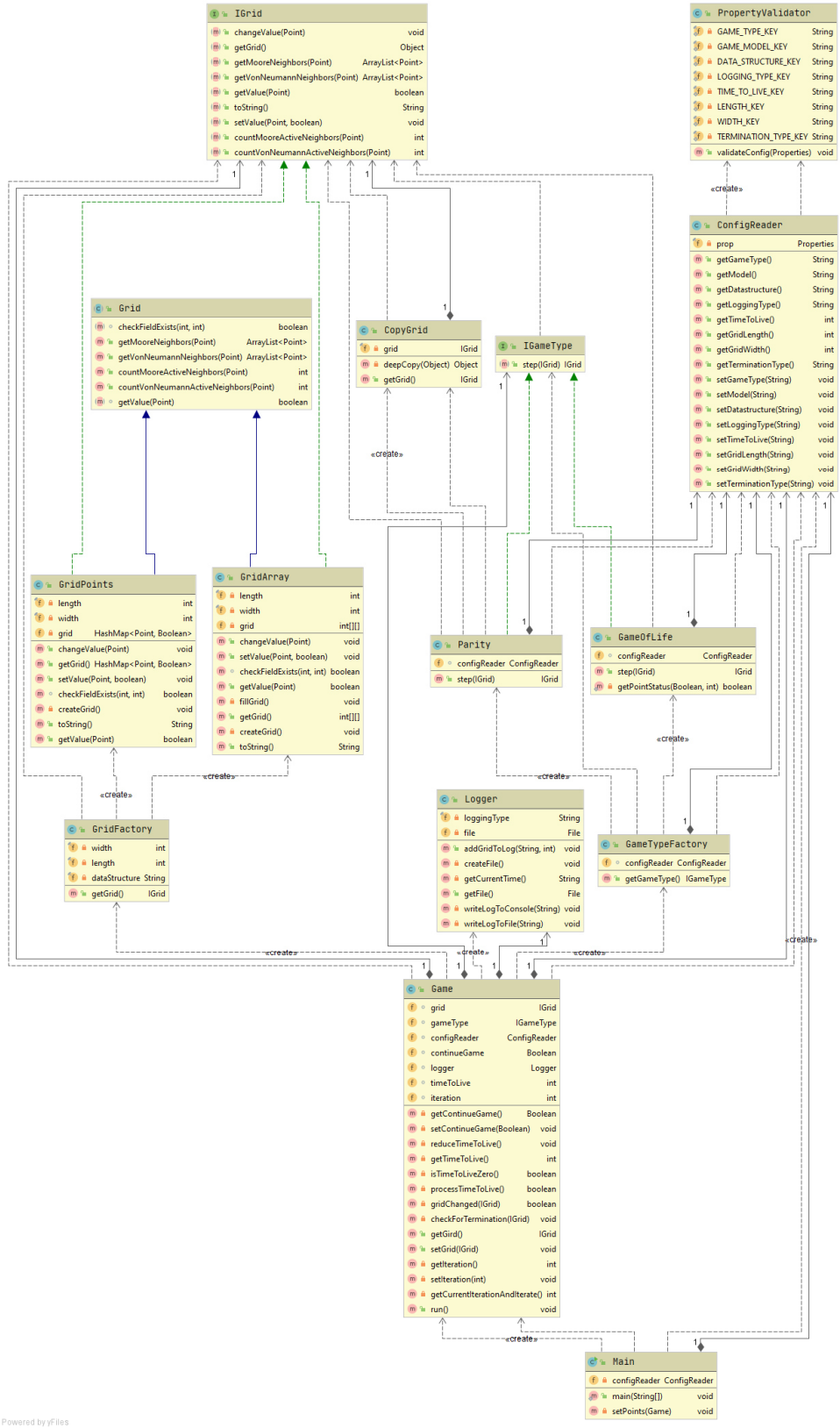


Abbildung 1 - UML ohne Experimente

Aus Gründen der Übersichtlichkeit wurden in dieser Abbildung die Experimente weggelassen. Das vollständige UML ist als Anhang beigefügt.

Nachfolgend wird unsere Struktur mit den einzelnen Objekten näher beschrieben.

Unsere OO-Struktur setzt sich aus zwei Interfaces zusammen. Das *IGrid*-Interface enthält wichtige Informationen für das Spielfeld (Grid) und enthält folgende Methoden:

Tabelle 1 IGrid-Methoden

Name der Methode	**Beschreibung**
changeValue	Setzt den Wert an einem Punkt – tauscht den Status, ob die Zelle an einem Punkt lebt oder nicht (true oder false)
getGrid	Gibt das Spielfeld zurück
getMooreNeighbors	Gibt die Nachbarn an einem Punkt nach dem Moore-Schema zurück
getVonNeumannNeighbors	Gibt die Nachbarn an einem Punkt nach dem Von-Neumann-Schema zurück
getValue	Gibt den Wert eines Punkts wider Lebt die Zelle oder nicht (true, false)
toString:	Umwandlung in einen String
setValuePoint	Setzt die Werte für einen Punkt
countMooreActiveNeighbors	Zählt die Anzahl der Nachbarn laut dem Moore-Schema
countNeumannActiveNeighbors	Zählt die Anzahl der Nachbarn laut dem Von-Neumann-Schema

Das *IGameType*-Interface enthält die Methode *step*. Dieses geht über das Spielfeld und startet die nächste Spieliteration. Es überprüft für jede Zelle, wie diese sich verändert und schreibt das Ergebnis in ein neues Spielfeld.

Tabelle 2 – IGameType-Methoden

Name der Methode	**Beschreibung**
Step	Startet nächste Spieliteration

Außerdem wurde die abstrakte Superklasse *Grid* erstellt. Diese enthält ebenfalls die Methoden *countMooreActiveNeighbors*, *countNeumannActiveNeighbors*, *getMooreNeighbors*, *getVonNeumannNeighbors* und *getValue* wie aus dem Interface *IGrid*. Zusätzlich enthält es die Methode *checkFieldExists*, die überprüft, ob ein Feld existiert.

Tabelle 3 – Grid-Methoden

Name der Methode	**Beschreibung**
countMooreActiveNeighbors	Zählt die Anzahl der Nachbarn laut dem Moore-Schema
countNeumannActiveNeighbors	Zählt die Anzahl der Nachbarn laut dem Von-Neumann-Schema
getMooreNeighbors	Gibt die Nachbarn an einem Punkt nach dem Moore-Schema zurück
getVonNeumannNeighbors	Gibt die Nachbarn an einem Punkt nach dem Von-Neumann-Schema zurück
getValue	Gibt den Wert eines Punkts wieder Lebt die Zelle oder nicht (true, false)
checkFieldExists	Überprüft, ob ein Feld bereits existiert

Im Folgenden werden die entwickelten Klassen näher erläutert.

GridArray und GridPoints

Enthalten die *Grid*-Klasse als Oberklasse und das *IGrid* als Interface. Zusätzlich sind folgende Methoden implementiert:

Tabelle 4 – GridArray- und GridPoints-Methoden

Name der Methode	**Beschreibung**
createGrid	Erstellt ein Spielfeld
fillGrid	Füllt das Spielfeld initial mit Nullen

GridFactory

Diese Factory wird verwendet, um mehrere Objektinstanzen erzeugen zu können, erbt aus dem Interface *IGrid* und nutzt die Methode *getGrid,* um das aktuelle Spielfeld zu bekommen.

Logger

Die Ausgabe der Simulation soll in eine Log-Datei erfolgen. Dafür wird diese Klasse mit folgenden Methoden verwendet:

Tabelle 5 – Logger-Methoden

Name der Methode	**Beschreibung**
addGridToLog	Fügt einen String hinzu und ruft dann writeLogToFile oder writeLogToConsole auf
createFile	Erstellt eine Log-Datei

getCurrentTime	Holt sich die aktuelle Zeit für den Namen der Logfile
getFile	Gibt die File zurück
writeLogToConsole	Schreibt den Log-Inhalt in die Konsole
writeLogToFile	Schreibt den Log-Inhalt in eine Datei

CopyGrid

Hiermit wird eine Kopie eines Spielfeldes erstellt. Dies ist notwendig, da sonst nur Pointer existieren würden.

Parity

Diese Klasse überprüft mithilfe der Methode *step,* ob ein Punkt "tot" oder "lebendig" ist.

GameOfLife

Enthält die Spiellogik für den Spielmodus des Game of Life, hat wie *Parity* die *step*-Methode und zusätzlich wegen weiteren Spielregeln eine erweiterte Logik in der Methode *getPointStatus*.

GameTypeFactory

Eine Fabrik, die eine Methode *getGameType* enthält, mit dessen Hilfe Instanzen von *Parity* und *GameOfLife* erstellt werden.

ConfigReader

Liest die Konfigurationsdatei aus und ist für die Anpassung von Eigenschaften zuständig. Mit den unterschiedlichen GET-Methoden können die Eigenschaften einzelner Werte zurückgegeben werden.

ConfigValidator

Diese Klasse besitzt die Methode *validateConfig*, die ein Property-Objekt von dem *ConfigReader* bekommt und prüft, ob alle Felder da sind und wenn ja, ob sie auch mit validen Daten gesetzt sind.

Main

Diese Klasse wird verwendet, um das Programm initial zu starten. Diese enthält die Methode *getPoints*, die ein Userinterface zur Eingabe von aktiven Punkten erzeugt.

Game

Diese Klasse ist das Kernstück des Spiels. Diese fasst die Spiellogik zusammen und erstellt das Spiel.

Tabelle 6 – Game-Methoden

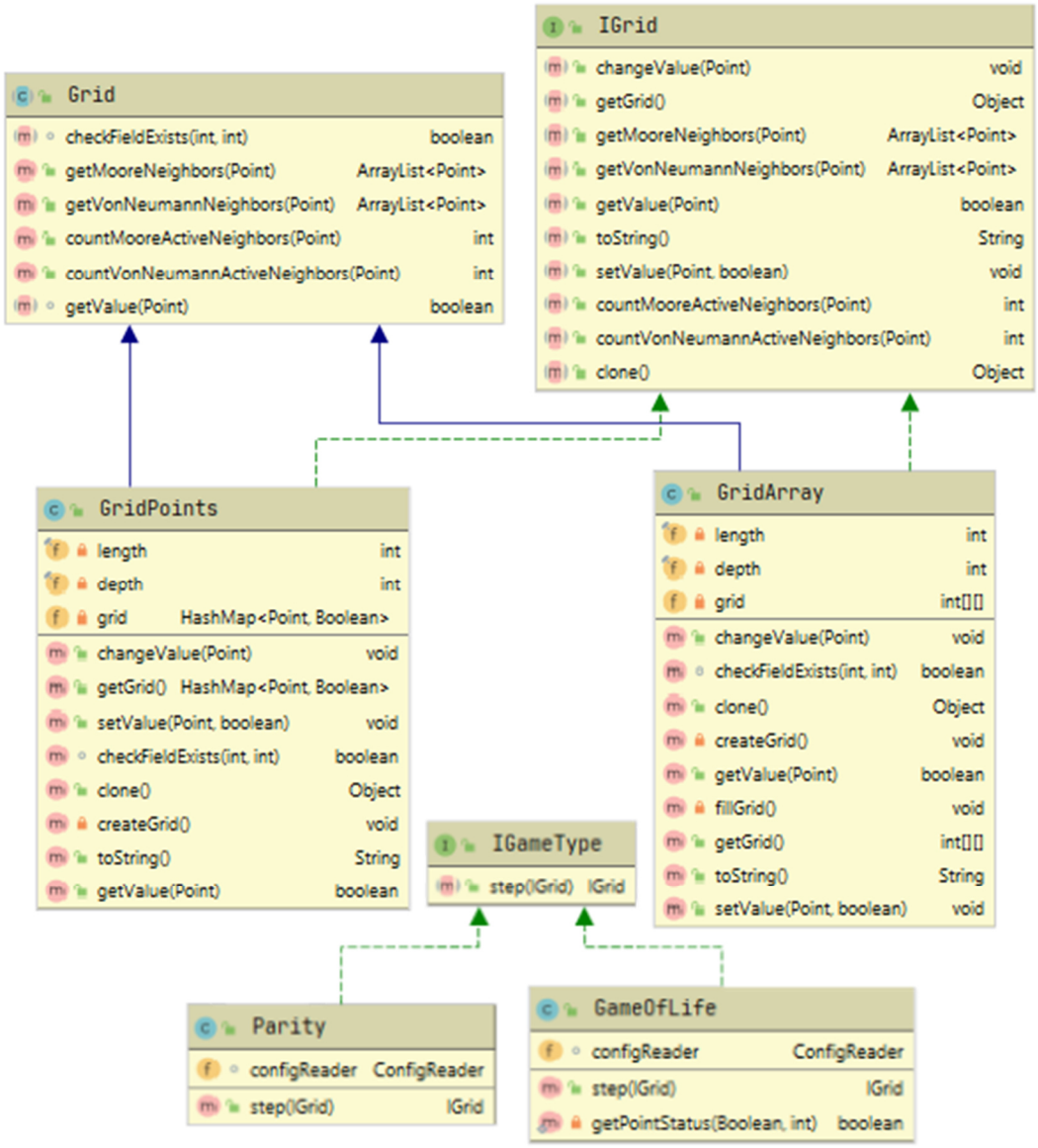
Name der Methode	**Beschreibung**
getContinueGame	Gibt zurück, ob das Spiel vorbei ist (true, false)
setContinueGame	Legt fest, ob das Spiel vorbei ist
checkForTermination	Liest aus dem ConfigReader, welche Bedingungen erfüllt sein müssen, und nach diesen Bedinungen wird ein Abbruch des Spiels ausgelöst
reduceTimeToLive	Reduziert die Lebenszeit, zählt herunter
processTimeToLive	Ruft die Methoden reduceTimeToLive und isTimeToLiveZero auf

isTimeToLiveZero	Gibt zurück, ob die die Lebenszeit den Wert Null entspricht
gridChange	Überprüft, ob das Spielfeld verändert wurde
setGrid	Überschreibt das Spielfeld mit aktuellen Werten
run	Startet das Spiel. Startet den GameOfLife oder Parity Spielmodus mit dem Game-Objekt und dem Spielfeld

3. Wart-, Wiederverwend- und Erweiterbarkeit

Interfaces dienen als Schnittstelle, um festzulegen welche Methoden die einzelnen Klassen enthalten müssen. Daher enthält ein Interface nur Konstanten und Funktionsköpfe. Dies ist sinnvoll, um die Wart-, Wiederverwend- und Erweiterbarkeit zu gewährleisten. Dafür nutzen wir die Interfaces *IGrid* und *IGameType*. Diese beinhalten die gewollten Methoden und werden an die gewünschten Klassen *GridArray*, *GridPoints*, *GameOfLife*, *Parity*, *GameTypeFactory* und *GridFactory* vererbt. Dazu nutzen wir Factories wie *GameTypeFactory* und *GridFactory*. Wenn man mehrere Instanzen eines Objekts schnell erstellen möchte, verwendet man eine Factory. Eine Factory ist eine Funktion, die ein Objekt zurückgibt und wiederverwendet werden kann, um viele Objektinstanzen zu erzeugen. Mit der Hilfe von Interfaces und Factories gewährleisten wir Wartbarkeit, Wiederverwendbarkeit und Erweiterbarkeit.

In der folgenden Abbildung sind die Beziehungen zwischen Interfaces und Klassen verdeutlicht:



4. Installations- und Nutzungsanleitung

Für die Nutzung der Anwendung wird Java 8 benötigt. Zusätzlich wird für die Ausführung JetBrains IntelliJ IDEA (Version 2020.2.3) benötigt. Um die Anwendung auszuführen, wird diese in IntelliJ importiert (***New->Project from version control***) und mit der Projekt-URL (https://gitlab2.nordakademie.de/ChristianApsel-A18/hausarbeit_i143_gamradt_cc.git) importiert. Anschließend wird über Maven der Befehl "*mvn clean install*" ausgeführt. Dadurch wird sichergestellt, dass die nötigen Abhängigkeiten vorhanden sind. Danach können beliebige Exprimente (src\main\java\de\nordakademie\pdse\experiments) ausgeführt werden oder (src\main\java\Main.java) mit beliebigen Einstellungen gestartet werden. Die Einstellungen werden in der config.properties festgelegt. Nun wird die Main gestartet. Die Zellenwerte sind initial auf „tot" gesetzt und können über die Abfrage aus dem Programm verändert werden. Mit dem Befehl „*run*" wird das Spiel gestartet. Die Ausgabe wird bei entsprechender Konfiguration in dem Stammverzeichnis erzeugt.

5. Selbstreflexion der Gruppenmitglieder

i. Christian Apsel

Die Arbeit über Distanz lief zumindest aus meiner Sicht ohne Probleme. Es gab mit dem Pair-Programming soweit mir bekannt auch keine Probleme, auch wenn es schwieriger ist, wirklich Test-first zu arbeiten, gerade wenn eine andere Person eine Methode benötigt, welche zu diesem Zeitpunkt noch nicht implementiert wurde. Die Aufgabenstellungen finde ich an der ein oder anderen Stelle etwas schwach formuliert. Ein weiteres Problem war, dass Gitlab über 24 stunden offline war, was im Zeitmanagement zu Problemen geführt hat.

ii. Fabian Forthmann

Meiner Meinung nach lief die Hausarbeit im Großen und Ganzen sehr gut, obwohl wir uns kein einziges Mal persönlich, sondern immer nur digital getroffen haben. Besonders sinnvoll fand ich unsere ausgiebige Planungsphase am Anfang des Projektes, wo wir uns die Architektur und die Arbeitsverteilung für unterschiedliche Funktionsbereiche des Programms überlegt haben. Positiv ist mir außerdem unsere Entscheidung aufgefallen, in Zweierpaaren zu programmieren. Herr Petersen und ich hatten uns nach kurzer Zeit als Team eingespielt, und ich hatte das Gefühl, dass wir zusammen effektiver besseren Quellcode programmiert haben, als wir es vielleicht allein gemacht hätten. Auch die Aufgabenstellung war aus meiner Sicht nach etwas Überlegen gut schaffbar. Bei meiner nächsten Hausarbeit würde ich die Planungsphase noch ausführlicher gestalten und so beispielsweise bereits definieren, welche Methoden genau in eine Klasse gehören. Außerdem könnte es aus meiner Sicht sinnvoll sein, Deadlines für bestimmte Meilensteile zu setzen, um das Warten auf benötigte Codestellen zu minimieren, und es könnte sinnvoll sein, bereits im Vorhinein Standards für die Ausdrucksweise im Code zu besprechen.

iii. Georg Mezlaw

Die Gruppenarbeit lief meiner Meinung nach gut. Trotz der örtlichen Distanz konnten wir uns virtuell verabreden. Das Pair-Programming lief ebenfalls gut. Wir haben uns geholfen und unterstützt. Die Aufteilung war meiner Meinung nach, auch im Nachhinein, sinnvoll, da so gezielt ohne Konfrontationen gearbeitet werden konnte. Trotz der Aufteilungen konnte man Anpassungen am gesamten Quellcode vornehmen. Zwischenzeitlich war das Gitlab der Nordakademie ausgefallen, was uns zuerst gehindert hat weiter zu arbeiten. Das Problem haben wir mit einem privaten Github-Repo gelöst, bis das GitLab wieder erreichbar war. Nächstes Mal würde ich direkt die Premium-Version von IntelliJ nutzen anstatt die Community-Edition, wie zu Beginn. Ich würde mir für das nächste Mal ein besser formatiertes PDF als Aufgabenstellung wünschen, da dieses Dokument nicht durchsuchbar ist bzw. so formatiert ist, dass Worte und Sätze zusammengerutscht sind.

iv. Rane Petersen

Die Gruppenarbeit als auch der Erstellungsprozess war ein Erfolg. Dennoch gibt es Punkte, die sich beikommenden Projekten durchaus verbessern lassen. Dazu zählen u. A. die Ausarbeitung der Architektur. Die Idee für die einzelnen Klassen wurde schnell gefunden, hatte aber nicht ausreichend Tiefe, was sich in den nächsten Schritten als problematisch erwiesen hat. Das fehlende Design der einzelnen Methoden sorgte für die mehrfache Anpassung der Klassen, welche sich durch eine gründlichere Planung hätte vermeiden lassen können. Selbes gilt für die Umsetzung der

Anforderungen. Diese wurden nicht gründlich genug von uns analysiert und sorgten somit ebenfalls für Anpassungen gegen Ende des Projekts, welche sich hätten vermeiden lassen können.

6. Quellen und Hilfsmittel

https://www.baeldung.com/java-testing-system-out-println (23.10.2020)

https://howtodoinjava.com/junit/junit-creating-temporary-filefolder-using-temporaryfolder-rule/ (24.10.2020)

https://alvinalexander.com/java/java-file-exists-directory-exists/ (30.10.2020)

https://www.w3schools.com/java/java_files_delete.asp (30.10.2020)

https://stackoverflow.com/questions/1459656/how-to-get-the-current-time-in-yyyy-mm-dd-hhmisec-millisecond-format-in-java (30.10.2020)

https://stackoverflow.com/questions/19628157/switch-ignore-case-in-java-7 (30.10.2020)

7. Abbildungs- und Tabellenverzeichnis

8. Anhang (Quellcode, Eidesstattliche Erklärungen, vollständiges UML)

Eidesstattliche Erklärung Fabian Forthmann

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder von mir noch von jemand anderem als Prüfungsleistung vorgelegt.

Datum: 29.10.2020 Unterschrift: 

Eidesstattliche Erklärung des Studenten / der Studentin

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder von mir noch von jemand anderem als Prüfungsleistung vorgelegt.

Datum: 02.11.2020 Unterschrift: G. Mezlaw

Eidesstattliche Erklärung des Studenten / der Studentin

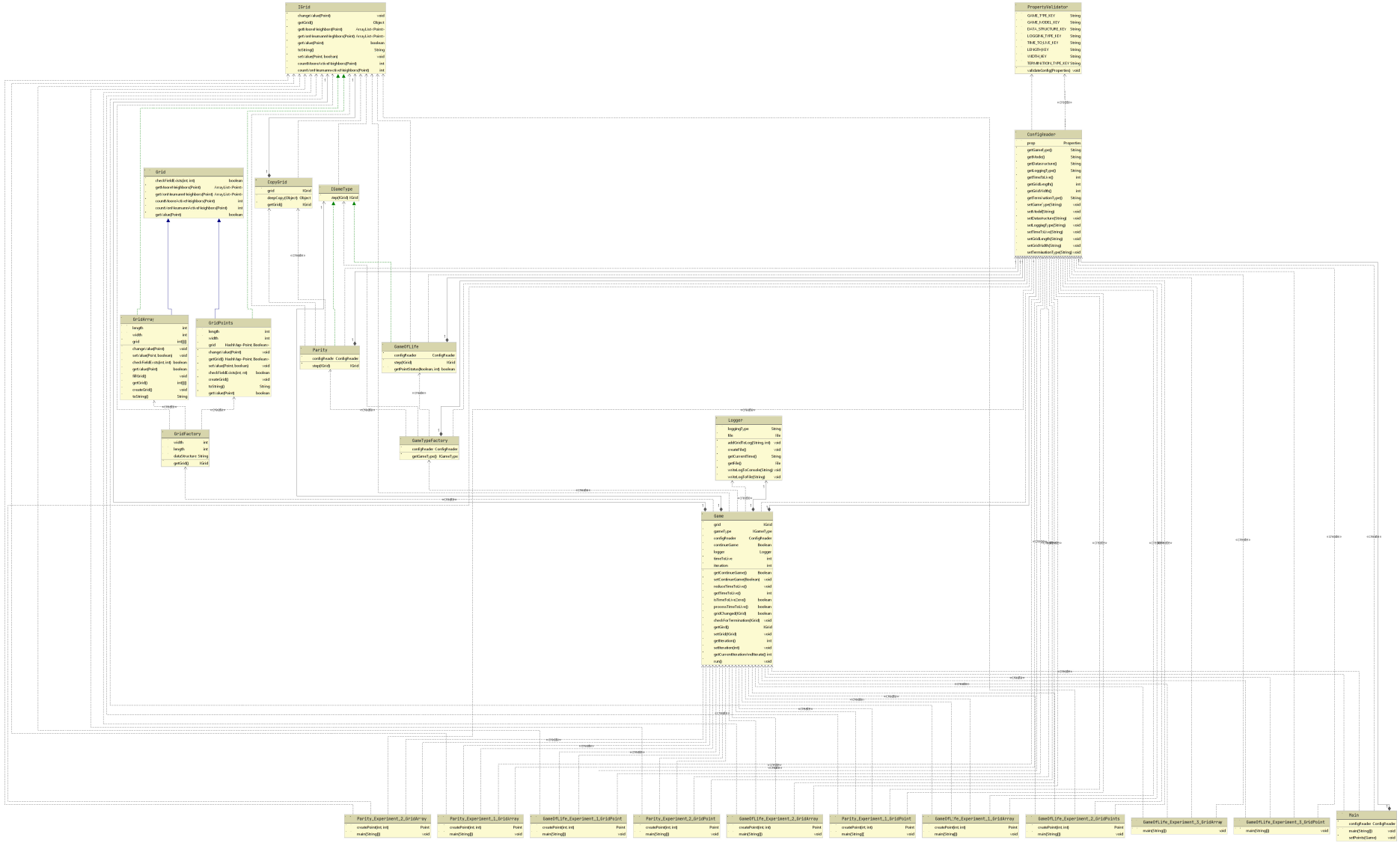
Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder von mir noch von jemand anderem als Prüfungsleistung vorgelegt.

Datum: 01.11.2020 Unterschrift: A. Petersen

Eidesstattliche Erklärung des Studenten / der Studentin

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder von mir noch von jemand anderem als Prüfungsleistung vorgelegt.

Datum: 01.11.2020 Unterschrift: C. Apsel


IGrid
changeValue(Point)
getGrid()
getBoolean(Point)
getNeighbors(Point)
countNeighbors(Point)
countNeighbors(Point, boolean)
toString()
setValue(Point, boolean)
Grid
checkFieldExists(int, int)
getBoolean(Point)
getNeighbors(Point)
countNeighbors(Point)
countNeighbors(Point, boolean)
getValue(Point)
CopyGrid
grid
deepCopy(IObject)
getGrid()
IGameType
stepGrid(IGrid)
GridArray
length
width
grid
changeValue(Point)
setValue(Point, boolean)
checkFieldExists(int, int)
getValue(Point)
fillGrid()
getGrid()
createGrid()
toString()
GridPoints
length
width
grid
changeValue(Point)
getGrid()
setValue(Point, boolean)
checkFieldExists(int, int)
createGrid()
toString()
getValue(Point)
Parity
configHeader
stepGrid(IGrid)
GameOfLife
configHeader
stepGrid(IGrid)
getPointStatus(boolean, int)
GridFactory
width
length
dataStructure
getGrid()
GameTypeFactory
configHeader
getGameType()
Logger
loggingType
file
addGridToLog(String, int)
createFile()
getCurrentTime()
getFile()
writeLogToConsole(String)
writeLogToFile(String)
PropertyValidator
GAME_TYPE_KEY
GAME_MODEL_KEY
DATA_STRUCTURE_KEY
LOGGING_TYPE_KEY
TRUE_TO_LIVE_KEY
LEDGER_KEY
VIDEO_KEY
TERMINATION_TYPE_KEY
validateConfigProperties()
ConfigHeader
prop
getGameType()
getModel()
getDataStructure()
getLoggingType()
getTimeToLive()
getGridLength()
getGridWidth()
getTerminationType()
setGame_type(String)
setModel(String)
setDataStructure(String)
setLoggingType(String)
setTimeToLive(String)
setGridLength(String)
setGridWidth(String)
setTerminationType(String)
Game
grid
gameType
configHeader
continueGame
logger
timeToLive
iteration
getContinueGame()
setContinueGame(Boolean)
reduceTimeToLive()
getTimeToLive()
isTimeToLiveZero()
processTimeToLive()
gridChanged(IGrid)
checkForTermination(IGrid)
getGrid()
setGrid(IGrid)
getIteration()
setIteration(int)
getCurrentIterationAndIterate()
run()
Parity_Experiment_2_GridArray
createPoint(int, int)
main(String[])
Parity_Experiment_1_GridArray
createPoint(int, int)
main(String[])
GameOfLife_Experiment_1_GridPoint
createPoint(int, int)
main(String[])
Parity_Experiment_2_GridPoint
createPoint(int, int)
main(String[])
GameOfLife_Experiment_2_GridArray
createPoint(int, int)
main(String[])
Parity_Experiment_1_GridPoint
createPoint(int, int)
main(String[])
GameOfLife_Experiment_1_GridArray
createPoint(int, int)
main(String[])
GameOfLife_Experiment_2_GridPoints
createPoint(int, int)
main(String[])
GameOfLife_Experiment_3_GridArray
main(String[])
GameOfLife_Experiment_3_GridPoint
main(String[])
Main
configHeader
main(String[])
setPoints(Game)
«create»

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.nordakademie</groupId>
    <artifactId>hausarbeit_i143_gamradt_cc</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>8</source>
                    <target>8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>3.5.13</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.6</version>
        </dependency>
    </dependencies>
</project>
```

```
.idea
*.iml
target

```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<module org.jetbrains.idea.maven.project.MavenProjectsManager.isMavenModule="true" type="JAVA_MODULE" version="4">
  <component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_8">
    <output url="file://$MODULE_DIR$/target/classes" />
    <output-test url="file://$MODULE_DIR$/target/test-classes" />
    <content url="file://$MODULE_DIR$">
      <sourceFolder url="file://$MODULE_DIR$/src/main/java" isTestSource="false" />
      <sourceFolder url="file://$MODULE_DIR$/src/main/resources" type="java-resource" />
      <sourceFolder url="file://$MODULE_DIR$/src/test/java" isTestSource="true" />
      <excludeFolder url="file://$MODULE_DIR$/target" />
    </content>
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
    <orderEntry type="library" scope="TEST" name="Maven: junit:junit:4.12" level="project" />
    <orderEntry type="library" scope="TEST" name="Maven: org.hamcrest:hamcrest-core:1.3" level="project" />
    <orderEntry type="library" scope="TEST" name="Maven: org.mockito:mockito-core:3.5.13" level="project" />
    <orderEntry type="library" scope="TEST" name="Maven: net.bytebuddy:byte-buddy:1.10.15" level="project" />
    <orderEntry type="library" scope="TEST" name="Maven: net.bytebuddy:byte-buddy-agent:1.10.15" level="project" />
    <orderEntry type="library" scope="TEST" name="Maven: org.objenesis:objenesis:3.1" level="project" />
    <orderEntry type="library" name="Maven: commons-io:commons-io:2.6" level="project" />
  </component>
</module>
```

```java
import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;

import java.awt.*;
import java.io.File;
import java.util.Scanner;

/**
 * This class is used to start the program initially. It contains the method getPoints, which creates a user interface for the input of active points.
 *
 * @author Christian Apsel, Rane Petersen
 */
public class Main {
    private ConfigReader configReader;

    public static void main(String[] args) throws Exception {

        Main main = new Main();
        File file;

        if (args.length != 0) {
            file = new File(args[0]);
        } else {
            file = new File("src/main/resources/config.properties");
        }

        main.configReader = new ConfigReader(file);
        Game game = new Game(main.configReader);
        main.setPoints(game);
        game.run();

    }

    private void setPoints(Game game) {
        String run = "";
        Scanner scanner = new Scanner(System.in);
        while (!run.equalsIgnoreCase("run")) {
            System.out.println("Bitte geben Sie aktive Punkte ein: X,Y / Bitte geben Sie run zum starten des Programms ein");
            String temp = scanner.next();
            if (temp.contains(",")) {
                String[] s = temp.split(",");
                if (s.length == 2 && Integer.parseInt(s[0]) < configReader.getGridLength() && Integer.parseInt(s[1]) < configReader.getGridWidth()) {
                    game.getGird().setValue(new Point(Integer.parseInt(s[0]), Integer.parseInt(s[1])), true);
                } else {
                    System.out.println("falsche Eingabe");
                }
            } else if (temp.equalsIgnoreCase("run")) {
                run = "run";
            } else {
                System.out.println("falsche Eingabe");
            }

        }

    }
}
```

```java
package de.nordakademie.pdse.grid;

import java.awt.*;
import java.util.ArrayList;

/**
 * abstract superclass, contains the logic for the playing field, can count neighbors and find out if a field exists
 *
 * @author Georg Mezlaw
 * @since 26.10.2020
 */
public abstract class Grid {

    abstract boolean checkFieldExists(int x, int y);

    public ArrayList<Point> getMooreNeighbors(Point position) {
        ArrayList<Point> neighbors = getVonNeumannNeighbors(position);
        if (checkFieldExists(position.x - 1, position.y - 1)) {
            neighbors.add(new Point(position.x - 1, position.y - 1));
        }
        if (checkFieldExists(position.x - 1, position.y + 1)) {
            neighbors.add(new Point(position.x - 1, position.y + 1));
        }
        if (checkFieldExists(position.x + 1, position.y - 1)) {
            neighbors.add(new Point(position.x + 1, position.y - 1));
        }
        if (checkFieldExists(position.x + 1, position.y + 1)) {
            neighbors.add(new Point(position.x + 1, position.y + 1));
        }
        return neighbors;
    }

    public ArrayList<Point> getVonNeumannNeighbors(Point position) {
        ArrayList<Point> neighbors = new ArrayList<>();
        if (checkFieldExists(position.x + 1, position.y)) {
            neighbors.add(new Point(position.x + 1, position.y));
        }
        if (checkFieldExists(position.x - 1, position.y)) {
            neighbors.add(new Point(position.x - 1, position.y));
        }
        if (checkFieldExists(position.x, position.y + 1)) {
            neighbors.add(new Point(position.x, position.y + 1));
        }
        if (checkFieldExists(position.x, position.y - 1)) {
            neighbors.add(new Point(position.x, position.y - 1));
        }
        return neighbors;
    }

    public int countMooreActiveNeighbors(Point position) {
        return (int) getMooreNeighbors(position).stream().filter(this::getValue).count();
    }

    public int countVonNeumannActiveNeighbors(Point position) {
        return (int) getVonNeumannNeighbors(position).stream().filter(this::getValue).count();
    }

    abstract boolean getValue(Point position);
}

```

```java
package de.nordakademie.pdse.grid;

import java.awt.*;
import java.io.Serializable;
import java.util.ArrayList;

/**
 * contains important information for the playing field. Can determine if a cell is alive or dead, return the playing field, return neighbors and count the neighbors
 *
 * @author Georg Mezlaw
 * @since 26.10.2020
 */
public interface IGrid extends Serializable {
    void changeValue(Point position);

    Object getGrid();

    ArrayList<Point> getMooreNeighbors(Point point);

    ArrayList<Point> getVonNeumannNeighbors(Point point);

    boolean getValue(Point position);

    String toString();

    void setValue(Point position, boolean value);

    public int countMooreActiveNeighbors(Point position);

    public int countVonNeumannActiveNeighbors(Point position);

}

```

```java
package de.nordakademie.pdse.grid;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

/**
 * This is used to create a copy of a playing field. This is necessary, because otherwise only pointers would exist
 *
 * @author Christian Apsel
 * @since 26.10.2020
 */
public class CopyGrid {
    private final IGrid grid;

    public CopyGrid(IGrid grid) {
        this.grid = (IGrid) deepCopy(grid);
    }

    private Object deepCopy(Object object) {
        try {
            ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
            ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
            objectOutputStream.writeObject(object);
            ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(byteArrayOutputStream.toByteArray());
            ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
            return objectInputStream.readObject();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    public IGrid getGrid() {
        return grid;
    }
}

```

```java
package de.nordakademie.pdse.grid;

import java.awt.*;
import java.util.Arrays;

/**
 * Contains the grid class as superclass and the IGrid as interface. Can create a playing field.
 *
 * @author Christian Apsel
 * @since 26.10.2020
 */
public class GridArray extends Grid implements IGrid {
    private final int length;
    private final int width;
    private int[][] grid;

    public GridArray(int length, int width) {
        this.length = length;
        this.width = width;
        createGrid();
    }

    @Override
    public void changeValue(Point position) {
        int width = position.x;
        int length = position.y;

        if (getGrid()[width][length] == 0) {
            getGrid()[width][length] = 1;
        } else {
            getGrid()[width][length] = 0;
        }
    }

    @Override
    public void setValue(Point position, boolean value) {
        int width = position.x;
        int length = position.y;
        grid[width][length] = value ? 1 : 0;
    }


    @Override
    boolean checkFieldExists(int x, int y) {
        return x >= 0 && x <= width - 1 && y >= 0 && y <= length - 1;
    }

    @Override
    public boolean getValue(Point position) {
        int width = position.x;
        int length = position.y;
        return getGrid()[width][length] != 0;
    }

    private void fillGrid() {
        for (int[] length : grid) {
            Arrays.fill(length, 0);
        }
    }

    @Override
    public int[][] getGrid() {
        return grid;
```

```
64      }
65
66      private void createGrid() {
67          grid = new int[width][length];
68          fillGrid();
69      }
70
71      //TODO Duplicate Code entfernen
72      @Override
73      public String toString() {
74          StringBuilder stringBuilder = new StringBuilder();
75          for (int i = 0; i < width; i++) {
76
77              for (int j = 0; j < length; j++) {
78
79                  if (getValue(new Point(i, j))) {
80                      stringBuilder.append(1);
81                  } else {
82                      stringBuilder.append(0);
83                  }
84              }
85              stringBuilder.append("\n");
86          }
87          return stringBuilder.toString();
88      }
89  }
90
```

```java
package de.nordakademie.pdse.grid;

import java.awt.*;
import java.util.HashMap;
/**
 * Contains the grid class as superclass and the IGrid as interface. Can create a playing field.
 *
 * @author Christian Apsel
 * @since 26.10.2020
 */
public class GridPoints extends Grid implements IGrid {
    private final int length;
    private final int width;
    private HashMap<Point, Boolean> grid;

    public GridPoints(int length, int width) {
        this.length = length;
        this.width = width;
        createGrid();
    }

    @Override
    public void changeValue(Point position) {
        if (getGrid().get(position).equals(false)) {
            getGrid().replace(position, true);
        } else {
            getGrid().replace(position, false);
        }
    }

    @Override
    public HashMap<Point, Boolean> getGrid() {
        return grid;
    }

    @Override
    public void setValue(Point position, boolean value) {
        getGrid().replace(position, value);
    }

    @Override
    boolean checkFieldExists(int x, int y) {
        return x >= 0 && x <= width - 1 && y >= 0 && y <= length - 1;
    }

    private void createGrid() {
        grid = new HashMap<>();
        for (int i = 0; i < width; i++) {
            for (int j = 0; j < length; j++) {
                grid.put(new Point(i, j), false);
            }
        }
    }

    //TODO Duplicate Code entfernen
    @Override
    public String toString() {
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < width; i++) {
            for (int j = 0; j < length; j++) {

                if (getValue(new Point(i, j))) {
                    stringBuilder.append(1);
```

```
64                  } else {
65                      stringBuilder.append(0);
66                  }
67              }
68              stringBuilder.append("\n");
69          }
70          return stringBuilder.toString();
71      }
72
73      @Override
74      public boolean getValue(Point position) {
75          return getGrid().get(position);
76      }
77  }
78
```

```java
package de.nordakademie.pdse.grid;

/**
 * This factory is used to create multiple object instances, inherits from the interface IGrid and uses the method getGrid to get the current grid.
 *
 * @author Christian Apsel
 * @since 26.10.2020
 */
public class GridFactory {
    private final int width;
    private final int length;
    private final String dataStructure;

    public GridFactory(int length, int width, String dataStructure) {
        this.width = width;
        this.length = length;
        this.dataStructure = dataStructure;
    }

    public IGrid getGrid() {
        if ("GridArray".equalsIgnoreCase(dataStructure)) {
            return new GridArray(length, width);
        } else {
            return new GridPoints(length, width);
        }
    }
}

```

```java
package de.nordakademie.pdse.config;

import java.io.File;
import java.io.FileInputStream;
import java.util.Properties;

/**
 * Reads the configuration file and is responsible for the adjustment of properties.
 * With the different GET methods the properties of single values can be returned.
 *
 * @author Fabian Forthmann
 */
public class ConfigReader {

    private final Properties prop = new Properties();

    public ConfigReader(File propertyFile) {
        try {
            FileInputStream inputStream = new FileInputStream(propertyFile);
            prop.load(inputStream);

            PropertyValidator propertyValidator = new PropertyValidator();
            propertyValidator.validateConfig(prop);
        } catch (Exception e) {
            throw new RuntimeException("The properties file could not be found or loaded.");
        }
    }

    public String getGameType() {
        return prop.getProperty("game.type");
    }

    public String getModel() {
        return prop.getProperty("game.model");
    }

    public String getDatastructure() {
        return prop.getProperty("game.dataStructure");
    }

    public String getLoggingType() {
        return prop.getProperty("game.loggingType");
    }

    public int getTimeToLive() {
        return Integer.parseInt(prop.getProperty("game.timeToLive"));
    }

    public int getGridLength() {
        return Integer.parseInt(prop.getProperty("game.grid.length"));
    }

    public int getGridWidth() {
        return Integer.parseInt(prop.getProperty("game.grid.width"));
    }

    public String getTerminationType() {
        return prop.getProperty("game.terminationType");
    }

    public void setGameType(String gameType) {
```

```
62          prop.setProperty("game.type", gameType);
63      }
64
65      public void setModel(String gameModel) {
66          prop.setProperty("game.model", gameModel);
67      }
68
69      public void setDatastructure(String datastructure) {
70          prop.setProperty("game.dataStructure", datastructure);
71      }
72
73      public void setLoggingType(String loggingType) {
74          prop.setProperty("game.loggingType", loggingType);
75      }
76
77      public void setTimeToLive(String timeToLive) {
78          prop.setProperty("game.timeToLive", timeToLive);
79      }
80
81      public void setGridLength(String gridLength) {
82          prop.setProperty("game.grid.length", gridLength);
83      }
84
85      public void setGridWidth(String gridWidth) {
86          prop.setProperty("game.grid.width", gridWidth);
87      }
88
89      public void setTerminationType(String terminationType) {
90          prop.setProperty("game.terminationType", terminationType);
91      }
92  }
93
```

```java
package de.nordakademie.pdse.config;

import java.util.Properties;

/**
 * This class gets a property object from the ConfigReader and checks if all fields are there and if yes, if they are set with valid data.
 *
 * @author Fabian Forthmann
 */
public class PropertyValidator {

    private static final String GAME_TYPE_KEY = "game.type";
    private static final String GAME_MODEL_KEY = "game.model";
    private static final String DATA_STRUCTURE_KEY = "game.dataStructure";
    private static final String LOGGING_TYPE_KEY = "game.loggingType";
    private static final String TIME_TO_LIVE_KEY = "game.timeToLive";
    private static final String LENGTH_KEY = "game.grid.length";
    private static final String WIDTH_KEY = "game.grid.width";
    private static final String TERMINATION_TYPE_KEY = "game.terminationType";

    public void validateConfig(Properties prop) {
        boolean validDataInFile = true;
        try {
            if ("ttlOrNoChange".equalsIgnoreCase(prop.getProperty(TERMINATION_TYPE_KEY)) || "ttl".equalsIgnoreCase(prop.getProperty(TERMINATION_TYPE_KEY))) {
                if (Integer.parseInt(prop.getProperty(TIME_TO_LIVE_KEY)) < 0) {
                    validDataInFile = false;
                }
            } else if (!"noChange".equalsIgnoreCase(prop.getProperty(TERMINATION_TYPE_KEY))) {
                validDataInFile = false;
            }
            if (!"Parity".equalsIgnoreCase(prop.getProperty(GAME_TYPE_KEY)) && !"GameOfLife".equalsIgnoreCase(prop.getProperty(GAME_TYPE_KEY))) {
                validDataInFile = false;
            }
            if (!"Moore".equalsIgnoreCase(prop.getProperty(GAME_MODEL_KEY)) && !"vonNeumann".equalsIgnoreCase(prop.getProperty(GAME_MODEL_KEY))) {
                validDataInFile = false;
            }
            if (!"GridPoints".equalsIgnoreCase(prop.getProperty(DATA_STRUCTURE_KEY)) && !"GridArray".equalsIgnoreCase(prop.getProperty(DATA_STRUCTURE_KEY))) {
                validDataInFile = false;
            }
            if (!"console".equalsIgnoreCase(prop.getProperty(LOGGING_TYPE_KEY)) && !"file".equalsIgnoreCase(prop.getProperty(LOGGING_TYPE_KEY)) && !"consoleAndFile".equalsIgnoreCase(prop.getProperty(LOGGING_TYPE_KEY)) && !"disable".equalsIgnoreCase(prop.getProperty(LOGGING_TYPE_KEY))) {
                validDataInFile = false;
            }
            if (Integer.parseInt(prop.getProperty(LENGTH_KEY)) < 1) {
                validDataInFile = false;
            }
            if (Integer.parseInt(prop.getProperty(WIDTH_KEY)) < 1) {
                validDataInFile = false;
            }
        } catch (RuntimeException e) {
            throw new RuntimeException("The given property file is incomplete.", e);
        }
        if (!validDataInFile) {
```

```
53              throw new RuntimeException("The given property file includes
   invalid data.");
54          }
55      }
56  }
57
```

```java
package de.nordakademie.pdse.logging;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

/**
 * Returns the output of the simulation to a log file or the console
 *
 * @author Georg Mezlaw
 * @since 26.10.2020
 */
public class Logger {
    private final String loggingType;
    private File file;
    private String fileName;

    public Logger(String loggingType, String fileName) {
        this.loggingType = loggingType;
        if (!"console".equalsIgnoreCase(this.loggingType)) {
            this.fileName = fileName + ".log";
            createFile();
        }

    }

    public Logger(String loggingType) {
        this.loggingType = loggingType;
        if (!"console".equalsIgnoreCase(this.loggingType)) {
            this.fileName = "GridLog" + getCurrentTime() + ".log";
            createFile();
        }
    }

    public void addGridToLog(String grid, int iteration) {
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("###" + iteration + "\n");
        stringBuilder.append(grid);
        switch (loggingType.toLowerCase()) {
            case "consoleandfile":
                writeLogToFile(stringBuilder.toString());
                writeLogToConsole(stringBuilder.toString());
                break;
            case "file":
                writeLogToFile(stringBuilder.toString());
                break;
            case "console":
                writeLogToConsole(stringBuilder.toString());
                break;
            case "disable":
                break;
        }
    }

    private void createFile() {
        Path path = Paths.get(fileName);
        try {
            Files.createFile(path);
            file = new File(String.valueOf(path));
```

```
65
66          } catch (IOException ignored) {
67              file = new File(String.valueOf(path));
68          }
69      }
70
71      private String getCurrentTime() {
72          DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("
    yyyy_MM_dd_HH-mm-ss-SSS");
73          LocalDateTime localDateTime = LocalDateTime.now();
74          return dateTimeFormatter.format(localDateTime);
75      }
76
77      public File getFile() {
78          return file;
79      }
80
81      private void writeLogToConsole(String input) {
82          System.out.println(input);
83      }
84
85      private void writeLogToFile(String input) {
86          try {
87              FileWriter fileWriter = new FileWriter(getFile(), true);
88              fileWriter.append(input);
89              fileWriter.append(System.lineSeparator());
90              fileWriter.close();
91          } catch (IOException ignored) {
92          }
93      }
94 }
95
```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.CopyGrid;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import de.nordakademie.pdse.logging.Logger;

/**
 * It summarizes the game logic and creates the game.
 *
 * @author Rane Petersen, Fabian Forthmann
 * @since 24.10.2020
 */
public class Game {

    IGrid grid;
    IGameType gameType;
    ConfigReader configReader;
    Boolean continueGame;
    Logger logger;
    int timeToLive;
    int iteration;


    public Game(ConfigReader configReader) {
        this.configReader = configReader;
        this.timeToLive = configReader.getTimeToLive();
        this.grid = new GridFactory(configReader.getGridLength(), configReader.getGridWidth(), configReader.getDatastructure()).getGrid();
        this.gameType = new GameTypeFactory(configReader).getGameType();
        this.logger = new Logger(configReader.getLoggingType());
        this.iteration = 0;
        if (configReader.getTerminationType().equalsIgnoreCase("ttl")) {
            this.continueGame = timeToLive > 0;
        } else {
            this.continueGame = true;
        }
    }

    public Game(ConfigReader configReader, String logFileName) {
        this.configReader = configReader;
        this.timeToLive = configReader.getTimeToLive();
        this.grid = new GridFactory(configReader.getGridLength(), configReader.getGridWidth(), configReader.getDatastructure()).getGrid();
        this.gameType = new GameTypeFactory(configReader).getGameType();
        this.logger = new Logger(configReader.getLoggingType(), logFileName);
        this.iteration = 0;
        if (configReader.getTerminationType().equals("ttl")) {
            this.continueGame = timeToLive > 0;
        } else {
            this.continueGame = true;
        }
    }

    private Boolean getContinueGame() {
        return continueGame;
    }

    private void setContinueGame(Boolean continueGame) {
        this.continueGame = continueGame;
    }

    private void reduceTimeToLive() {
```

```
 63          timeToLive--;
 64      }
 65
 66      private int getTimeToLive() {
 67          return timeToLive;
 68      }
 69
 70      private boolean isTimeToLiveZero() {
 71          if (getTimeToLive() > 0) {
 72              return false;
 73          } else {
 74              return true;
 75          }
 76      }
 77
 78      private boolean processTimeToLive() {
 79          reduceTimeToLive();
 80          return isTimeToLiveZero();
 81      }
 82
 83      private boolean gridChanged(IGrid grid) {
 84          if (this.grid.toString().equals(grid.toString())) {
 85              return false;
 86          } else {
 87              return true;
 88          }
 89      }
 90
 91      private void checkForTermination(IGrid newGrid) {
 92          switch (configReader.getTerminationType().toLowerCase()) {
 93              case "ttl":
 94                  this.setContinueGame(!processTimeToLive());
 95                  break;
 96              case "nochange":
 97                  this.setContinueGame(gridChanged(newGrid));
 98                  break;
 99              case "ttlornochange":
100                  this.setContinueGame(!processTimeToLive() && this.gridChanged(
    newGrid));
101                  break;
102          }
103      }
104
105      public IGrid getGird() {
106          return grid;
107      }
108
109      public void setGrid(IGrid grid) {
110          this.grid = grid;
111      }
112
113      private int getIteration() {
114          return iteration;
115      }
116
117      private void setIteration(int iteration) {
118          this.iteration = iteration;
119      }
120
121      private int getCurrentIterationAndIterate() {
122          int i = this.getIteration();
123          i++;
124          this.setIteration(i);
125          i--;
```

```
126         return i;
127     }
128 
129     public void run() throws Exception {
130         logger.addGridToLog(getGird().toString(),
    getCurrentIterationAndIterate());
131         while (this.getContinueGame()) {
132             CopyGrid copyGrid = new CopyGrid(getGird());
133             IGrid newGrid = gameType.step(copyGrid.getGrid());
134             checkForTermination(newGrid);
135             if (gridChanged(newGrid) && !this.configReader.getTerminationType
    ().equalsIgnoreCase("ttl")) {
136                 this.setGrid(newGrid);
137                 logger.addGridToLog(getGird().toString(),
    getCurrentIterationAndIterate());
138             } else if (this.configReader.getTerminationType().equalsIgnoreCase
    ("ttl")){
139                 this.setGrid(newGrid);
140                 logger.addGridToLog(getGird().toString(),
    getCurrentIterationAndIterate());
141             }
142         }
143     }
144 }
145 
```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.CopyGrid;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;

/**
 * This class uses the step method to check whether a point is "dead" or "alive".
 *
 * @author Rane Petersen
 */
public class Parity implements IGameType {
    ConfigReader configReader;

    public Parity(ConfigReader configReader) {
        this.configReader = configReader;
    }

    public IGrid step(IGrid oldGrid) {
        CopyGrid copyGrid = new CopyGrid(oldGrid);
        IGrid newGrid = oldGrid;
        if ("vonNeumann".equalsIgnoreCase(configReader.getModel())) {
            Point point;
            for (int length = 0; length < configReader.getGridLength(); length++) {
                for (int width = 0; width < configReader.getGridWidth(); width++) {
                    point = new Point(length, width);
                    newGrid.setValue(point, copyGrid.getGrid().countVonNeumannActiveNeighbors(point) % 2 == 1);
                }
            }
        }
        return newGrid;
    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.grid.IGrid;

/**
 * It goes over the playing field and starts the next iteration of the game. It checks for each cell how it changes and writes the result into a new game iteration
 *
 * @author Rane Petersen
 */
public interface IGameType {

    public IGrid step(IGrid oldGrid) throws Exception;
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.CopyGrid;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;

/**
 * Contains the game logic for the game mode of the Game of Life, has like Parity the step method and additionally because of further game rules an extended logic in the method getPointStatus
 *
 * @author Rane Petersen
 */
public class GameOfLife implements IGameType {
    ConfigReader configReader;

    public GameOfLife(ConfigReader configReader) {
        this.configReader = configReader;
    }

    @Override
    public IGrid step(IGrid oldGrid) {
        CopyGrid copyGrid = new CopyGrid(oldGrid);
        IGrid newGrid = oldGrid;
        Point point;
        if ("Moore".equalsIgnoreCase(configReader.getModel())) {
            for (int width = 0; width < configReader.getGridLength(); width ++) {
                for (int length = 0; length < configReader.getGridWidth(); length++) {
                    point = new Point(length, width);
                    newGrid.setValue(point, getPointStatus(copyGrid.getGrid().getValue(point), copyGrid.getGrid().countMooreActiveNeighbors(point)));
                }
            }
        } else if ("vonNeumann".equalsIgnoreCase(configReader.getModel())) {
            for (int width = 0; width < configReader.getGridLength(); width ++) {
                for (int length = 0; length < configReader.getGridWidth(); length++) {
                    point = new Point(length, width);
                    newGrid.setValue(point, getPointStatus(copyGrid.getGrid().getValue(point), copyGrid.getGrid().countVonNeumannActiveNeighbors(point)));
                }
            }
        }
        return newGrid;
    }

    private static boolean getPointStatus(Boolean point, int aliveNeighbors) {

        if (point.equals(true)) {
            point = aliveNeighbors == 2 || aliveNeighbors == 3;
        } else {
            if (aliveNeighbors == 3) {
                point = true;
            }
        }
        return point;
    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;

/**
 * A factory that contains a getGameType method that is used to create instances of Parity and GameOfLife.
 *
 * @author Rane Petersen
 */
public class GameTypeFactory {
    ConfigReader configReader;

    public GameTypeFactory(ConfigReader configReader) {
        this.configReader = configReader;
    }

    public IGameType getGameType() {
        if (configReader.getGameType().equalsIgnoreCase("Parity")) {
            return new Parity(configReader);
        } else {
            return new GameOfLife(configReader);
        }
    }
}

```

```java
package de.nordakademie.pdse.experiments.parity;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class Parity_Experiment_1_GridArray {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("400");
        configReader.setGridWidth("400");
        configReader.setModel("vonNeumann");
        configReader.setGameType("Parity");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridArray");
        Game game = new Game(configReader,"Parity_Experiment_1_GridArray");
        IGrid grid = game.getGird();
        grid.setValue(createPoint(200, 200), true);
        grid.setValue(createPoint(200, 201), true);
        grid.setValue(createPoint(201, 200), true);
        grid.setValue(createPoint(201, 201), true);
        game.run();
    }
}

```

```java
package de.nordakademie.pdse.experiments.parity;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class Parity_Experiment_1_GridPoint {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("400");
        configReader.setGridWidth("400");
        configReader.setModel("vonNeumann");
        configReader.setGameType("Parity");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridPoint");
        Game game = new Game(configReader,"Parity_Experiment_1_GridPoint");
        IGrid grid = game.getGird();
        grid.setValue(createPoint(200, 200), true);
        grid.setValue(createPoint(200, 201), true);
        grid.setValue(createPoint(201, 200), true);
        grid.setValue(createPoint(201, 201), true);
        game.run();
    }
}

```

```java
package de.nordakademie.pdse.experiments.parity;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class Parity_Experiment_2_GridArray {
    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("100");
        configReader.setGridWidth("100");
        configReader.setModel("vonNeumann");
        configReader.setGameType("Parity");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridArray");
        Game game = new Game(configReader,"Parity_Experiment_2_GridArray");
        IGrid grid = game.getGird();
        for (int length = 0; length < configReader.getGridLength(); length++) {
            for (int width = 0; width < configReader.getGridWidth(); width++) {
                if (width % 2 == 0) {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), false);
                    } else {
                        grid.setValue(createPoint(length, width), true);
                    }
                } else {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), true);
                    } else {
                        grid.setValue(createPoint(length, width), false);
                    }
                }
            }
        }
        game.setGrid(grid);
        game.run();
    }
}

```

```java
package de.nordakademie.pdse.experiments.parity;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class Parity_Experiment_2_GridPoint {
    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("100");
        configReader.setGridWidth("100");
        configReader.setModel("vonNeumann");
        configReader.setGameType("Parity");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridPoints");
        Game game = new Game(configReader,"Parity_Experiment_2_GridPoint");
        IGrid grid = game.getGird();
        for (int length = 0; length < configReader.getGridLength(); length++) {
            for (int width = 0; width < configReader.getGridWidth(); width++) {
                if (width % 2 == 0) {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), false);
                    } else {
                        grid.setValue(createPoint(length, width), true);
                    }
                } else {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), true);
                    } else {
                        grid.setValue(createPoint(length, width), false);
                    }
                }
            }
        }
        game.setGrid(grid);
        game.run();
    }
}

```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_1_GridArray {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("41");
        configReader.setGridWidth("40");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridArray");
        Game game = new Game(configReader,"GameOfLife_Experiment_1_GridArray");
        IGrid grid = game.getGird();

        grid.setValue(createPoint(18, 18), true);
        grid.setValue(createPoint(18, 19), true);
        grid.setValue(createPoint(18, 21), true);
        grid.setValue(createPoint(18, 22), true);

        grid.setValue(createPoint(19, 18), true);
        grid.setValue(createPoint(19, 19), true);
        grid.setValue(createPoint(19, 21), true);
        grid.setValue(createPoint(19, 22), true);

        grid.setValue(createPoint(20, 19), true);
        grid.setValue(createPoint(20, 21), true);

        grid.setValue(createPoint(21, 17), true);
        grid.setValue(createPoint(21, 19), true);
        grid.setValue(createPoint(21, 21), true);
        grid.setValue(createPoint(21, 23), true);

        grid.setValue(createPoint(22, 17), true);
        grid.setValue(createPoint(22, 19), true);
        grid.setValue(createPoint(22, 21), true);
        grid.setValue(createPoint(22, 23), true);

        grid.setValue(createPoint(23, 17), true);
        grid.setValue(createPoint(23, 18), true);
        grid.setValue(createPoint(23, 22), true);
        grid.setValue(createPoint(23, 23), true);

        game.setGrid(grid);
        game.run();
    }
}
```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_1_GridPoint {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("41");
        configReader.setGridWidth("40");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridPoints");
        Game game = new Game(configReader,"GameOfLife_Experiment_1_GridPoint");
        IGrid grid = game.getGird();
        grid.setValue(createPoint(18, 18), true);
        grid.setValue(createPoint(18, 19), true);
        grid.setValue(createPoint(18, 21), true);
        grid.setValue(createPoint(18, 22), true);

        grid.setValue(createPoint(19, 18), true);
        grid.setValue(createPoint(19, 19), true);
        grid.setValue(createPoint(19, 21), true);
        grid.setValue(createPoint(19, 22), true);

        grid.setValue(createPoint(20, 19), true);
        grid.setValue(createPoint(20, 21), true);

        grid.setValue(createPoint(21, 17), true);
        grid.setValue(createPoint(21, 19), true);
        grid.setValue(createPoint(21, 21), true);
        grid.setValue(createPoint(21, 23), true);

        grid.setValue(createPoint(22, 17), true);
        grid.setValue(createPoint(22, 19), true);
        grid.setValue(createPoint(22, 21), true);
        grid.setValue(createPoint(22, 23), true);

        grid.setValue(createPoint(23, 17), true);
        grid.setValue(createPoint(23, 18), true);
        grid.setValue(createPoint(23, 22), true);
        grid.setValue(createPoint(23, 23), true);

        game.setGrid(grid);
        game.run();
    }
}

```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_2_GridArray {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("100");
        configReader.setGridWidth("100");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridArray");
        Game game = new Game(configReader,"GameOfLife_Experiment_2_GridArray");
        IGrid grid = game.getGird();
        for (int length = 0; length < configReader.getGridLength(); length++) {
            for (int width = 0; width < configReader.getGridWidth(); width++) {
                if (width % 2 == 0) {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), false);
                    } else {
                        grid.setValue(createPoint(length, width), true);
                    }
                } else {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), true);
                    } else {
                        grid.setValue(createPoint(length, width), false);
                    }
                }
            }
        }
        game.setGrid(grid);
        game.run();
    }
}
```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;

import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_3_GridArray {

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("300");
        configReader.setGridWidth("300");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridArray");
        Game game = new Game(configReader,"GameOfLife_Experiment_3_GridArray");
        game.run();
    }

}

```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;

import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_3_GridPoint {

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("300");
        configReader.setGridWidth("300");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridPoints");
        Game game = new Game(configReader,"GameOfLife_Experiment_3_GridPoint");
        game.run();
    }

}

```

```java
package de.nordakademie.pdse.experiments.gameoflife;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.gamelogic.Game;
import de.nordakademie.pdse.grid.IGrid;

import java.awt.*;
import java.io.File;

/**
 * @author Georg Mezlaw, Rane Petersen
 */
public class GameOfLife_Experiment_2_GridPoints {

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    public static void main(String[] args) throws Exception {
        ConfigReader configReader = new ConfigReader(new File("src/main/resources/config.properties"));
        configReader.setGridLength("100");
        configReader.setGridWidth("100");
        configReader.setModel("Moore");
        configReader.setGameType("GameOfLife");
        configReader.setTerminationType("ttl");
        configReader.setTimeToLive("100");
        configReader.setLoggingType("file");
        configReader.setDatastructure("GridPoint");
        Game game = new Game(configReader,"GameOfLife_Experiment_2_GridPoints");
        IGrid grid = game.getGird();
        for (int length = 0; length < configReader.getGridLength(); length++) {
            for (int width = 0; width < configReader.getGridWidth(); width++) {
                if (width % 2 == 0) {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), false);
                    } else {
                        grid.setValue(createPoint(length, width), true);
                    }
                } else {
                    if ((length % 2 == 1)) {
                        grid.setValue(createPoint(length, width), true);
                    } else {
                        grid.setValue(createPoint(length, width), false);
                    }
                }
            }
        }
        game.setGrid(grid);
        game.run();
    }
}
```

```
game.type=Parity
#game.type= "Parity", "GameOfLife"#
game.model=vonNeumann
#game.model= "Moore", "vonNeumann"#
game.dataStructure=GridPoints
#game.dataStructure= "GridPoints", "GridArray"#
game.loggingType=console
#game.loggingType= "console","file","consoleAndFile","disable"#
game.timeToLive=10
game.grid.length=10
game.grid.width=10
game.terminationType=ttl
#game.terminationType= "ttl", "noChange", "ttlOrNoChange"

```

```java
package de.nordakademie.pdse.grid;

import org.junit.Test;

import java.awt.*;

import static org.junit.Assert.assertEquals;

public class GridTest {

    @Test
    public void countMooreActiveNeighbors() {
        GridPoints currentGrid = new GridPoints(4, 4);
        currentGrid.changeValue(new Point(0, 1));
        currentGrid.changeValue(new Point(1, 1));
        assertEquals(2, currentGrid.countMooreActiveNeighbors(new Point(0, 0
));
    }

    @Test
    public void countVonNeumannActiveNeighbors() {
        GridPoints currentGrid = new GridPoints(4, 4);
        currentGrid.changeValue(new Point(0, 1));
        currentGrid.changeValue(new Point(1, 1));
        assertEquals(1, currentGrid.countVonNeumannActiveNeighbors(new Point(0
, 0)));
    }
}

```

```java
package de.nordakademie.pdse.grid;

import org.junit.Test;

import java.awt.*;

import static org.junit.Assert.assertNotEquals;

public class CopyGridTest {
    @Test
    public void testGetGrid() {
        IGrid originalGrid = new GridPoints(5, 5);
        CopyGrid copyGrid = new CopyGrid(originalGrid);
        originalGrid.setValue(new Point(0, 0), true);
        assertNotEquals(originalGrid.getValue(new Point(0, 0)), copyGrid.
getGrid().getValue(new Point(0, 0)));
    }
}

```

```java
package de.nordakademie.pdse.grid;

import org.junit.Test;

import java.awt.*;
import java.util.ArrayList;
import java.util.Arrays;

import static org.junit.Assert.*;

public class GridArrayTest {

    @Test
    public void testCreateGrid() {
        int[][] testGrid = {{0}};
        assertEquals(Arrays.deepToString(testGrid), Arrays.deepToString(new GridArray(1, 1).getGrid()));
    }

    @Test
    public void testChangeValue() {
        int[][] testGrid = {{1}};
        GridArray currentGrid = new GridArray(1, 1);
        currentGrid.changeValue(new Point(0, 0));
        assertEquals(testGrid, currentGrid.getGrid());

    }


    @Test
    public void testGetVonNeumannNeighbors() {
        ArrayList<Point> expectedValues = new ArrayList<>();
        expectedValues.add(new Point(0, 0));
        expectedValues.add(new Point(1, 1));
        expectedValues.add(new Point(2, 0));
        GridArray currentGrid = new GridArray(4, 4);

        assertTrue(expectedValues.containsAll(currentGrid.getVonNeumannNeighbors(new Point(1, 0)))
                && currentGrid.getVonNeumannNeighbors(new Point(1, 0)).size() == expectedValues.size());

    }

    @Test
    public void testGetMooreNeighbors() {
        ArrayList<Point> expectedValues = new ArrayList<>();
        expectedValues.add(new Point(0, 0));
        expectedValues.add(new Point(0, 1));
        expectedValues.add(new Point(1, 1));
        expectedValues.add(new Point(2, 1));
        expectedValues.add(new Point(2, 0));
        GridArray currentGrid = new GridArray(4, 4);

        assertTrue(expectedValues.containsAll(currentGrid.getMooreNeighbors(new Point(1, 0)))
                && currentGrid.getMooreNeighbors(new Point(1, 0)).size() == expectedValues.size());
    }

    @Test
    public void testToString() {
        GridArray currentGrid = new GridArray(4, 4);
        String testString;
```

```
60          testString = "0000" + "\n" + "0000" + "\n" + "0000" + "\n" + "0000" +
    "\n";
61          assertEquals(testString, currentGrid.toString());
62      }
63
64      @Test
65      public void testSetValue(){
66          ArrayList<Point> expectedValues = new ArrayList<>();
67          GridArray currentGrid = new GridArray(4, 4);
68          assertFalse(currentGrid.getValue(new Point(0,0)));
69          currentGrid.setValue(new Point(0,0), true);
70          assertTrue(currentGrid.getValue(new Point(0,0)));
71      }
72
73
74  }
75
```

```java
package de.nordakademie.pdse.grid;

import org.junit.Test;

import java.awt.*;
import java.util.ArrayList;
import java.util.HashMap;

import static org.junit.Assert.*;

public class GridPointsTest {
    @Test
    public void testCreateGrid() {
        HashMap<Point, Boolean> testGrid = new HashMap<>();
        testGrid.put(new Point(0, 0), false);
        assertEquals(testGrid, new GridPoints(1, 1).getGrid());
    }

    @Test
    public void testChangeValue() {
        HashMap<Point, Boolean> testGrid = new HashMap<>();
        testGrid.put(new Point(0, 0), true);
        GridPoints currentGrid = new GridPoints(1, 1);
        currentGrid.changeValue(new Point(0, 0));
        assertEquals(testGrid, currentGrid.getGrid());
    }

    @Test
    public void testGetVonNeumannNeighbors() {
        ArrayList<Point> expectedValues = new ArrayList<>();
        expectedValues.add(new Point(0, 0));
        expectedValues.add(new Point(1, 1));
        expectedValues.add(new Point(2, 0));
        GridPoints currentGrid = new GridPoints(4, 4);

        assertTrue(expectedValues.containsAll(currentGrid.
getVonNeumannNeighbors(new Point(1, 0)))
                && currentGrid.getVonNeumannNeighbors(new Point(1, 0)).size
() == expectedValues.size());

    }

    @Test
    public void testGetMooreNeighbors() {
        ArrayList<Point> expectedValues = new ArrayList<>();
        expectedValues.add(new Point(0, 0));
        expectedValues.add(new Point(0, 1));
        expectedValues.add(new Point(1, 1));
        expectedValues.add(new Point(2, 1));
        expectedValues.add(new Point(2, 0));
        GridPoints currentGrid = new GridPoints(4, 4);

        assertTrue(expectedValues.containsAll(currentGrid.getMooreNeighbors(new
 Point(1, 0)))
                && currentGrid.getMooreNeighbors(new Point(1, 0)).size() ==
expectedValues.size());
    }

    @Test
    public void testSetValue(){
        ArrayList<Point> expectedValues = new ArrayList<>();
        GridPoints currentGrid = new GridPoints(4, 4);
        assertFalse(currentGrid.getValue(new Point(0,0)));
        currentGrid.setValue(new Point(0,0), true);
```

```
61          assertTrue(currentGrid.getValue(new Point(0,0)));
62      }
63
64      @Test
65      public void testToString() {
66          GridPoints currentGrid = new GridPoints(4, 4);
67          String testString;
68          testString = "0000" + "\n" + "0000" + "\n" + "0000" + "\n" + "0000" +
    "\n";
69          assertEquals(testString, currentGrid.toString());
70      }
71
72  }
73
```

```java
package de.nordakademie.pdse.grid;

import de.nordakademie.pdse.config.ConfigReader;
import org.junit.Test;
import org.mockito.internal.matchers.apachecommons.ReflectionEquals;

import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public class GridFactoryTest {

    @Test
    public void testGetGridAsGridArray() {
        ConfigReader configReader = mock(ConfigReader.class);
        when(configReader.getDatastructure()).thenReturn("GridArray");
        GridFactory gridFactory = new GridFactory(4, 4, (configReader.
getDatastructure()));
        IGrid currentGrid = new GridArray(4, 4);
        assertTrue(new ReflectionEquals(currentGrid).matches(gridFactory.
getGrid()));
    }

    @Test
    public void testGetGridAsGridPoints() {
        ConfigReader configReader = mock(ConfigReader.class);
        when(configReader.getDatastructure()).thenReturn("GridPoints");
        GridFactory gridFactory = new GridFactory(4, 4, configReader.
getDatastructure());
        IGrid currentGrid = new GridPoints(4, 4);
        assertTrue(new ReflectionEquals(currentGrid).matches(gridFactory.
getGrid()));
    }
}

```

```java
package de.nordakademie.pdse.config;

import org.apache.commons.io.FileUtils;

import java.io.File;
import java.io.IOException;
import java.util.Properties;

import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.TemporaryFolder;

import static org.junit.Assert.assertEquals;

/**
 * Unit test vor getters and setters in the config class.
 *
 * @author Fabian Forthmann
 */
public class ConfigReaderTest {

    private final static String GAME_TYPE = "Parity";
    private final static String MODEL = "Moore";
    private final static String DATA_STRUCTURE = "GridPoints";
    private final static String LOGGING_TYPE = "console";
    private final static String TIME_TO_LIVE = "500";
    private final static String GRID_LENGTH = "40";
    private final static String GRID_WIDTH = "67";
    private final static String TERMINATION_TYPE = "ttl";

    private final static String ALTERNATIVE_GAME_TYPE = "GameOfLife";
    private final static String ALTERNATIVE_MODEL = "vonNeumann";
    private final static String ALTERNATIVE_DATA_STRUCTURE = "GridArray";
    private final static String ALTERNATIVE_LOGGING_TYPE = "file";
    private final static String ALTERNATIVE_TIME_TO_LIVE = "30";
    private final static String ALTERNATIVE_GRID_LENGTH = "3";
    private final static String ALTERNATIVE_GRID_WIDTH = "98";
    private final static String ALTERNATIVE_TERMINATION_TYPE = "noChange";

    @Rule
    public TemporaryFolder tempFolder = new TemporaryFolder();

    @Test
    public void readExistingPropertiesFromFile() throws IOException {
        Properties properties = initializeValidPropertiesFile();

        final File tempFile = tempFolder.newFile("tempFile.properties");
        // Writing String into the temporary file and formatting it
        FileUtils.writeStringToFile(tempFile, properties.toString().replaceAll(
",", "\n").replaceFirst("\\{", "").replaceAll("}", "").replaceAll(" ", ""));
        ConfigReader configReader = new ConfigReader(tempFile);

        assertEquals(GAME_TYPE, configReader.getGameType());
        assertEquals(MODEL, configReader.getModel());
        assertEquals(DATA_STRUCTURE, configReader.getDatastructure());
        assertEquals(LOGGING_TYPE, configReader.getLoggingType());
        assertEquals(Integer.parseInt(TIME_TO_LIVE), configReader.getTimeToLive
());
        assertEquals(Integer.parseInt(GRID_LENGTH), configReader.getGridLength
());
        assertEquals(Integer.parseInt(GRID_WIDTH), configReader.getGridWidth
());
        assertEquals(TERMINATION_TYPE, configReader.getTerminationType());
    }
```

```
61
62      @Test
63      public void setPropertyValues() throws IOException {
64          Properties properties = initializeValidPropertiesFile();
65
66          final File tempFile = tempFolder.newFile("tempFile.properties");
67          // Writing String into the temporary file and formatting it
68          FileUtils.writeStringToFile(tempFile, properties.toString().replaceAll
    (",", "\n").replaceFirst("\\{", "").replaceAll("}", "").replaceAll(" ", ""));
69          ConfigReader configReader = new ConfigReader(tempFile);
70
71          configReader.setGameType(ALTERNATIVE_GAME_TYPE);
72          configReader.setModel(ALTERNATIVE_MODEL);
73          configReader.setDatastructure(ALTERNATIVE_DATA_STRUCTURE);
74          configReader.setLoggingType(ALTERNATIVE_LOGGING_TYPE);
75          configReader.setTimeToLive(ALTERNATIVE_TIME_TO_LIVE);
76          configReader.setGridLength(ALTERNATIVE_GRID_LENGTH);
77          configReader.setGridWidth(ALTERNATIVE_GRID_WIDTH);
78          configReader.setTerminationType(ALTERNATIVE_TERMINATION_TYPE);
79
80          assertEquals(ALTERNATIVE_GAME_TYPE, configReader.getGameType());
81          assertEquals(ALTERNATIVE_MODEL, configReader.getModel());
82          assertEquals(ALTERNATIVE_DATA_STRUCTURE, configReader.getDatastructure
    ());
83          assertEquals(ALTERNATIVE_LOGGING_TYPE, configReader.getLoggingType());
84          assertEquals(Integer.parseInt(ALTERNATIVE_TIME_TO_LIVE), configReader.
    getTimeToLive());
85          assertEquals(Integer.parseInt(ALTERNATIVE_GRID_LENGTH), configReader.
    getGridLength());
86          assertEquals(Integer.parseInt(ALTERNATIVE_GRID_WIDTH), configReader.
    getGridWidth());
87          assertEquals(ALTERNATIVE_TERMINATION_TYPE, configReader.
    getTerminationType());
88      }
89
90      @Test
91      public void changeExistingPropertyValues() throws IOException {
92          Properties properties = initializeValidPropertiesFile();
93
94          final File tempFile = tempFolder.newFile("tempFile.properties");
95          // Writing String into the temporary file and formatting it
96          FileUtils.writeStringToFile(tempFile, properties.toString().replaceAll
    (",", "\n").replaceFirst("\\{", "").replaceAll("}", "").replaceAll(" ", ""));
97          ConfigReader configReader = new ConfigReader(tempFile);
98
99          configReader.setGameType(ALTERNATIVE_GAME_TYPE);
100         configReader.setModel(ALTERNATIVE_MODEL);
101         configReader.setDatastructure(ALTERNATIVE_DATA_STRUCTURE);
102         configReader.setLoggingType(ALTERNATIVE_LOGGING_TYPE);
103         configReader.setTimeToLive(ALTERNATIVE_TIME_TO_LIVE);
104         configReader.setGridLength(ALTERNATIVE_GRID_LENGTH);
105         configReader.setGridWidth(ALTERNATIVE_GRID_WIDTH);
106         configReader.setTerminationType(ALTERNATIVE_TERMINATION_TYPE);
107
108         assertEquals(ALTERNATIVE_GAME_TYPE, configReader.getGameType());
109         assertEquals(ALTERNATIVE_MODEL, configReader.getModel());
110         assertEquals(ALTERNATIVE_DATA_STRUCTURE, configReader.getDatastructure
    ());
111         assertEquals(ALTERNATIVE_LOGGING_TYPE, configReader.getLoggingType());
112         assertEquals(Integer.parseInt(ALTERNATIVE_TIME_TO_LIVE), configReader.
    getTimeToLive());
113         assertEquals(Integer.parseInt(ALTERNATIVE_GRID_LENGTH), configReader.
    getGridLength());
114         assertEquals(Integer.parseInt(ALTERNATIVE_GRID_WIDTH), configReader.
```

```
114 getGridWidth());
115         assertEquals(ALTERNATIVE_TERMINATION_TYPE, configReader.
    getTerminationType());
116     }
117 
118     @Test(expected = RuntimeException.class)
119     public void readNotExistingConfigFile() {
120         File configPropertyFile = new File("fileThatDoesNotExist.properties");
121         ConfigReader configReader = new ConfigReader(configPropertyFile);
122         configReader.getGameType();
123     }
124 
125     private Properties initializeValidPropertiesFile() {
126         Properties properties = new Properties();
127 
128         properties.setProperty("game.type", GAME_TYPE);
129         properties.setProperty("game.model", MODEL);
130         properties.setProperty("game.dataStructure", DATA_STRUCTURE);
131         properties.setProperty("game.loggingType", LOGGING_TYPE);
132         properties.setProperty("game.timeToLive", TIME_TO_LIVE);
133         properties.setProperty("game.grid.length", GRID_LENGTH);
134         properties.setProperty("game.grid.width", GRID_WIDTH);
135         properties.setProperty("game.terminationType", TERMINATION_TYPE);
136 
137         return properties;
138     }
139 }
140 
```

```java
package de.nordakademie.pdse.config;

import org.junit.Test;

import java.util.Properties;

/**
 * Unit tests for the validation of the given property parameters.
 *
 * @author Fabian Forthmann
 */
public class ConfigValidatorTest {

    private final static String GAME_TYPE = "game.type";
    private final static String GAME_MODEL = "game.model";
    private final static String DATA_STRUCTURE = "game.dataStructure";
    private final static String LOGGING_TYPE = "game.loggingType";
    private final static String TIME_TO_LIVE = "game.timeToLive";
    private final static String LENGTH = "game.grid.length";
    private final static String WIDTH = "game.grid.width";
    private final static String TERMINATION_TYPE = "game.terminationType";

    @Test
    public void validateValidPropertyConfig() {
        Properties prop = new Properties();
        prop.setProperty(GAME_TYPE, "GameOfLife");
        prop.setProperty(GAME_MODEL, "vonNeumann");
        prop.setProperty(DATA_STRUCTURE, "GridArray");
        prop.setProperty(LOGGING_TYPE, "disable");
        prop.setProperty(LENGTH, "10");
        prop.setProperty(WIDTH, "10");
        prop.setProperty(TIME_TO_LIVE, "589");
        prop.setProperty(TERMINATION_TYPE, "ttlOrNoChange");

        PropertyValidator propertyValidator = new PropertyValidator();
        propertyValidator.validateConfig(prop);
    }

    @Test
    public void validateValidPropertyConfigWithTtl() {
        Properties prop = new Properties();
        prop.setProperty(GAME_TYPE, "GameOfLife");
        prop.setProperty(GAME_MODEL, "vonNeumann");
        prop.setProperty(DATA_STRUCTURE, "GridArray");
        prop.setProperty(LOGGING_TYPE, "disable");
        prop.setProperty(LENGTH, "10");
        prop.setProperty(WIDTH, "10");
        prop.setProperty(TERMINATION_TYPE, "ttl");
        prop.setProperty(TIME_TO_LIVE, "99");

        PropertyValidator propertyValidator = new PropertyValidator();
        propertyValidator.validateConfig(prop);
    }

    @Test(expected = RuntimeException.class)
    public void validateInvalidPropertyConfigWithFalseStrings() {
        Properties prop = new Properties();
        prop.setProperty(GAME_TYPE, "GameOfLifeA");
        prop.setProperty(GAME_MODEL, "vonNeumannB");
        prop.setProperty(DATA_STRUCTURE, "GridArrayC");
        prop.setProperty(LOGGING_TYPE, "disableD");
        prop.setProperty(LENGTH, "10");
        prop.setProperty(WIDTH, "10");
        prop.setProperty(TIME_TO_LIVE, "92");
```

```
65          prop.setProperty(TERMINATION_TYPE, "ttlOrNoChangeE");
66
67          PropertyValidator propertyValidator = new PropertyValidator();
68          propertyValidator.validateConfig(prop);
69      }
70
71      @Test(expected = RuntimeException.class)
72      public void validateInvalidPropertyConfigWithFalseIntegers() {
73          Properties prop = new Properties();
74          prop.setProperty(GAME_TYPE, "GameOfLife");
75          prop.setProperty(GAME_MODEL, "vonNeumann");
76          prop.setProperty(DATA_STRUCTURE, "GridArray");
77          prop.setProperty(LOGGING_TYPE, "disable");
78          prop.setProperty(LENGTH, "0");
79          prop.setProperty(WIDTH, "-10");
80          prop.setProperty(TIME_TO_LIVE, "-987");
81          prop.setProperty(TERMINATION_TYPE, "ttlOrNoChange");
82
83          PropertyValidator propertyValidator = new PropertyValidator();
84          propertyValidator.validateConfig(prop);
85      }
86
87      @Test(expected = RuntimeException.class)
88      public void validateInvalidPropertyConfigWithInvalidTtl() {
89          Properties prop = new Properties();
90          prop.setProperty(GAME_TYPE, "GameOfLife");
91          prop.setProperty(GAME_MODEL, "vonNeumann");
92          prop.setProperty(DATA_STRUCTURE, "GridArray");
93          prop.setProperty(LOGGING_TYPE, "disable");
94          prop.setProperty(LENGTH, "10");
95          prop.setProperty(WIDTH, "10");
96          prop.setProperty(TERMINATION_TYPE, "ttl");
97          prop.setProperty(TIME_TO_LIVE, "-54");
98
99          PropertyValidator propertyValidator = new PropertyValidator();
100         propertyValidator.validateConfig(prop);
101     }
102
103     @Test(expected = RuntimeException.class)
104     public void validateInvalidPropertyConfigWithMissingFields() {
105         Properties prop = new Properties();
106         prop.setProperty(GAME_TYPE, "GameOfLife");
107         prop.setProperty(DATA_STRUCTURE, "GridArray");
108         prop.setProperty(LOGGING_TYPE, "disable");
109         prop.setProperty(WIDTH, "10");
110         prop.setProperty(TERMINATION_TYPE, "noChange");
111
112         PropertyValidator propertyValidator = new PropertyValidator();
113         propertyValidator.validateConfig(prop);
114     }
115 }
116
```

```java
package de.nordakademie.pdse.logging;

import de.nordakademie.pdse.grid.GridFactory;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.io.*;

import static org.junit.Assert.assertArrayEquals;
import static org.junit.Assert.assertTrue;

public class LoggerTest {

    private final PrintStream standardOut = System.out;
    private final ByteArrayOutputStream outputStreamCaptor = new
ByteArrayOutputStream();

    @Before
    public void setUp() {
        System.setOut(new PrintStream(outputStreamCaptor));
    }

    @After
    public void after() {
        outputStreamCaptor.reset();
    }

    @Test
    public void testAddGridToFile() {
        String expectedOutput = "###2\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" + "\n\n";
        Logger log = new Logger("file");
        log.addGridToLog(new GridFactory(4, 4, "GridArray").getGrid().toString
(), 2);
        StringBuilder current = new StringBuilder();
        String line;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(log.
getFile()));
            while ((line = reader.readLine()) != null) {
                current.append(line).append("\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        current.append("\n");

        assertArrayEquals(expectedOutput.toCharArray(), current.toString().
toCharArray());

    }

    @Test
    public void testAddGridToLogConsole() {
        String expectedOutput = "###2\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" + "\r\n";
        System.setOut(new PrintStream(outputStreamCaptor));
```

```
61          Logger log = new Logger("console");
62          log.addGridToLog(new GridFactory(4, 4, "GridArray").getGrid().toString
    (), 2);
63          String current = outputStreamCaptor.toString();
64          assertArrayEquals(expectedOutput.toCharArray(), current.toCharArray
    ());
65
66      }
67
68      @Test
69      public void testCustomConstructor() {
70          Logger log = new Logger("file", "experiment1");
71          assertTrue(log.getFile().exists());
72      }
73
74  }
```

```java
package de.nordakademie.pdse.logging;

import de.nordakademie.pdse.grid.GridFactory;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.io.*;

import static org.junit.Assert.assertArrayEquals;

public class TestConsoleAndFileOutput {

    private final PrintStream standardOut = System.out;
    private final ByteArrayOutputStream outputStreamCaptor = new ByteArrayOutputStream();

    @After
    public void after() {
        outputStreamCaptor.reset();
    }

    @Before
    public void setUp() {
        System.setOut(new PrintStream(outputStreamCaptor));
    }

    @Test
    public void testAddGridToLogToConsoleAndToFile() {
        String expectedConsoleOutput = "###2\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" + "\r\n";
        String expectedFileOutput = "###2\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" +
                "0000\n" + "\n\n";
        System.setOut(new PrintStream(outputStreamCaptor));
        Logger log = new Logger("consoleAndFile");
        log.addGridToLog(new GridFactory(4, 4, "GridArray").getGrid().toString(), 2);
        String currentConsole = outputStreamCaptor.toString();
        StringBuilder currentFile = new StringBuilder();
        String line;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(log.getFile()));
            while ((line = reader.readLine()) != null) {
                currentFile.append(line).append("\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        currentFile.append("\n");

        assertArrayEquals(expectedFileOutput.toCharArray(), currentFile.toString().toCharArray());

        assertArrayEquals(expectedConsoleOutput.toCharArray(), currentConsole.toCharArray());
    }
}
```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.PrintStream;

import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen
 */

public class GameTest {

    private final ByteArrayOutputStream outputStreamCaptor = new
ByteArrayOutputStream();

    @After
    public void after() {
        outputStreamCaptor.reset();
    }

    @Before
    public void setUp() {
        System.setOut(new PrintStream(outputStreamCaptor));
    }

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @Mock
    IGrid grid = mock(IGrid.class);

    @Mock
    IGrid grid2 = mock(IGrid.class);

    @Test
    public void getGird() {
        when(configReader.getTerminationType()).thenReturn("noChange");
        when(configReader.getLoggingType()).thenReturn("console");
        when(configReader.getDatastructure()).thenReturn("GridArray");
        when(configReader.getGameType()).thenReturn("Parity");
        when(configReader.getTimeToLive()).thenReturn(2);
        GridFactory gridFactory = new GridFactory(configReader.getGridLength
(), configReader.getGridWidth(), configReader.getDatastructure());
        grid = gridFactory.getGrid();
        Game game = new Game(configReader);
        game.setGrid(grid);
        assertEquals(grid, game.getGird());
    }

    @Test
    public void setGrid() {
```

```
63          when(configReader.getTerminationType()).thenReturn("noChange");
64          when(configReader.getLoggingType()).thenReturn("console");
65          when(configReader.getDatastructure()).thenReturn("GridArray");
66          when(configReader.getGameType()).thenReturn("Parity");
67          when(configReader.getTimeToLive()).thenReturn(2);
68          GridFactory gridFactory = new GridFactory(configReader.getGridLength
    (), configReader.getGridWidth(), configReader.getDatastructure());
69          grid = gridFactory.getGrid();
70          Game game = new Game(configReader);
71          game.setGrid(grid);
72          assertEquals(grid, game.getGird());
73          game.setGrid(grid2);
74          assertEquals(grid2, game.getGird());
75          assertNotEquals(grid, game.getGird());
76      }
77
78      @Test
79      public void run() throws Exception {
80          System.setOut(new PrintStream(outputStreamCaptor));
81          GridFactory gridFactory = new GridFactory(3, 3, "GridArray");
82          IGrid grid = gridFactory.getGrid();
83          when(configReader.getLoggingType()).thenReturn("console");
84          when(configReader.getDatastructure()).thenReturn("GridArray");
85          when(configReader.getGameType()).thenReturn("Parity");
86          when(configReader.getModel()).thenReturn("vonNeumann");
87          when(configReader.getTerminationType()).thenReturn("ttl");
88          when(configReader.getGridLength()).thenReturn(3);
89          when(configReader.getGridWidth()).thenReturn(3);
90          when(configReader.getTimeToLive()).thenReturn(2);
91          grid.setValue(new Point(1, 1), true);
92          Game game = new Game(configReader);
93          game.setGrid(grid);
94          game.run();
95          IGrid checkGrid = game.getGird();
96          for (int width = 0; width < 3; width++) {
97              for (int length = 0; length < 3; length++) {
98                  assertFalse(checkGrid.getValue(new Point(length, width)));
99              }
100         }
101         String expectedOutput = "###0\n" +
102                 "000\n" +
103                 "010\n" +
104                 "000\n" +
105                 "\r\n" +
106                 "###1\n" +
107                 "010\n" +
108                 "101\n" +
109                 "010\n" +
110                 "\r\n" +
111                 "###2\n" +
112                 "000\n" +
113                 "000\n" +
114                 "000\n" +
115                 "\r\n";
116         assertArrayEquals(expectedOutput.toCharArray(), outputStreamCaptor.
    toString().toCharArray());
117     }
118
119     @Test
120     public void secondaryConstructor() throws Exception {
121         GridFactory gridFactory = new GridFactory(3, 3, "GridArray");
122         IGrid grid = gridFactory.getGrid();
123         String fileName = "testSecondaryConstructor";
124         assertFalse(new File(fileName + ".log").exists());
```

```
125         when(configReader.getLoggingType()).thenReturn("file");
126         when(configReader.getDatastructure()).thenReturn("GridArray");
127         when(configReader.getGameType()).thenReturn("Parity");
128         when(configReader.getModel()).thenReturn("vonNeumann");
129         when(configReader.getTerminationType()).thenReturn("ttl");
130         when(configReader.getGridLength()).thenReturn(3);
131         when(configReader.getGridWidth()).thenReturn(3);
132         when(configReader.getTimeToLive()).thenReturn(0);
133         grid.setValue(new Point(1, 1), true);
134         Game game = new Game(configReader, fileName);
135         assertTrue(new File(fileName + ".log").exists());
136         game.setGrid(grid);
137         game.run();
138         assertTrue(new File(fileName + ".log").exists());
139         new File(fileName + ".log").delete();
140         assertFalse(new File(fileName + ".log").exists());
141     }
142 }
143
```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen
 */

public class ParityTestSetAlive {

    GridFactory gridFactory = new GridFactory(3, 3, "GridArray");
    IGrid grid = gridFactory.getGrid();

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    private static Point createPoint(int x, int y) {
        return new Point(x, y);
    }

    @Test
    public void trueValueToFalseThroughStepsInGrid() throws Exception {
        grid.setValue(createPoint(1, 1), true);
        when(configReader.getGridLength()).thenReturn(3);
        when(configReader.getGridWidth()).thenReturn(3);
        when(configReader.getModel()).thenReturn("vonNeumann");
        Parity parity = new Parity(configReader);
        IGrid stepGrid = parity.step(grid);
        stepGrid.getValue(createPoint(0, 2));
        assertTrue(stepGrid.getValue(createPoint(0, 1)));
        assertFalse(stepGrid.getValue(createPoint(0, 2)));
        assertTrue(stepGrid.getValue(createPoint(1, 0)));
        assertFalse(stepGrid.getValue(createPoint(1, 1)));
        assertTrue(stepGrid.getValue(createPoint(1, 2)));
        assertFalse(stepGrid.getValue(createPoint(2, 0)));
        assertTrue(stepGrid.getValue(createPoint(2, 1)));
        assertFalse(stepGrid.getValue(createPoint(2, 2)));
    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import org.junit.Test;
import org.mockito.Mock;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen
 */

public class GameTypeFactoryTest {

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @Test
    public void returnGameOfLife() {
        when(configReader.getGameType()).thenReturn("GameOfLife");
        IGameType gameType = new GameTypeFactory(configReader).getGameType();
        assertEquals(gameType.getClass(), GameOfLife.class);
    }

    @Test
    public void returnParity() {
        when(configReader.getGameType()).thenReturn("Parity");
        IGameType gameType = new GameTypeFactory(configReader).getGameType();
        assertEquals(gameType.getClass(), Parity.class);

    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;

import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public class GameTestLoggerAndRun {

    private final ByteArrayOutputStream outputStreamCaptor = new
ByteArrayOutputStream();

    IGrid grid;

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @After
    public void after() {
        outputStreamCaptor.reset();
    }

    @Before
    public void setUp() {
        System.setOut(new PrintStream(outputStreamCaptor));
    }

    @Test
    public void noChange() throws Exception {
        System.setOut(new PrintStream(outputStreamCaptor));
        when(configReader.getTerminationType()).thenReturn("noChange");
        when(configReader.getGridLength()).thenReturn(3);
        when(configReader.getGridWidth()).thenReturn(3);
        when(configReader.getLoggingType()).thenReturn("console");
        when(configReader.getDatastructure()).thenReturn("GridArray");
        when(configReader.getGameType()).thenReturn("Parity");
        when(configReader.getModel()).thenReturn("vonNeumann");
        when(configReader.getTimeToLive()).thenReturn(10);
        GridFactory gridFactory = new GridFactory(configReader.getGridLength
(), configReader.getGridWidth(), configReader.getDatastructure());
        grid = gridFactory.getGrid();
        grid.setValue(new Point(1, 1), true);
        Game game = new Game(configReader);
        game.setGrid(grid);
        game.run();
        String expectedOutput = "###0\n" +
                "000\n" +
                "010\n" +
                "000\n" +
                "\r\n" +
                "###1\n" +
                "010\n" +
                "101\n" +
```

```
63                  "010\n" +
64                  "\r\n" +
65                  "###2\n" +
66                  "000\n" +
67                  "000\n" +
68                  "000\n" +
69                  "\r\n";
70          assertArrayEquals(expectedOutput.toCharArray(), outputStreamCaptor.
    toString().toCharArray());
71      }
72
73      @Test
74      public void ttlZero() throws Exception {
75          System.setOut(new PrintStream(outputStreamCaptor));
76          when(configReader.getTerminationType()).thenReturn("ttl");
77          when(configReader.getGridLength()).thenReturn(3);
78          when(configReader.getGridWidth()).thenReturn(3);
79          when(configReader.getLoggingType()).thenReturn("console");
80          when(configReader.getDatastructure()).thenReturn("GridArray");
81          when(configReader.getGameType()).thenReturn("Parity");
82          when(configReader.getModel()).thenReturn("vonNeumann");
83          when(configReader.getTimeToLive()).thenReturn(0);
84          GridFactory gridFactory = new GridFactory(configReader.getGridLength
    (), configReader.getGridWidth(), configReader.getDatastructure());
85          grid = gridFactory.getGrid();
86          grid.setValue(new Point(1, 1), true);
87          Game game = new Game(configReader);
88          game.setGrid(grid);
89          game.run();
90          assertTrue(game.getGird().getValue(new Point(1, 1)));
91          String expectedOutput = "###0\n" +
92                  "000\n" +
93                  "010\n" +
94                  "000\n" +
95                  "\r\n";
96          assertArrayEquals(expectedOutput.toCharArray(), outputStreamCaptor.
    toString().toCharArray());
97      }
98
99      @Test
100     public void ttlOrNoChangeBreakttl() throws Exception {
101         when(configReader.getTerminationType()).thenReturn("ttlOrNoChange");
102         when(configReader.getGridLength()).thenReturn(3);
103         when(configReader.getGridWidth()).thenReturn(3);
104         when(configReader.getLoggingType()).thenReturn("console");
105         when(configReader.getDatastructure()).thenReturn("GridArray");
106         when(configReader.getGameType()).thenReturn("Parity");
107         when(configReader.getModel()).thenReturn("vonNeumann");
108         when(configReader.getTimeToLive()).thenReturn(1);
109         GridFactory gridFactory = new GridFactory(configReader.getGridLength
    (), configReader.getGridWidth(), configReader.getDatastructure());
110         grid = gridFactory.getGrid();
111         grid.setValue(new Point(1, 1), true);
112         Game game = new Game(configReader);
113         game.setGrid(grid);
114         game.run();
115     }
116
117     @Test
118     public void ttlOrNoChangeBreakNoChange() throws Exception {
119         System.setOut(new PrintStream(outputStreamCaptor));
120         when(configReader.getTerminationType()).thenReturn("ttlOrNoChange");
121         when(configReader.getGridLength()).thenReturn(3);
122         when(configReader.getGridWidth()).thenReturn(3);
```

```
123         when(configReader.getLoggingType()).thenReturn("console");
124         when(configReader.getDatastructure()).thenReturn("GridArray");
125         when(configReader.getGameType()).thenReturn("Parity");
126         when(configReader.getModel()).thenReturn("vonNeumann");
127         when(configReader.getTimeToLive()).thenReturn(10);
128         GridFactory gridFactory = new GridFactory(configReader.getGridLength
    (), configReader.getGridWidth(), configReader.getDatastructure());
129         grid = gridFactory.getGrid();
130         grid.setValue(new Point(1, 1), true);
131         Game game = new Game(configReader);
132         game.setGrid(grid);
133         game.run();
134         String expectedOutput = "###0\n" +
135                 "000\n" +
136                 "010\n" +
137                 "000\n" +
138                 "\r\n" +
139                 "###1\n" +
140                 "010\n" +
141                 "101\n" +
142                 "010\n" +
143                 "\r\n" +
144                 "###2\n" +
145                 "000\n" +
146                 "000\n" +
147                 "000\n" +
148                 "\r\n";
149         assertArrayEquals(expectedOutput.toCharArray(), outputStreamCaptor.
    toString().toCharArray());
150     }
151 }
152
```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen
 */

public class GameOfLifeTestSetAlive {

    GridFactory gridFactory = new GridFactory(3, 3, "GridArray");
    IGrid grid = gridFactory.getGrid();

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @Test
    public void trueValueToFalseThroughStep(){
        grid.setValue(new Point(1, 1), true);
        when(configReader.getGridLength()).thenReturn(3);
        when(configReader.getGridWidth()).thenReturn(3);
        when(configReader.getModel()).thenReturn("Moore");

        GameOfLife gameOfLife = new GameOfLife(configReader);
        IGrid stepGrid = gameOfLife.step(grid);
        assertFalse(stepGrid.getValue(new Point(1,1)));
    }

    @Test
    public void falseValueToTrueThroughStep() {
        grid.setValue(new Point(0, 0), true);
        grid.setValue(new Point(0, 1), true);
        grid.setValue(new Point(0, 2), true);
        when(configReader.getGridLength()).thenReturn(3);
        when(configReader.getGridWidth()).thenReturn(3);
        when(configReader.getModel()).thenReturn("Moore");

        GameOfLife gameOfLife = new GameOfLife(configReader);
        IGrid stepGrid = gameOfLife.step(grid);
        assertTrue(stepGrid.getValue(new Point(0,1)));
        assertTrue(stepGrid.getValue(new Point(1,1)));
        assertFalse(stepGrid.getValue(new Point(0,0)));
        assertFalse(stepGrid.getValue(new Point(0,2)));
        assertFalse(stepGrid.getValue(new Point(1,0)));
        assertFalse(stepGrid.getValue(new Point(1,2)));
    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen, Fabian Forthmann
 */

public class ParityTestOneByOneGrid {

    GridFactory gridFactory = new GridFactory(1, 1, "GridArray");
    IGrid grid = gridFactory.getGrid();
    Point point = new Point(0, 0);

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @Test
    public void emptyParityGridVonNeumann() {
        grid.setValue(point, false);
        when(configReader.getGridLength()).thenReturn(1);
        when(configReader.getGridWidth()).thenReturn(1);
        when(configReader.getModel()).thenReturn("vonNeumann");
        Parity parity = new Parity(configReader);
        assertFalse(parity.step(grid).getValue(point));
    }

    @Test
    public void trueParityGridVonNeumann() {
        grid.setValue(point, true);
        when(configReader.getGridLength()).thenReturn(1);
        when(configReader.getGridWidth()).thenReturn(1);
        when(configReader.getModel()).thenReturn("vonNeumann");
        Parity parity = new Parity(configReader);
        assertFalse(parity.step(grid).getValue(point));
    }
}

```

```java
package de.nordakademie.pdse.gamelogic;

import de.nordakademie.pdse.config.ConfigReader;
import de.nordakademie.pdse.grid.GridFactory;
import de.nordakademie.pdse.grid.IGrid;
import org.junit.Test;
import org.mockito.Mock;

import java.awt.*;

import static org.junit.Assert.assertFalse;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * @author Rane Petersen, Fabian Forthmann
 */

public class GameOfLifeTestOneByOneGrid {

    GridFactory gridFactory = new GridFactory(1, 1, "GridArray");
    IGrid grid = gridFactory.getGrid();
    Point point = new Point(0, 0);

    @Mock
    ConfigReader configReader = mock(ConfigReader.class);

    @Test
    public void oneByOneGridWithFalseValueMoore(){
        grid.setValue(point, false);
        when(configReader.getGridLength()).thenReturn(1);
        when(configReader.getGridWidth()).thenReturn(1);
        when(configReader.getModel()).thenReturn("Moore");

        GameOfLife gameOfLife = new GameOfLife(configReader);
        assertFalse(gameOfLife.step(grid).getValue(point));
    }

    @Test
    public void oneByOneGridWithTrueValueMoore(){
        grid.setValue(point, true);
        when(configReader.getGridLength()).thenReturn(1);
        when(configReader.getGridWidth()).thenReturn(1);
        when(configReader.getModel()).thenReturn("Moore");

        GameOfLife gameOfLife = new GameOfLife(configReader);
        assertFalse(gameOfLife.step(grid).getValue(point));
    }

    @Test
    public void oneByOneGridWithFalseValueVonNeumann() {
        grid.setValue(point, false);
        when(configReader.getGridLength()).thenReturn(1);
        when(configReader.getGridWidth()).thenReturn(1);
        when(configReader.getModel()).thenReturn("vonNeumann");

        GameOfLife gameOfLife = new GameOfLife(configReader);
        assertFalse(gameOfLife.step(grid).getValue(point));
    }

    @Test
    public void oneByOneGridWithTrueValueVonNeumann() {
        grid.setValue(point, true);
        when(configReader.getGridLength()).thenReturn(1);
```

```
65          when(configReader.getGridWidth()).thenReturn(1);
66          when(configReader.getModel()).thenReturn("vonNeumann");
67
68          GameOfLife gameOfLife = new GameOfLife(configReader);
69          assertFalse(gameOfLife.step(grid).getValue(point));
70      }
71  }
72
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

public class Main {
    private de.nordakademie.pdse.config.ConfigReader configReader;

    public Main() { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception
   { /* compiled code */ }

    private void setPoints(de.nordakademie.pdse.gamelogic.Game game) { /*
  compiled code */ }
}
```

```
game.type=Parity
#game.type= "Parity", "GameOfLife"#
game.model=vonNeumann
#game.model= "Moore", "vonNeumann"#
game.dataStructure=GridPoints
#game.dataStructure= "GridPoints", "GridArray"#
game.loggingType=console
#game.loggingType= "console","file","consoleAndFile","disable"#
game.timeToLive=10
game.grid.length=10
game.grid.width=10
game.terminationType=ttl
#game.terminationType= "ttl", "noChange", "ttlOrNoChange"

```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public abstract class Grid {
    public Grid() { /* compiled code */ }

    abstract boolean checkFieldExists(int i, int i1);

    public java.util.ArrayList<java.awt.Point> getMooreNeighbors(java.awt.Point position) { /* compiled code */ }

    public java.util.ArrayList<java.awt.Point> getVonNeumannNeighbors(java.awt.Point position) { /* compiled code */ }

    public int countMooreActiveNeighbors(java.awt.Point position) { /* compiled code */ }

    public int countVonNeumannActiveNeighbors(java.awt.Point position) { /* compiled code */ }

    abstract boolean getValue(java.awt.Point point);
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public interface IGrid extends java.io.Serializable {
    void changeValue(java.awt.Point point);

    java.lang.Object getGrid();

    java.util.ArrayList<java.awt.Point> getMooreNeighbors(java.awt.Point point
);

    java.util.ArrayList<java.awt.Point> getVonNeumannNeighbors(java.awt.Point
point);

    boolean getValue(java.awt.Point point);

    java.lang.String toString();

    void setValue(java.awt.Point point, boolean b);

    int countMooreActiveNeighbors(java.awt.Point point);

    int countVonNeumannActiveNeighbors(java.awt.Point point);
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class CopyGrid {
    private final de.nordakademie.pdse.grid.IGrid grid;

    public CopyGrid(de.nordakademie.pdse.grid.IGrid grid) { /* compiled code */ }

    private java.lang.Object deepCopy(java.lang.Object object) { /* compiled code */ }

    public de.nordakademie.pdse.grid.IGrid getGrid() { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class GridArray extends de.nordakademie.pdse.grid.Grid implements de.nordakademie.pdse.grid.IGrid {
    private final int length;
    private final int width;
    private int[][] grid;

    public GridArray(int length, int width) { /* compiled code */ }

    public void changeValue(java.awt.Point position) { /* compiled code */ }

    public void setValue(java.awt.Point position, boolean value) { /* compiled code */ }

    boolean checkFieldExists(int x, int y) { /* compiled code */ }

    public boolean getValue(java.awt.Point position) { /* compiled code */ }

    private void fillGrid() { /* compiled code */ }

    public int[][] getGrid() { /* compiled code */ }

    private void createGrid() { /* compiled code */ }

    public java.lang.String toString() { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class GridPoints extends de.nordakademie.pdse.grid.Grid implements de.nordakademie.pdse.grid.IGrid {
    private final int length;
    private final int width;
    private java.util.HashMap<java.awt.Point,java.lang.Boolean> grid;

    public GridPoints(int length, int width) { /* compiled code */ }

    public void changeValue(java.awt.Point position) { /* compiled code */ }

    public java.util.HashMap<java.awt.Point,java.lang.Boolean> getGrid() { /* compiled code */ }

    public void setValue(java.awt.Point position, boolean value) { /* compiled code */ }

    boolean checkFieldExists(int x, int y) { /* compiled code */ }

    private void createGrid() { /* compiled code */ }

    public java.lang.String toString() { /* compiled code */ }

    public boolean getValue(java.awt.Point position) { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class GridFactory {
    private final int width;
    private final int length;
    private final java.lang.String dataStructure;

    public GridFactory(int length, int width, java.lang.String dataStructure
) { /* compiled code */ }

    public de.nordakademie.pdse.grid.IGrid getGrid() { /* compiled code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.config;

public class ConfigReader {
    private final java.util.Properties prop;

    public ConfigReader(java.io.File propertyFile) { /* compiled code */ }

    public java.lang.String getGameType() { /* compiled code */ }

    public java.lang.String getModel() { /* compiled code */ }

    public java.lang.String getDatastructure() { /* compiled code */ }

    public java.lang.String getLoggingType() { /* compiled code */ }

    public int getTimeToLive() { /* compiled code */ }

    public int getGridLength() { /* compiled code */ }

    public int getGridWidth() { /* compiled code */ }

    public java.lang.String getTerminationType() { /* compiled code */ }

    public void setGameType(java.lang.String gameType) { /* compiled code */ }

    public void setModel(java.lang.String gameModel) { /* compiled code */ }

    public void setDatastructure(java.lang.String datastructure) { /* compiled code */ }

    public void setLoggingType(java.lang.String loggingType) { /* compiled code */ }

    public void setTimeToLive(java.lang.String timeToLive) { /* compiled code */ }

    public void setGridLength(java.lang.String gridLength) { /* compiled code */ }

    public void setGridWidth(java.lang.String gridWidth) { /* compiled code */ }

    public void setTerminationType(java.lang.String terminationType) { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.config;

public class PropertyValidator {
    private static final java.lang.String GAME_TYPE_KEY = "game.type";
    private static final java.lang.String GAME_MODEL_KEY = "game.model";
    private static final java.lang.String DATA_STRUCTURE_KEY = "game.dataStructure";
    private static final java.lang.String LOGGING_TYPE_KEY = "game.loggingType";
    private static final java.lang.String TIME_TO_LIVE_KEY = "game.timeToLive";
    private static final java.lang.String LENGTH_KEY = "game.grid.length";
    private static final java.lang.String WIDTH_KEY = "game.grid.width";
    private static final java.lang.String TERMINATION_TYPE_KEY = "game.terminationType";

    public PropertyValidator() { /* compiled code */ }

    public void validateConfig(java.util.Properties prop) { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.logging;

public class Logger {
    private final java.lang.String loggingType;
    private java.io.File file;
    private java.lang.String fileName;

    public Logger(java.lang.String loggingType, java.lang.String fileName) { /* compiled code */ }

    public Logger(java.lang.String loggingType) { /* compiled code */ }

    public void addGridToLog(java.lang.String grid, int iteration) { /* compiled code */ }

    private void createFile() { /* compiled code */ }

    private java.lang.String getCurrentTime() { /* compiled code */ }

    public java.io.File getFile() { /* compiled code */ }

    private void writeLogToConsole(java.lang.String input) { /* compiled code */ }

    private void writeLogToFile(java.lang.String input) { /* compiled code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.gamelogic;

public class Game {
    de.nordakademie.pdse.grid.IGrid grid;
    de.nordakademie.pdse.gamelogic.IGameType gameType;
    de.nordakademie.pdse.config.ConfigReader configReader;
    java.lang.Boolean continueGame;
    de.nordakademie.pdse.logging.Logger logger;
    int timeToLive;
    int iteration;

    public Game(de.nordakademie.pdse.config.ConfigReader configReader) { /* compiled code */ }

    public Game(de.nordakademie.pdse.config.ConfigReader configReader, java.lang.String logFileName) { /* compiled code */ }

    private java.lang.Boolean getContinueGame() { /* compiled code */ }

    private void setContinueGame(java.lang.Boolean continueGame) { /* compiled code */ }

    private void reduceTimeToLive() { /* compiled code */ }

    private int getTimeToLive() { /* compiled code */ }

    private boolean isTimeToLiveZero() { /* compiled code */ }

    private boolean processTimeToLive() { /* compiled code */ }

    private boolean gridChanged(de.nordakademie.pdse.grid.IGrid grid) { /* compiled code */ }

    private void checkForTermination(de.nordakademie.pdse.grid.IGrid newGrid) { /* compiled code */ }

    public de.nordakademie.pdse.grid.IGrid getGird() { /* compiled code */ }

    public void setGrid(de.nordakademie.pdse.grid.IGrid grid) { /* compiled code */ }

    private int getIteration() { /* compiled code */ }

    private void setIteration(int iteration) { /* compiled code */ }

    private int getCurrentIterationAndIterate() { /* compiled code */ }

    public void run() throws java.lang.Exception { /* compiled code */ }
}
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class Parity implements de.nordakademie.pdse.gamelogic.IGameType {
8     de.nordakademie.pdse.config.ConfigReader configReader;
9
10     public Parity(de.nordakademie.pdse.config.ConfigReader configReader) { /*
   compiled code */ }
11
12     public de.nordakademie.pdse.grid.IGrid step(de.nordakademie.pdse.grid.IGrid
    oldGrid) { /* compiled code */ }
13 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public interface IGameType {
8     de.nordakademie.pdse.grid.IGrid step(de.nordakademie.pdse.grid.IGrid iGrid
  ) throws java.lang.Exception;
9 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class GameOfLife implements de.nordakademie.pdse.gamelogic.IGameType {
8     de.nordakademie.pdse.config.ConfigReader configReader;
9
10     public GameOfLife(de.nordakademie.pdse.config.ConfigReader configReader) {
   /* compiled code */ }
11
12     public de.nordakademie.pdse.grid.IGrid step(de.nordakademie.pdse.grid.IGrid
    oldGrid) { /* compiled code */ }
13
14     private static boolean getPointStatus(java.lang.Boolean point, int
   aliveNeighbors) { /* compiled code */ }
15 }
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.gamelogic;

public class GameTypeFactory {
    de.nordakademie.pdse.config.ConfigReader configReader;

    public GameTypeFactory(de.nordakademie.pdse.config.ConfigReader configReader) { /* compiled code */ }

    public de.nordakademie.pdse.gamelogic.IGameType getGameType() { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.parity;

public class Parity_Experiment_1_GridArray {
    public Parity_Experiment_1_GridArray() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.parity;

public class Parity_Experiment_1_GridPoint {
    public Parity_Experiment_1_GridPoint() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.parity;

public class Parity_Experiment_2_GridArray {
    public Parity_Experiment_2_GridArray() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.parity;

public class Parity_Experiment_2_GridPoint {
    public Parity_Experiment_2_GridPoint() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_1_GridArray {
    public GameOfLife_Experiment_1_GridArray() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_1_GridPoint {
    public GameOfLife_Experiment_1_GridPoint() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_2_GridArray {
    public GameOfLife_Experiment_2_GridArray() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_3_GridArray {
    public GameOfLife_Experiment_3_GridArray() { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception
   { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_3_GridPoint {
    public GameOfLife_Experiment_3_GridPoint() { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java
// IntelliJ API Decompiler stub source generated from a class file
// Implementation of methods is not available

package de.nordakademie.pdse.experiments.gameoflife;

public class GameOfLife_Experiment_2_GridPoints {
    public GameOfLife_Experiment_2_GridPoints() { /* compiled code */ }

    private static java.awt.Point createPoint(int x, int y) { /* compiled code */ }

    public static void main(java.lang.String[] args) throws java.lang.Exception { /* compiled code */ }
}
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class GridTest {
    public GridTest() { /* compiled code */ }

    @org.junit.Test
    public void countMooreActiveNeighbors() { /* compiled code */ }

    @org.junit.Test
    public void countVonNeumannActiveNeighbors() { /* compiled code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class CopyGridTest {
    public CopyGridTest() { /* compiled code */ }

    @org.junit.Test
    public void testGetGrid() { /* compiled code */ }
}
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.grid;
6
7 public class GridArrayTest {
8     public GridArrayTest() { /* compiled code */ }
9
10    @org.junit.Test
11    public void testCreateGrid() { /* compiled code */ }
12
13    @org.junit.Test
14    public void testChangeValue() { /* compiled code */ }
15
16    @org.junit.Test
17    public void testGetVonNeumannNeighbors() { /* compiled code */ }
18
19    @org.junit.Test
20    public void testGetMooreNeighbors() { /* compiled code */ }
21
22    @org.junit.Test
23    public void testToString() { /* compiled code */ }
24
25    @org.junit.Test
26    public void testSetValue() { /* compiled code */ }
27 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.grid;
6
7 public class GridPointsTest {
8     public GridPointsTest() { /* compiled code */ }
9
10    @org.junit.Test
11    public void testCreateGrid() { /* compiled code */ }
12
13    @org.junit.Test
14    public void testChangeValue() { /* compiled code */ }
15
16    @org.junit.Test
17    public void testGetVonNeumannNeighbors() { /* compiled code */ }
18
19    @org.junit.Test
20    public void testGetMooreNeighbors() { /* compiled code */ }
21
22    @org.junit.Test
23    public void testSetValue() { /* compiled code */ }
24
25    @org.junit.Test
26    public void testToString() { /* compiled code */ }
27 }
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.grid;

public class GridFactoryTest {
    public GridFactoryTest() { /* compiled code */ }

    @org.junit.Test
    public void testGetGridAsGridArray() { /* compiled code */ }

    @org.junit.Test
    public void testGetGridAsGridPoints() { /* compiled code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.config;

public class ConfigReaderTest {
    private static final java.lang.String GAME_TYPE = "Parity";
    private static final java.lang.String MODEL = "Moore";
    private static final java.lang.String DATA_STRUCTURE = "GridPoints";
    private static final java.lang.String LOGGING_TYPE = "console";
    private static final java.lang.String TIME_TO_LIVE = "500";
    private static final java.lang.String GRID_LENGTH = "40";
    private static final java.lang.String GRID_WIDTH = "67";
    private static final java.lang.String TERMINATION_TYPE = "ttl";
    private static final java.lang.String ALTERNATIVE_GAME_TYPE = "GameOfLife";
    private static final java.lang.String ALTERNATIVE_MODEL = "vonNeumann";
    private static final java.lang.String ALTERNATIVE_DATA_STRUCTURE = "
GridArray";
    private static final java.lang.String ALTERNATIVE_LOGGING_TYPE = "file";
    private static final java.lang.String ALTERNATIVE_TIME_TO_LIVE = "30";
    private static final java.lang.String ALTERNATIVE_GRID_LENGTH = "3";
    private static final java.lang.String ALTERNATIVE_GRID_WIDTH = "98";
    private static final java.lang.String ALTERNATIVE_TERMINATION_TYPE = "
noChange";
    @org.junit.Rule
    public org.junit.rules.TemporaryFolder tempFolder;

    public ConfigReaderTest() { /* compiled code */ }

    @org.junit.Test
    public void readExistingPropertiesFromFile() throws java.io.IOException {
/* compiled code */ }

    @org.junit.Test
    public void setPropertyValues() throws java.io.IOException { /* compiled
code */ }

    @org.junit.Test
    public void changeExistingPropertyValues() throws java.io.IOException { /*
compiled code */ }

    @org.junit.Test(expected = java.lang.RuntimeException.class)
    public void readNotExistingConfigFile() { /* compiled code */ }

    private java.util.Properties initializeValidPropertiesFile() { /* compiled
code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.config;

public class ConfigValidatorTest {
    private static final java.lang.String GAME_TYPE = "game.type";
    private static final java.lang.String GAME_MODEL = "game.model";
    private static final java.lang.String DATA_STRUCTURE = "game.dataStructure"
;
    private static final java.lang.String LOGGING_TYPE = "game.loggingType";
    private static final java.lang.String TIME_TO_LIVE = "game.timeToLive";
    private static final java.lang.String LENGTH = "game.grid.length";
    private static final java.lang.String WIDTH = "game.grid.width";
    private static final java.lang.String TERMINATION_TYPE = "game.
terminationType";

    public ConfigValidatorTest() { /* compiled code */ }

    @org.junit.Test
    public void validateValidPropertyConfig() { /* compiled code */ }

    @org.junit.Test
    public void validateValidPropertyConfigWithTtl() { /* compiled code */ }

    @org.junit.Test(expected = java.lang.RuntimeException.class)
    public void validateInvalidPropertyConfigWithFalseStrings() { /* compiled
code */ }

    @org.junit.Test(expected = java.lang.RuntimeException.class)
    public void validateInvalidPropertyConfigWithFalseIntegers() { /* compiled
code */ }

    @org.junit.Test(expected = java.lang.RuntimeException.class)
    public void validateInvalidPropertyConfigWithInvalidTtl() { /* compiled
code */ }

    @org.junit.Test(expected = java.lang.RuntimeException.class)
    public void validateInvalidPropertyConfigWithMissingFields() { /* compiled
code */ }
}
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.logging;
6
7 public class LoggerTest {
8     private final java.io.PrintStream standardOut;
9     private final java.io.ByteArrayOutputStream outputStreamCaptor;
10
11    public LoggerTest() { /* compiled code */ }
12
13    @org.junit.Before
14    public void setUp() { /* compiled code */ }
15
16    @org.junit.After
17    public void after() { /* compiled code */ }
18
19    @org.junit.Test
20    public void testAddGridToFile() { /* compiled code */ }
21
22    @org.junit.Test
23    public void testAddGridToLogConsole() { /* compiled code */ }
24
25    @org.junit.Test
26    public void testCustomConstructor() { /* compiled code */ }
27 }
```

```java

  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.logging;

public class TestConsoleAndFileOutput {
    private final java.io.PrintStream standardOut;
    private final java.io.ByteArrayOutputStream outputStreamCaptor;

    public TestConsoleAndFileOutput() { /* compiled code */ }

    @org.junit.After
    public void after() { /* compiled code */ }

    @org.junit.Before
    public void setUp() { /* compiled code */ }

    @org.junit.Test
    public void testAddGridToLogToConsoleAndToFile() { /* compiled code */ }
}
```

```java
  // IntelliJ API Decompiler stub source generated from a class file
  // Implementation of methods is not available

package de.nordakademie.pdse.gamelogic;

public class GameTest {
    private final java.io.ByteArrayOutputStream outputStreamCaptor;
    @org.mockito.Mock
    de.nordakademie.pdse.config.ConfigReader configReader;
    @org.mockito.Mock
    de.nordakademie.pdse.grid.IGrid grid;
    @org.mockito.Mock
    de.nordakademie.pdse.grid.IGrid grid2;

    public GameTest() { /* compiled code */ }

    @org.junit.After
    public void after() { /* compiled code */ }

    @org.junit.Before
    public void setUp() { /* compiled code */ }

    @org.junit.Test
    public void getGird() { /* compiled code */ }

    @org.junit.Test
    public void setGrid() { /* compiled code */ }

    @org.junit.Test
    public void run() throws java.lang.Exception { /* compiled code */ }

    @org.junit.Test
    public void secondaryConstructor() throws java.lang.Exception { /* compiled code */ }
}
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class ParityTestSetAlive {
8     de.nordakademie.pdse.grid.GridFactory gridFactory;
9     de.nordakademie.pdse.grid.IGrid grid;
10    @org.mockito.Mock
11    de.nordakademie.pdse.config.ConfigReader configReader;
12
13    public ParityTestSetAlive() { /* compiled code */ }
14
15    private static java.awt.Point createPoint(int x, int y) { /* compiled code
    */ }
16
17    @org.junit.Test
18    public void trueValueToFalseThroughStepsInGrid() throws java.lang.Exception
    { /* compiled code */ }
19 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class GameTypeFactoryTest {
8     @org.mockito.Mock
9     de.nordakademie.pdse.config.ConfigReader configReader;
10
11    public GameTypeFactoryTest() { /* compiled code */ }
12
13    @org.junit.Test
14    public void returnGameOfLife() { /* compiled code */ }
15
16    @org.junit.Test
17    public void returnParity() { /* compiled code */ }
18 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class GameTestLoggerAndRun {
8     private final java.io.ByteArrayOutputStream outputStreamCaptor;
9     de.nordakademie.pdse.grid.IGrid grid;
10    @org.mockito.Mock
11    de.nordakademie.pdse.config.ConfigReader configReader;
12
13    public GameTestLoggerAndRun() { /* compiled code */ }
14
15    @org.junit.After
16    public void after() { /* compiled code */ }
17
18    @org.junit.Before
19    public void setUp() { /* compiled code */ }
20
21    @org.junit.Test
22    public void noChange() throws java.lang.Exception { /* compiled code */ }
23
24    @org.junit.Test
25    public void ttlZero() throws java.lang.Exception { /* compiled code */ }
26
27    @org.junit.Test
28    public void ttlOrNoChangeBreakttl() throws java.lang.Exception { /*
  compiled code */ }
29
30    @org.junit.Test
31    public void ttlOrNoChangeBreakNoChange() throws java.lang.Exception { /*
  compiled code */ }
32 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class GameOfLifeTestSetAlive {
8     de.nordakademie.pdse.grid.GridFactory gridFactory;
9     de.nordakademie.pdse.grid.IGrid grid;
10    @org.mockito.Mock
11    de.nordakademie.pdse.config.ConfigReader configReader;
12
13    public GameOfLifeTestSetAlive() { /* compiled code */ }
14
15    @org.junit.Test
16    public void trueValueToFalseThroughStep() { /* compiled code */ }
17
18    @org.junit.Test
19    public void falseValueToTrueThroughStep() { /* compiled code */ }
20 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class ParityTestOneByOneGrid {
8     de.nordakademie.pdse.grid.GridFactory gridFactory;
9     de.nordakademie.pdse.grid.IGrid grid;
10    java.awt.Point point;
11    @org.mockito.Mock
12    de.nordakademie.pdse.config.ConfigReader configReader;
13
14    public ParityTestOneByOneGrid() { /* compiled code */ }
15
16    @org.junit.Test
17    public void emptyParityGridVonNeumann() { /* compiled code */ }
18
19    @org.junit.Test
20    public void trueParityGridVonNeumann() { /* compiled code */ }
21 }
```

```
1
2   // IntelliJ API Decompiler stub source generated from a class file
3   // Implementation of methods is not available
4
5 package de.nordakademie.pdse.gamelogic;
6
7 public class GameOfLifeTestOneByOneGrid {
8     de.nordakademie.pdse.grid.GridFactory gridFactory;
9     de.nordakademie.pdse.grid.IGrid grid;
10    java.awt.Point point;
11    @org.mockito.Mock
12    de.nordakademie.pdse.config.ConfigReader configReader;
13
14    public GameOfLifeTestOneByOneGrid() { /* compiled code */ }
15
16    @org.junit.Test
17    public void oneByOneGridWithFalseValueMoore() { /* compiled code */ }
18
19    @org.junit.Test
20    public void oneByOneGridWithTrueValueMoore() { /* compiled code */ }
21
22    @org.junit.Test
23    public void oneByOneGridWithFalseValueVonNeumann() { /* compiled code */ }
24
25    @org.junit.Test
26    public void oneByOneGridWithTrueValueVonNeumann() { /* compiled code */ }
27 }
```