

System-Spezifikation

Hochschule Luzern Technik & Architektur

Programmieren 2 - FS12

Gruppe 10

Redzepi Iljasa | Reichmuth Marco | Rey Philipp | Rohrer Felix

Eine interdisziplinäre Projektarbeit der Studiengänge Elektrotechnik und Informatik.

Autoren

Redzepi Iljas	a	Reichmuth N	Reichmuth Marco	
Studiengang	Informatiker (Berufsbegleitend)	Studiengang	Elektroniker (Berufsbegleitend)	
Adresse		Adresse		
Telefon		Telefon		
E-Mail	iljasa.redzepi@stud.hslu.ch	E-Mail	marco.reichmuth@stud.hslu.ch	
Rey Philipp		Rohrer Felix		
Studiengang	Elektroniker (Vollzeit)	Studiengang	Informatiker (Berufsbegleitend)	
Adresse		Adresse		
Telefon		Telefon		
E-Mail	philipp.rey@stud.hslu.ch	E-Mail	felix.rohrer@stud.hslu.ch	

Änderungskontrolle

Version	Datum	Autor	Beschreibung
1.0	22.04.2012	Felix Rohrer	Vorlage erstellen, erste Texte schreiben
1.1	10.05.2012	Iljasa Redzepi	Kapitel Architektur beschrieben
1.2	10.05.2012	Felix Rohrer	Klassendiagramm, Design-Entscheide
1.3	10.05.2012	Iljasa Redzepi	Sequenzdiagramm erstellt
1.4	10.05.2012	Rey Philipp	Softwareanforderungen und Design-Entscheide
1.5	11.05.2012	Felix Rohrer	CLI Client dokumentiert
-	11.05.2012	Felix Rohrer	Version 1.5 freigeben
1.6	30.05.2012	Felix Rohrer	Klassendiagramm auf die neuste Version angepasst
			GUI – Client dokumentieren
-	31.05.2012	Felix Rohrer	Version 1.6 freigeben

Gruppe 10 I Programmieren 2 - FS12

Inhalt

3.1 Softwareanforderungen 6 3.2 Design-Entscheide 7 4 Environment-Anforderungen 7 4.1 Hardware 7 4.2 Software 7 4.3 Java Virtual Machine 7 5 CLI - Client 8 5.1 Unterstützte CLI Parameter 8 5.2 CLI-Client Bedienung 9 6 GUI - Client 11 6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15	1	Einfü	ihrung	1
2.1 Funktionale Sicht .1 2.2 Entwicklungssicht .2 2.3 Verteilungs- und Betriebssicht .5 3 Softwareanforderungen und Design-Entscheide .6 3.1 Softwareanforderungen .6 3.2 Design-Entscheide .7 4 Environment-Anforderungen .7 4.1 Hardware .7 4.2 Software .7 4.3 Java Virtual Machine .7 5 CLI – Client .8 5.1 Unterstützte CLI Parameter .8 5.2 CLI-Client Bedienung .9 6 GUI – Client .11 6.2 Settings .12 6.3 Cheat / Hilfe .12 6.4 Solver .13 6.5 Design .14 6.6 Help / About .15	2	Arch	itektur-Beschreihung	1
3.1 Softwareanforderungen 6 3.2 Design-Entscheide 7 4 Environment-Anforderungen 7 4.1 Hardware 7 4.2 Software 7 4.3 Java Virtual Machine 7 5 CLI - Client 8 5.1 Unterstützte CLI Parameter 8 5.2 CLI-Client Bedienung 9 6 GUI - Client 11 6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15		2.1 2.2	Funktionale Sicht Entwicklungssicht	1
3.2 Design-Entscheide 7 4 Environment-Anforderungen 7 4.1 Hardware 7 4.2 Software 7 4.3 Java Virtual Machine 7 5 CLI – Client 8 5.1 Unterstützte CLI Parameter 8 5.2 CLI-Client Bedienung 9 6 GUI – Client 11 6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15	3	Softv	wareanforderungen und Design-Entscheide	6
4.2 Software 7 4.3 Java Virtual Machine 7 5 CLI – Client 8 5.1 Unterstützte CLI Parameter 8 5.2 CLI-Client Bedienung 9 6 GUI – Client 11 6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15	4	3.2	Design-Entscheide	7
5.1 Unterstützte CLI Parameter 8 5.2 CLI-Client Bedienung 9 6 GUI – Client 11 6.1 Spielen im GUI 12 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15		4.2	Software	7
5.2 CLI-Client Bedienung 9 6 GUI – Client 11 6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15	5	CLI –	Client	8
6.1 Spielen im GUI 11 6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15		_		
6.2 Settings 12 6.3 Cheat / Hilfe 12 6.4 Solver 13 6.5 Design 14 6.6 Help / About 15	6	GUI -	– Client	11
		6.2 6.3 6.4 6.5	Settings	12 12 13
				16

1 Einführung

Informationen über das System "Mastermind" werden im Dokument "Kundenanforderungen" spezifiziert.

2 Architektur-Beschreibung

2.1 Funktionale Sicht

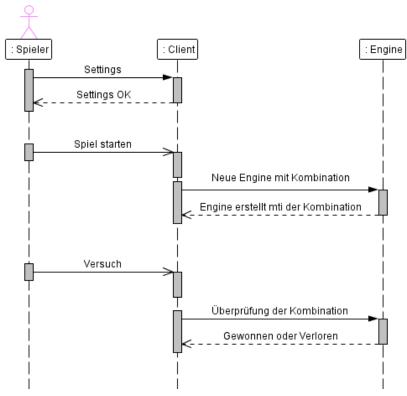


Abb. 1 Mastermind: Funktionale Sicht (Sequenzdiagramm)

Gruppe 10 Seite 1 von 16 Programmieren 2 - FS12

2.2 Entwicklungssicht

2.2.1 Packages Übersicht

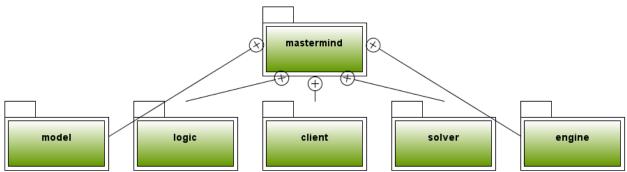


Abb. 2 Mastermind: Package Übersicht

2.2.2 Mastermind Klassenübersicht

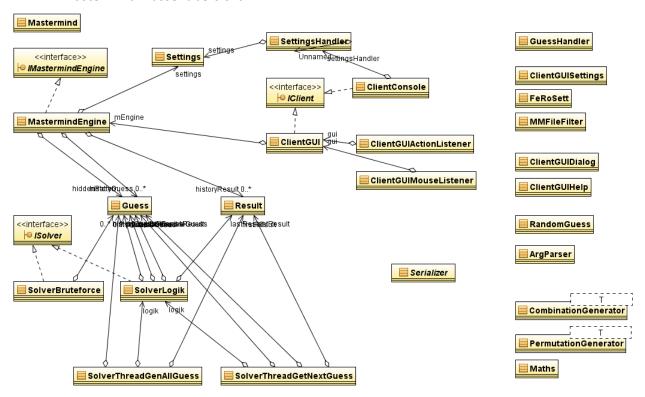


Abb. 3 Mastermind: Klassen Übersicht

2.2.3 Package Model

```
Settings
                                                 Attributes
public int ANZ_COLOR_MIN = 4
public int ANZ_COLOR_MAX = 8
public int ANZ_PEG_MIN = 4
public int ANZ_PEG_MAX = 10
public int ANZ GUESS MIN = 6
public int ANZ_GUESS_MAX = 12
public int ANZ_DESIGN_MAX = 6
private int numberOfColors
private int numberOfPeas
private boolean allowDuplicateColor
private int numberOfRows
private int idDesignUsed
                                                Operations
public Settings( )
public void setSettings(int numberOfColors, boolean allowDuplicateColor, int numberOfPegs, int numberOfRows
public int getNumberOfColors(
public void setNumberOfColors(int numberOfColors)
public int getNumberOfPegs()
public void setNumberOfPegs(int numberOfPegs)
public boolean getAllowDuplicateColor( )
public void setAllowDuplicateColor( boolean allowDuplicateColor )
public int getNumberOfRows()
public void setNumberOfRows(int numberOfRows)
public int getDesign()
public void setDesign(intidDesign)
```

```
Guess

Attributes

private int pegs[0..*]

private int score

Operations

public Guess()

public Guess( int pegs[0..*])

public outses( int pegs[0..*])

public outsetPegs( int pegs[0..*])

public void setPegs( int pegs[0..*])

public void setPegs( int pos, int color)

public int getPeg( int pos)

public int getPeg( int pos)

public string toString()

public String toString()

public Result compareGuess( Guess otherGuess)

public int hashCode()
```

```
### Result

Attributes

private int anzPegColorPosRight = 0

private int anzPegColorRight = 0

**Coperations**

public Result() public Result() int anzPegColorPosRight, int anzPegColorRight )

public void setAnzPegColorPosRight() int anzPegColorPosRight )

public void setAnzPegColorPosRight() int anzPegColorPosRight )

public void setAnzPegColorRight() int anzPegColorRight )

public void incAnzPegColorRight() public void incAnzPegColorRight() )

public void incAnzPegColorRight() )

public string getResultAsString() )

public String toString() )

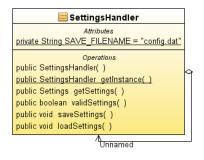
public String toString() )

public boolean equals(Object otherResult) )

public int hashCode()
```

Abb. 4 Package Model: Klassenübersicht

2.2.4 Package Logic







RandomGuess

Attributes

Operations
public Guess genRandomGuess(_)

```
ArgParser

Attributes

private String params[0..*] = new Vector<String>()
private int paramIndex = 0
private int numberOfParams = 0

Operations

public ArgParser( String args[0..*] )
public boolean hasOption( String opt )
public String getOption( String opt )
public String nextParam( )
public int numberOfParams( )
```

Abb. 5 Package Logic: Klassenübersicht

2.2.5 Package Client

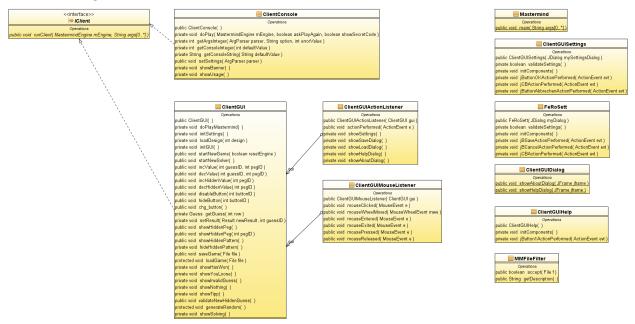


Abb. 6 Package Client: Klassenübersicht

2.2.6 Package Solver

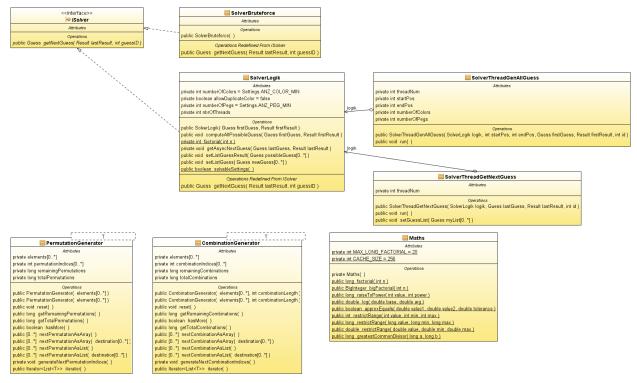


Abb. 7 Package Solver: Klassenübersicht

Gruppe 10 Seite 4 von 16 Programmieren 2 - FS12

2.2.7 Package Engine

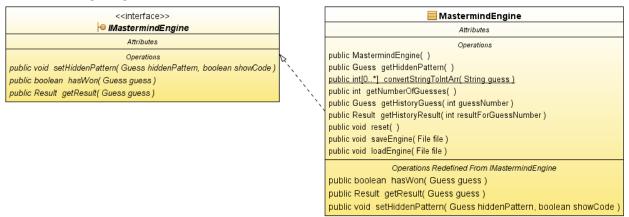


Abb. 8 Package Engine: Klassenübersicht

2.3 Verteilungs- und Betriebssicht

In unserem Projekt wird aktuell die Anwendung nicht auf verschiedene Systeme verteilt. In einem späteren Zeitpunkt, wenn die Option Netzwerk implementiert wird, wird die Anwendung auf verschieden System verteilt.

Gruppe 10 Seite 5 von 16 Programmieren 2 - FS12

3 Softwareanforderungen und Design-Entscheide

3.1 Softwareanforderungen

3.1.1 Nichtfunktionale Anforderungen

Anforderung	Beschreibung
Zuverlässigkeit	Es sollen keine Abstürze des Spiels vorkommen
Benutzerfreundlichkeit	Intuitive und verständliche Oberfläche (GUI)
Effizienz	Systemressourcen werden nicht unnötig ausgelastet
Performance	Anweisungen des Benutzers sollen ohne Verzögerung ausgeführt werden
Wartbarkeit	Nachträgliche und Änderungen sind ohne grosse Umstände möglich
Erweiterbarkeit	Nachträgliche Ergänzungen sind ohne grosse Umstände möglich
Korrektheit	Auf die Ergebnisse/Ausgaben des Spiels ist Verlass

3.1.2 Funktionale Anforderungen

Anforderung	Beschreibung
Spiel Speichern	Aktuellen Spielstand abspeichern
Spiel laden	Gespeicherten Spielstand wieder abrufen
Hilfe	Hilfestellung anzeigen
About-Dialog	Informationen über Spiel anzeigen
Optionen	Rolle und Peg-Anzahl wählen

Gruppe 10 Seite 6 von 16 Programmieren 2 - FS12

3.2 Design-Entscheide

Nach Möglichkeit wird das Model-View-Controller (MVC) Architekturmuster angewendet.

Das Mastermind wird so aufgebaut, dass der "Client" unabhängig von der Engine ist. Die definierten Schnittstellen ermöglichen eine möglichst kleine Abhängigkeit zwischen diesen zwei Logischen Einheiten. Dadurch ist es in einem späteren Entwicklungsschritt kein Problem einen weiteren Client zu implementieren.

3.2.1 Singleton Settings

Für die Einstellungen (Settings) wird das Singleton Entwurfsmuster verwendet. Zusammen mit dem SettingsHandler ist sichergestellt das es jeweils nur eine Instanz von diesem Objekt gibt.

Jeder Softwareteil ist selber bemüht eine entsprechende Referenz anzufordern und die entsprechenden Settings zu verwenden.

3.2.2 Interface zwischen Engine und Client

Das Interface zwischen Mastermind Engine und Client ist so implementiert, dass entsprechende Objekte übergeben werden.

Der Client übergibt der Engine ein "Guess" Objekt, darin enthalten ist eine Zeile, resp. Rateversuch.

Als Rückgabewert liefert die Engine ein "Result" Objekt, darin ist die Anzahl Schwarzer- resp. Weisser-Stifte (richtige Farbe am Richten Ort, resp. richtige Farbe) enthalten.

Durch weitere Methoden, welche von der Engine zur Verfügung gestellt werden, kann z.B. direkt Abgefragt werden ob der Spieler gewonnen hat oder nicht.

4 Environment-Anforderungen

4.1 Hardware

Das Mastermind-Spiel läuft auf fast jeder Hardware. Grundsätzlich muss Java lauffähig sein, an die Ressourcen werden vom Spiel keine speziellen Anforderungen gestellt.

Für den Solver, auf einem 8 Farben mit 8 Position Spielfeld, ist es von Vorteil wenn man ein Multi-Core System mit ausreichend Ram besitzt.

4.2 Software

Das Mastermind-Spiel läuft auf allen Betriebssystemen für welche es Java gibt.

4.3 Java Virtual Machine

Damit das Mastermind-Spiel läuft, muss die JVM installiert sein.

Gruppe 10 Seite 7 von 16 Programmieren 2 - FS12

5 CLI - Client

Abb. 9: CLI - Hilfe / About

5.1 Unterstützte CLI Parameter

5.1.1 -? | Hilfe

Hilfe / Usage anzeigen

5.1.2 -c | Anzahl Farben

Anzahl Farben definieren, z.B. für 4 Farben: -c=4

5.1.3 -d | Doppelte Farben (Ja/Nein)

Doppelte Farben erlaubt, z.B. für "Ja": -d=j

5.1.4 -p | Anzahl Steckplätze

Anzahl Steckplätze definieren, z.B. für 6 Positionen: -p=6

5.1.5 -a | Anzahl Rateversuche

Anzahl Versuche, z.B. für 10 Versuche: -a=10

5.1.6 -s | Secret Code setzen

Das zu erratende Muster definieren, z.B. für 4132: -s=4132

5.1.7 -n | Nochmals spielen (Ja/Nein)

Abfrage ob nochmals spielen, z.B. für "Nein": -n

Gruppe 10 Seite 8 von 16 Programmieren 2 - FS12

5.2 CLI-Client Bedienung

5.2.1 Optionen definieren

Falls die Optionen nicht via Parameter definiert wurden, werden diese Abgefragt. In den Eckigen-Klammern ist der Default-Wert vorgegeben, welcher mittels Return übernommen werden kann.

Sobald alle Optionen definiert sind, wird der "Secret Code" erzeugt.

```
Anzahl Farben (4-8) [4]:
Doppelte Farben (j/n) [n]:
Anzahl Steckplaetze (4-6) [4]:
Anzahl Versuche (1-30) [6]:
Secret Code wird erzeugt...
```

Abb. 10: CLI - Optionen definieren

Wird eine ungültige Option eingegeben, wird dies abgefangen und solange wiederholt bist es eine gültige Eingabe war.

```
Anzahl Farben (4-8) [4]: 2
> Bitte gueltige Anzahl Farben eingeben!
Anzahl Farben (4-8) [4]: -5
> Bitte gueltige Anzahl Farben eingeben!
Anzahl Farben (4-8) [4]: a
> Bitte gueltige Anzahl Farben eingeben!
Anzahl Farben (4-8) [4]: 6
```

Abb. 11: CLI – Ungültige Optionen abfangen

Die Optionen müssen "spielbar" sein, z.B. ist die Kombination mit 4 Farben, ohne Doppelte Farben und 6 Steckplätze nicht möglich. Dies wird solange wiederholt bis eine spielbare Konfiguration eingegeben wurde.

```
Anzahl Farben (4-8) [4]: 4
Doppelte Farben (j/n) [n]: n
Anzahl Steckplaetze (4-6) [4]: 6
Anzahl Versuche (1-30) [6]: 8
Diese Optionen sind nicht gueltig! Bitte neue eingeben...
```

Abb. 12: CLI – Ungültige Kombination abfangen

5.2.2 Spielen

Die Eingabe der Kombination erfolgt als Zahlen, pro Position ist es eine Ziffer.

Für die Kombination 1-2-3-4 wird "1234" eingegeben. Nach der Eingabe erfolgt das Resultat in Anzahl Schwarzer, resp. Anzahl Weisser Stiften.

Abb. 13: CLI - Rateversuche

Falsche und ungültige Eingaben werden hier ebenfalls abgefangen.

```
Versuch 1/6: 123456
> Bitte eine gueltige Kombination eingeben!
Versuch 1/6: asdf
> Bitte eine gueltige Kombination eingeben!
Versuch 1/6: a s d f
> Bitte eine gueltige Kombination eingeben!
Versuch 1/6: 1 2 3 4
> Bitte eine gueltige Kombination eingeben!
```

Abb. 14: CLI – Ungültige Rateversuche abfangen

Gruppe 10 Seite 10 von 16 Programmieren 2 - FS12

6 GUI - Client

Das GUI wird primär mit der Maus bedient. In der obersten Zeile ist das zu erratende Muster. In den darunter liegenden Zeilen können die Rateversuche eingegeben werden. Rechts davon ist der Knopf für die Überprüfung sowie das Ergebnis (Schwarze / Weisse Pins).

Um eine Farbe zu setzen kann die Maustaste oder aber das Mausrad verwendet werden.

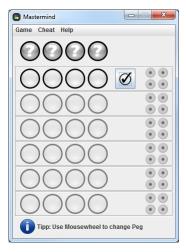


Abb. 15: GUI - Startscreen

6.1 Spielen im GUI

Eine Überprüfung des Rateversuches ist erst möglich nachdem eine gültige Kombination eingegeben wurde. Zuunterst wird jeweils angezeigt ob man Gewonnen oder Verloren hat.

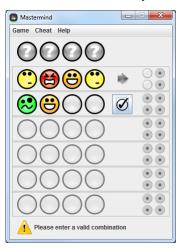


Abb. 16: GUI –Ungültige Eingabe

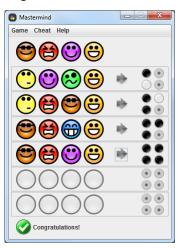


Abb. 17: GUI – Gewonnen

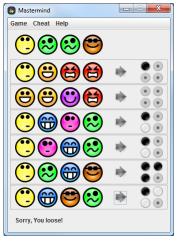


Abb. 18: GUI – Verloren

Gruppe 10 Seite 11 von 16 Programmieren 2 - FS12

6.2 Settings

In den Settings können die verschiedenen Optionen definiert werden:

- Anzahl Farben
- Doppelte Farben
- Anzahl Positionen
- Anzahl Rateversuche
- Design

Es können nur "gültige" (spielbare) Optionen gespeichert werden.

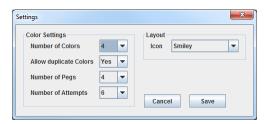


Abb. 19: GUI - Settings Dialog

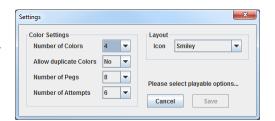


Abb. 20: GUI - Settings Dialog: Ungültige Eingabe

6.3 Cheat / Hilfe

Unter dem Menü punkt "Cheat" kann Wahlweise ausgewählt werden ob nur eine einzelne Position oder die gesamte Lösung angezeigt werden soll.

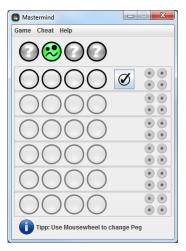


Abb. 21: GUI - Cheat: One Peg

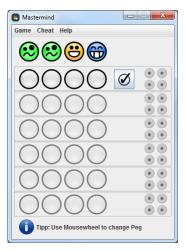


Abb. 22: GUI – Cheat: Ganze Lösung anzeigen

Gruppe 10 Seite 12 von 16 Programmieren 2 - FS12

6.4 Solver

Für den Solver muss zuerst das zu erratende Muster definiert werden. Entweder gibt mal selber eines ein, oder aber lässt eines per Zufall erzeugt.



Abb. 23: GUI - Solver: Set Hidden Pattern



Abb. 24: GUI - Solver: Pattern definieren

Mit der aktuellen Implementation des Solvers ist die komplexeste Version von 8 Farben bei 8 Positionen. Dies ergibt 16'777'216 Möglichkeiten!

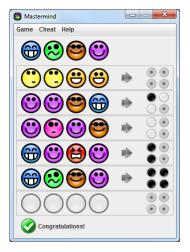


Abb. 25: GUI – Solver: Lösung gefunden

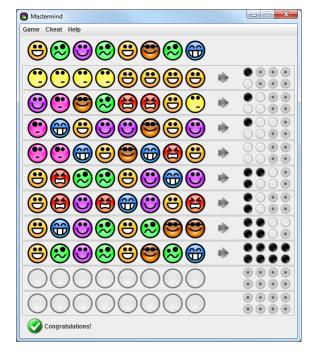
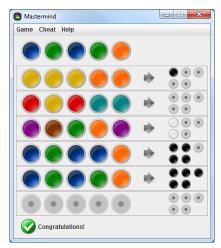


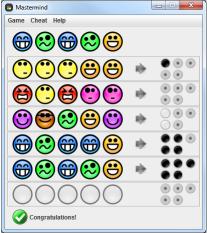
Abb. 26: GUI – Solver: Multi-Thread Solver

Gruppe 10 Seite 13 von 16 Programmieren 2 - FS12

6.5 Design

In der aktuellen Version stehen 6 Verschieden Designs zur Auswahl: Color, Smiley, Cube, Owl, Car, World und Tux.





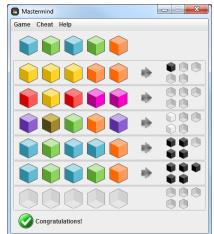


Abb. 27: GUI - Design: Color

Abb. 28: GUI - Design: Smiley

Abb. 29: GUI - Design: Cube



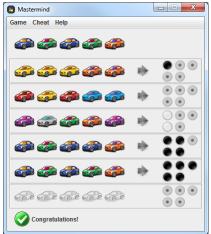




Abb. 30: GUI - Design: Owl

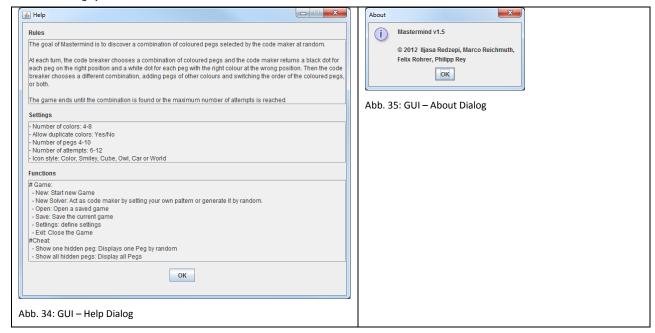
Abb. 31: GUI - Design: Car

Abb. 32: GUI – Design: World



Abb. 33: GUI - Design: Tux

6.6 Help / About



Gruppe 10 Seite 15 von 16 Programmieren 2 - FS12

Abbildungsverzeichnis

Abb. 1 Mastermind: Funktionale Sicht (Sequenzdiagramm)	1
Abb. 2 Mastermind: Package Übersicht	2
Abb. 3 Mastermind: Klassen Übersicht	2
Abb. 4 Package Model: Klassenübersicht	3
Abb. 5 Package Logic: Klassenübersicht	3
Abb. 6 Package Client: Klassenübersicht	4
Abb. 7 Package Solver: Klassenübersicht	4
Abb. 8 Package Engine: Klassenübersicht	5
Abb. 9: CLI – Hilfe / About	8
Abb. 10: CLI – Optionen definieren	9
Abb. 11: CLI – Ungültige Optionen abfangen	9
Abb. 12: CLI – Ungültige Kombination abfangen	9
Abb. 13: CLI – Rateversuche	10
Abb. 14: CLI – Ungültige Rateversuche abfangen	10
Abb. 15: GUI – Startscreen	11
Abb. 16: GUI –Ungültige Eingabe	11
Abb. 17: GUI – Gewonnen	11
Abb. 18: GUI – Verloren	11
Abb. 19: GUI – Settings Dialog	12
Abb. 20: GUI – Settings Dialog: Ungültige Eingabe	12
Abb. 21: GUI – Cheat: One Peg	12
Abb. 22: GUI – Cheat: Ganze Lösung anzeigen	12
Abb. 23: GUI – Solver: Set Hidden Pattern	13
Abb. 24: GUI – Solver: Pattern definieren	13
Abb. 25: GUI – Solver: Lösung gefunden	13
Abb. 26: GUI – Solver: Multi-Thread Solver	13
Abb. 27: GUI – Design: Color	14
Abb. 28: GUI – Design: Smiley	14
Abb. 29: GUI – Design: Cube	14
Abb. 30: GUI – Design: Owl	14
Abb. 31: GUI – Design: Car	14
Abb. 32: GUI – Design: World	14
Abb. 33: GUI – Design: Tux	14
Abb. 34: GUI – Help Dialog	15
Abb. 35: GUI – About Dialog	15