**Assignment #1 : Sudoku Checker**

**Data Structures**

***Functional Specification***

|  |  |
| --- | --- |
| Student: Frederic Melanson |  |
| Instructor: Stephen Johnson |  |
| Information Technology: Electronic Gaming Development |  |
| September 21, 2017 |  |

# Table of Contents

[Table of Contents 2](#_Toc483315682)

[Requirements Summary 3](#_Toc483315683)

[System Requirements Summary 5](#_Toc483315684)

[Use Case Design 6](#_Toc483315685)

[Class Diagram Design 10](#_Toc483315686)

[Activity Diagram Design 11](#_Toc483315687)

[Assumptions and Dependencies 12](#_Toc483315688)

[Feature Cuts and Unsupported Scenarios 12](#_Toc483315689)

[References 13](#_Toc483315690)

# 

# Requirements Summary

Application: Sudoku Checker

Applicaton Overview:

Sudoku is a popular puzzle game where one puts numbers on a grid. The grid has 9 row and 9 columns, and is divided into 9 blocks, each with 9 squares. In each square, one of the numbers from '1' to '9' is placed. For example, the following is an example of a completed Sudoku grid:

123 456 789

456 789 123

789 123 456

234 567 891

567 891 234

891 234 567

345 678 912

678 912 345

912 345 678

The grid is considered to be valid if all of the following rules are met:

1. Each row contains the numbers from 1 to 9 (inclusive) exactly once each.
2. Each column contains the numbers from 1 to 9 (inclusive) exactly once each.
3. Each 3 by 3 block, of which there are 9, contains the numbers from 1 to 9 (inclusive) exactly once each.

Thus, the above example satisfies these rules and can be considered as valid. The grid is considered invalid if any of the rules are broken. For example, the following is an example of an invalid grid:

123 456 789

912 345 678

891 234 567

789 123 456

678 912 345

567 891 234

456 789 123

345 678 912

234 567 891

In this grid, the first block contains the number '1' three times, the number '2' twice, the number '9' twice, and is missing the numbers 4, 5, 6, and 7. For these reasons, among others, the grid is invalid.

\*\* Sudoku is a registered trademark of Nikoli Co. Ltd.

The application will allow for the user to manually enter and save up to three puzzles on file.

A message will be displayed informing the user if the puzzle is valid and if not, what the errors are.

# System Requirements Summary

**Hardware requirements:**

* **Processor:**

1.5 gigahertz (GHz) or faster processor

* **RAM:**

New device: 2 gigabyte (GB) for 32-bit or 64-bit

* **Hard disk space:**

22 GB

* **Graphics card:**

DirectX 9 or later with WDDM 1.0 driver

* **Display:**

800x600

**Software requirements:**

* **Oracle Java SE Development Kit 8u131**
* **Eclipse Neon.2 Release (4.6.2)**
* **Latest version of Windows 10 64-bit**
* **DIA diagram tool v0.97.2**
* **Microsoft Word 365**
* **Photoshop CC 2017 1.1 release**

# Use Case Design

**Name:** ClickNewGame

Participating actors: User

Entry conditions:

* Application is running

Exit conditions:

User clicks the “New Game” button

**Name:** setUser

Participating actors: User

Entry conditions:

* New Game has been chosen

Exit conditions:

User enters his id and clicks ok.

**Name:** LoadGame

Partipating actors: Game

Entry condition:

* User has entered his ID

Exit conditions:

If the user ID exists, do a validation (is this you?), else create a new file for new user.

Name: GenerateTrackList

Participating actors: Game

Entry conditions:

* No track list appropriate for the current level exists

Exit conditions:

A TrackList is generated.

Event flow:

* When the game is launched, a dynamic splash screen is displayed with a New Game button.
* When the user clicks the button, he is prompted to enter an identifier. (nickname, initials etc.)

Event flow:

* When prompted, the user will enter his unique identifier.
* If the identifier matches one of those on file, the user will be asked to confirm if it is him.
* If the identifier is not on file, a new file is created.
* If the identifier is on file, but it is not this user’s file, the user will be asked for a new identifier.

Event flow:

* Using the user’s identifier, the game will load a file containing the user’s current level, score, and remaining guesses per track.
* If the user is not on file, a new file is created.

Event flow:

* Depending on the level, the game generates a list of 3, 5 or 7 track objects each with a distinct random number and its own “tries” counter.

Name: GenerateRandNum

Participating actors: Game

Entry conditions:

* Generating a new level

Exit conditions:

A random number is generated.

Name: setGuessNum

Participating actors: User

Entry conditions:

* No match has been made on current track

Exit condition:

User has entered a guess.

Name: isMatch

Participating actors: Game

Entry condition:

* User has entered a guess

Exit Condition:

Returns true if guess matches the random number.

Name: isHigher

Participating actors: Game

Entry condition:

* isMatch returns false

Exit conditions:

Returns true if guess is higher than the random number.

Event flow:

* Generates a list of 1000 random numbers between one and the current maximum value
* Take whichever number appears most often on the list
* If two numbers have the same appearance frequency, start over.

Event flow:

* User inputs his guess in the appropriate track

Event flow:

* Check user input against random number.
* If they match returns true
* If not returns false

Event flow:

* Check user input against random number
* If user input is higher than random number, return true
* If not, return false

Name: SaveGame

Participating actors: Game

Entry conditions:

* User has completed a level

Exit conditions:

Current level, score and remaining tries for every track are saved in a file.

Name: UpdateUI

Participating actors: Game

Entry conditions:

* Information has changed

Exit Conditions:

Changed information is updated and ready to be displayed

Name: DrawUI

Participating actors: Game

Entry conditions:

* UI has been updated

Exit conditions:

Draw the newly updated UI.

Name: setCurrentTrack

Participating actors: Game

Entry conditions:

* Player has entered a guess on a track

Exit conditions:

A new track is set as the current track.

Event flow:

* Saves level, score and remaining tries for each track in the user’s save file.

Event flow:

* Updates all the information in the user interface.

Event flow:

* Displays the user interface.

Event flow:

* Once the user has guessed a number on a track, the current track is set to the next one in the track array.
* If the current track is the last in the array, set current track to the first one in the array.
* If all tracks are completed, generate next level

# Class Diagram Design

Number Cruncher will have 6 classes, one of which (the level class) will have three subclasses. The “NumberCruncher” class will handle all the displaying and updating of the user interface, the “Game” class will handle the game mechanics such as scoring and our current level, the “Level” class manages all the guessing tracks while “Track” handles the guessing and matching of numbers. “RNG” generates a random number, and “Number” is used to get our modes in “RNG”.

# Activity Diagram Design

Once the user clicks the new game button and inputs their user id, the last level they started is loaded. A new set of tracks is generated, each with a new random number. The User then inputs a guess for the first track, the game checks if it is a match, if it isn’t, it checks if the guess is higher or lower. The game then switches to the next track and starts over. If all tracks are completed, the game saves and generates the next level.

# 

# Assumptions and Dependencies

The lack of tutorial in this game assumes the player has read the present document in order to know how the game works.

# Feature Cuts and Unsupported Scenarios

Some parts of the prototyped user interface suggested in the “Requirement summary” of this document may not be implemented.

# References

<https://www.microsoft.com/en-ca/windows/windows-10-specifications#sysreqs>

<https://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_system_requirements.html#BABHCGGA>