#### Test Plan

Project : Oil’s Well

Bedrijf : MBO Utrecht



|  |  |  |  |
| --- | --- | --- | --- |
| **Plaats, datum:**  **Versie:** | Utrecht, 15-10-2015  0.2 | | |
| **Opgesteld door:** | Projectgroep Oil’s Well | | |
|  | **[Studentnaam]** | **[Studentnummer]** | **[Mailadres]** |
|  | Gerben Tesselaar | 305931 | Gemme123@hotmail.com |

#### Inhoudsopgave Test Plan

[1. Testomgeving 2](#_Toc434232403)

[2. Eigen klassen 3](#_Toc434232404)

[3. Controller-klassen 4](#_Toc434232405)

[4. Datasheet 5](#_Toc434232406)

[5. Codestandaard 6](#_Toc434232407)

[1 Introduction 6](#_Toc434232408)

[2 Source file basics 6](#_Toc434232409)

[6. Test Rollen & Data 8](#_Toc434232410)

**Versie beheer:**

|  |  |
| --- | --- |
| **Versie** | **Wijzigingen** |
| **V0.1** | **Toegevoegd: Testomgeving, Eigen klassen, Controller-klassen, Datasheet, Codestandaard** |
| **V0.2** | **Toegevoegd: Test Rollen & Data** |
|  |  |
|  |  |

# 1. Testomgeving

De tester krijgt de Testformulieren, van de onderwerpen die getest gaan worden, van de developer. De tests worden gedaan op een ASUS R704A 6GB RAM Intel i5 2.6 GHz laptop. Andere informatie over de testomgeving staat genoteerd in het Technisch Ontwerp.

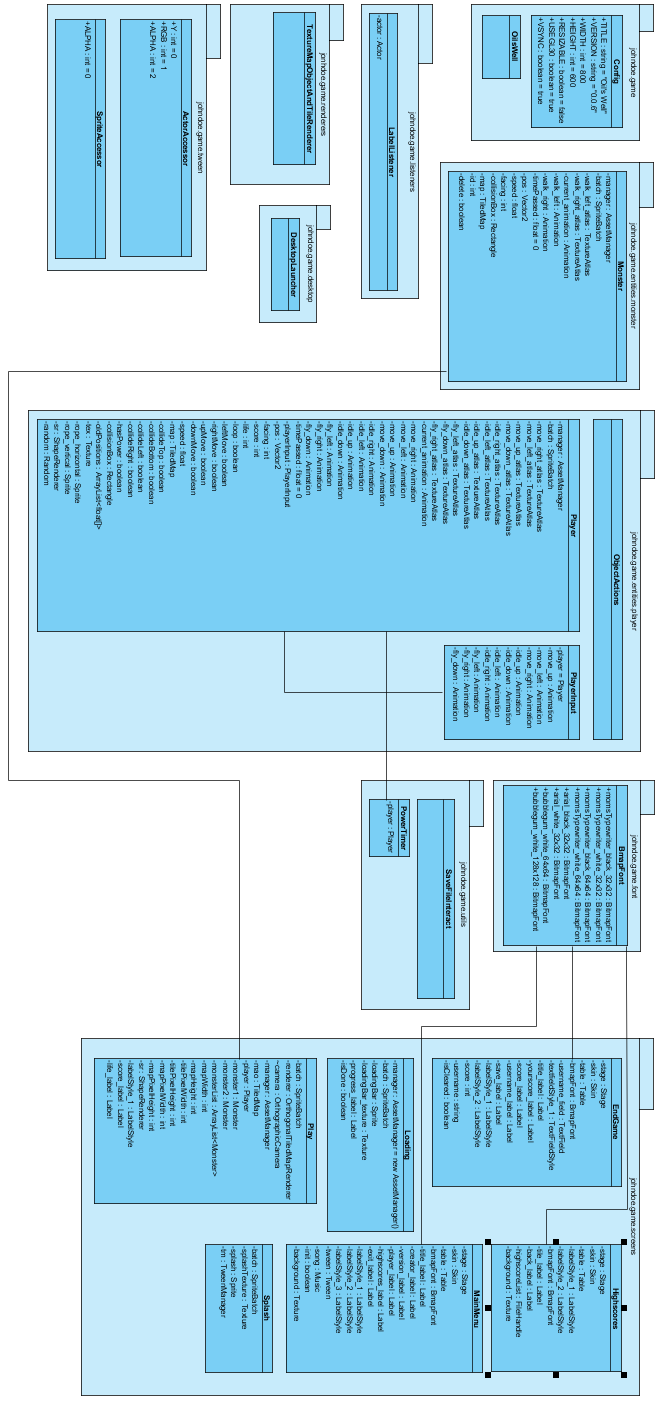
De formulieren zien er als volgt uit:

|  |  |
| --- | --- |
| Naam tester:  Datum:  Tijd: | Handtekening tester: |

|  |  |
| --- | --- |
| Onderwerp | Aantekeningen |
|  |  |
| Game of Website |  |
| Scherm 1 |  |
| 1. Actie 1 2. Actie 2 3. Actie 3 |  |
| Scherm 2 |  |
| 1. Actie 1 |  |
| 1. Actie .. |  |
| 1. … |  |

De formulieren moeten ondertekend worden door de tester om te bewijzen dat deze naar waarheid ingevuld is. De onderwerpen bestaan uit “Game” of “Website”. De schermen zijn variabel en staan in het vormgevingsdocument. De verschillende acties zijn de acties die door de tester getest moeten worden. Er is ruimte voor de tester om aantekeningen te maken aan de rechterkant van het formulier.

# 2. Eigen klassen



# 3. Controller-klassen

**DesktopLauncher.java**

package johndoe.game.desktop;

import johndoe.game.Config;

import johndoe.game.OilsWell;

import com.badlogic.gdx.backends.lwjgl.LwjglApplication;

import com.badlogic.gdx.backends.lwjgl.LwjglApplicationConfiguration;

public class DesktopLauncher implements Config{

public static void main (String[] arg) {

LwjglApplicationConfiguration configuration = new LwjglApplicationConfiguration();

configuration.width = Config.WIDTH;

configuration.height = Config.HEIGHT;

configuration.title = Config.TITLE + " " + Config.VERSION;

configuration.resizable = Config.RESIZABLE;

configuration.useGL30 = Config.USEGL30;

configuration.vSyncEnabled = Config.VSYNC;

new LwjglApplication(new OilsWell(), configuration);

}

}

# 4. Datasheet

Bestand pad: *[installatie folder]/save/users.txt*

Syntax: USERNAME,SCORE

USERNAME,SCORE

USERNAME,SCORE

USERNAME,SCORE

USERNAME,SCORE

Als dit bestand niet aanwezig is wordt deze vanzelf aangemaakt als de gebruiker het “highscore scherm” bekijkt.

# 5. Codestandaard

<https://google.github.io/styleguide/javaguide.html>

## 1 Introduction

This document serves as the **complete** definition of Google's coding standards for source code in the Java™ Programming Language. A Java source file is described as being in Google Style if and only if it adheres to the rules herein.

Like other programming style guides, the issues covered span not only aesthetic issues of formatting, but other types of conventions or coding standards as well. However, this document focuses primarily on the **hard-and-fast rules** that we follow universally, and avoids giving advice that isn't clearly enforceable (whether by human or tool).

### 1.1 Terminology notes

In this document, unless otherwise clarified:

1. The term class is used inclusively to mean an "ordinary" class, enum class, interface or annotation type (@interface).
2. The term comment always refers to implementation comments. We do not use the phrase "documentation comments", instead using the common term "Javadoc."

Other "terminology notes" will appear occasionally throughout the document.

### 1.2 Guide notes

Example code in this document is **non-normative**. That is, while the examples are in Google Style, they may not illustrate the only stylish way to represent the code. Optional formatting choices made in examples should not be enforced as rules.

## 

## 2 Source file basics

### 2.1 File name

The source file name consists of the case-sensitive name of the top-level class it contains, plus the .java extension.

### 2.2 File encoding: UTF-8

Source files are encoded in **UTF-8**.

### 2.3 Special characters

### 2.3.1 Whitespace characters

Aside from the line terminator sequence, the **ASCII horizontal space character** (**0x20**) is the only whitespace character that appears anywhere in a source file. This implies that:

1. All other whitespace characters in string and character literals are escaped.
2. Tab characters are **not** used for indentation.

### 2.3.2 Special escape sequences

For any character that has a special escape sequence (\b, \t, \n, \f, \r, \", \' and \\), that sequence is used rather than the corresponding octal (e.g. \012) or Unicode (e.g. \u000a) escape.

### 2.3.3 Non-ASCII characters

For the remaining non-ASCII characters, either the actual Unicode character (e.g. ∞) or the equivalent Unicode escape (e.g. \u221e) is used, depending only on which makes the code **easier to read and understand**.

**Tip:** In the Unicode escape case, and occasionally even when actual Unicode characters are used, an explanatory comment can be very helpful.

Examples:

|  |  |
| --- | --- |
| **Example** | **Discussion** |
| String unitAbbrev = "μs"; | Best: perfectly clear even without a comment. |
| String unitAbbrev = "\u03bcs"; // "μs" | Allowed, but there's no reason to do this. |
| String unitAbbrev = "\u03bcs"; // Greek letter mu, "s" | Allowed, but awkward and prone to mistakes. |
| String unitAbbrev = "\u03bcs"; | Poor: the reader has no idea what this is. |
| return '\ufeff' + content; // byte order mark | Good: use escapes for non-printable characters, and comment if necessary. |

**Tip:** Never make your code less readable simply out of fear that some programs might not handle non-ASCII characters properly. If that should happen, those programs are **broken** and they must be **fixed**.

# 6. Test Rollen & Data

In de testomgeving zijn er 2 verschillende rollen:

* De tester
* De developer

**De tester**

Deze zal de applicatie testen met behulp van de testformulieren van de developer. De testomgeving is ook opgezet door de developer.

**De developer**

Deze zal niet aanwezig zijn tijdens de test. De developer zet de testomgeving op en maakt de testformulieren voor de tester.

De testomgeving zal van te voren ingevoerde data bevatten zodat de tester deze kan testen.

Data voor de website database:

* 19, admin, password, [admin@mail.com](mailto:admin@mail.com), administrator, 1
* 38, Jan, password, [jan@mail.com](mailto:jan@mail.com), user, 1
* 39, Peter, password, [peterVos@hotmail.com](mailto:peterVos@hotmail.com), 1
* 40, Bart, password1, [bart.de.graaf@live.nl](mailto:bart.de.graaf@live.nl), 0

Data voor de game score database:

* Jan, 10
* Peter, 25
* Bart, 8