# Lern- und Arbeitsauftrag Modul 404 1401

|  |  |
| --- | --- |
| **Titel:** | **GUI-Projekt** |
| **Modul:** | IMS-Lernatelier Informatiker/in EFZ |
| **Autor / Version:** | Michael Schneider |

## Ausgangslage

In dieser Aufgabe geht es darum, ein grösseres Projekt in WinForms zu lösen, sodass Sie Erfahrungen und Routine sammeln können.

Die folgenden Themen sind normalerweise die Knackpunkte in diesem Modul. Nutzen Sie die Zeit im Lernatelier um sich mit den Punkten zu beschäftigen:

* Komplexere GUI-Komponenten (ComboBox, Listbox, Menü-Strip, Status-Strip, Kontextmenü, Progress Bar, Timer)
* Zustandsdiagramme
* Ausnahmebehandlung mit try und catch
* Korrekte Anwendungsfalldiagramme
* Korrekte Testdokumente

## Rahmenbedingungen

* Sie arbeiten in Dreiergruppen.
* Alle Gruppenmitglieder sind an der Programmierung beteiligt und realisieren mindestens eine klare Programmieraufgabe.
* Sie können selbst ein Projekt aus dem Anhang auswählen.
* Verwenden Sie beim Benutzerinterface WinForms.
* Die zeitlichen Rahmenbedingungen sind im Ablaufplan für Ihre Klasse vorgegeben.
* Der Projektablauf soll nach IPERKA laufen.
* Bewertet wird ein individueller Portfolioeintrag.
* Achten Sie auf eine professionelle Realisierung des Resultats (Titel, Verhalten bei Grössenänderungen, Icon, …).

## Ziele

* Sie wenden IPERKA an.
* Sie erstellen ein mittelgrosses Programm mit GUI.
* Sie wenden GUI-Komponenten korrekt an.

## Aufgaben

### 1) Informieren

1. Schliessen Sie sich in Dreiergruppen zusammen.
2. Wählen Sie in der Gruppe ein Projekt aus (siehe Anhang dieses Auftrages). Beachten Sie den Schwierigkeitsgrad, Ihren Wissensstand und die zur Verfügung stehende Zeit bei der Auswahl.
3. Legen Sie fest, wo und wie Sie in der Gruppe Arbeitsergebnisse speichern und austauschen möchten.
4. Machen Sie eine Anforderungsanalyse. Werden Sie sich in der Gruppe einig, was Sie genau realisieren möchten. Das Projekt sollte durch die Anforderungen so präzise wie möglich beschrieben sein. Legen Sie die Muss-Anforderungen und Kann-Anforderungen fest, die bei genügend Zeit realisiert werden können.
5. Erstellen Sie Testfälle für das gewünschte Programm.
6. Erarbeiten Sie sich noch fehlende Grundlagen für das Projekt.
7. Erstellen Sie ein Anwendungsfalldiagramm und ein Zustandsdiagram für Ihr Projekt.

### 2) Planen

1. Teilen Sie das Projekt in klare, einzelne Arbeitspakete auf.
2. Verteilen Sie diese Arbeiten an die Gruppenmitglieder und setzen Sie für jede Arbeit ein Abschlussdatum fest, das mit dem Plan Ihrer Klasse übereinstimmt.
3. Listen Sie Ihre alle Arbeitspakete jeweils mit verantwortlicher Person, Soll-Aufwand und Abschlussdatum auf.

### 3) Entscheiden.

Fällen und dokumentieren Sie wichtige Entscheidungen.

### 4) Realisieren

Erstellen Sie das Programm und die notwendigen Dokumente. Erstellen Sie das Programm selbstständig und verwenden Sie nicht einfach das Resultat eines Lernvideos oder fertigen Code.

### 5) Kontrolle

Testen Sie Ihr Programm mittels der erstellten Testfälle und schreiben Sie ein kurzes Testprotokoll.

### 6) Auswertung

Überlegen Sie sich, was gut gelaufen ist und was eher nicht. Sie können diese Informationen in Ihrem Portfolioeintrag verwenden.

### 7) Portfolioeintrag

Schreiben Sie den Portfolioeintrag und reichen Sie ihn ein.

## Gütekriterien

Der Lern- und Arbeitsauftrag ist erfüllt, wenn …

* Wenn Sie das Projekt nach IPERKA abgearbeitet haben.
* Wenn Sie eine lauffähige Applikation produziert haben.

## Zusätzliche Angaben zum Auftrag

Die beigefügten Abbildungen dienen nur zur Illustration, Ihr Programm kann sich auch davon unterscheiden.

### Hacking - Knacken Sie .zip-Dateien

Sie haben im Modul 114 gelernt, wie man komprimierte Dateien verschlüsseln und mit einem Passwort versehen kann. Eine so verschlüsselte Datei kann mit brute force angegriffen werden, indem einfach möglichst viele Passworte ausprobiert werden können.

Die Library DotNetZip[[1]](#footnote-1) erlaubt das Entschlüsseln von .zip-Dateien mit Passwörtern.

Erstellen Sie eine Applikation, die es erlaubt eine .zip-Datei auszuwählen und verschiedene Strategien anwenden kann:

* Alle Varianten von einer Start- zu einer End- Zeichenkette, bspw. von aaaaaa bis ZZZZZZ.
* Eine einstellbare Anzahl zufälliger Passwörter soll ausprobiert werden. Es gibt verschiedene Listen[[2]](#footnote-2),[[3]](#footnote-3) mit bekannten Passwörtern, welche Ihre Applikaton laden und ausprobieren soll.

Geben Sie aus, wie viele Passworte pro Sekunde das Programm ausprobieren kann.

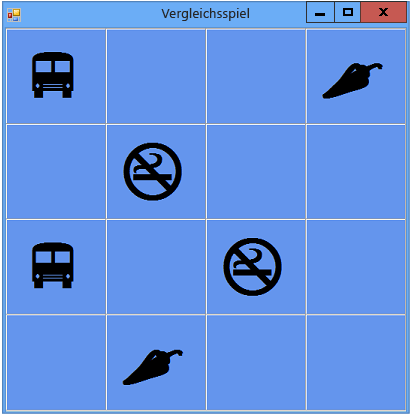
### Einfache Spiele

Implementieren Sie ein einfaches Spiel, das nach den bekannten Regeln gespielt werden kann. Das Spielewiki[[4]](#footnote-4) hat viele kleine Spiele, die man als WinForms-Applikation umsetzen kann; oder Sie wählen ein Spiel aus den Untenstehenden aus.

Für alle Spiele wäre es interessant, wenn die Punkte dargestellt werden, es eine Highscore-Liste gäbe und aktuelle Spielstände geladen und gespeichert werden könnten.

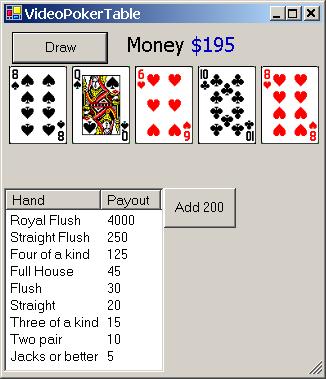
**Memory**

Bei Memory werden Karten gemischt und verdeckt hingelegt. Immer zwei der Karten sind gleich. Die Spieler dürfen nun reihum zwei Karten aufdecken. Finden sie ein Paar, dann erhalten sie einen Punkt und dürfen nochmals aufdecken. Wer am meisten Punkte hat, wenn keine Karten mehr auf der Spielfläche sind, hat gewonnen.



**Poker**

Es gibt viele verschiedene Varianten von Poker, die einfachste ist "Videopoker", das normalerweise auf einem Glücksspielautomaten gespielt wird. Der Spieler erhält 5 Karten und kann danach wählen, welche Karten er ausgetauscht haben möchte. Je nach Karten erhält er Geld oder verliert seinen Einsatz.

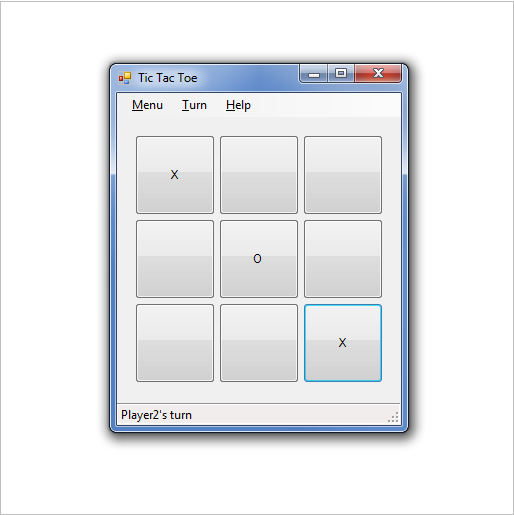


**Bauernkrieg[[5]](#footnote-5)**

Ein sehr einfaches Kartenspiel, das vollständig vom Glück und nicht vom Können abhängt:

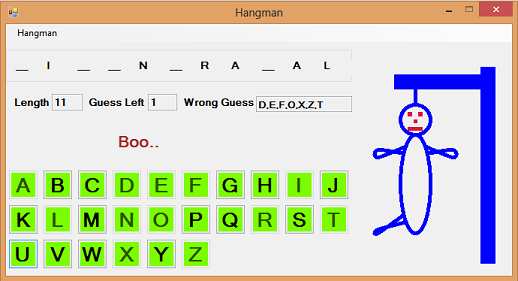
**TicTacToe**

Das allseits bekannte und beliebte Spiel in dem drei Symbole in eine Reihe gebracht werden müssen. Interessant wäre hier ein Computergegner.



**Hangman**

Bei Hangman wird für ein Wort die Anzahl Buchstaben vorgegeben. Der Spieler darf nun eine gewisse Anzahl Buchstaben raten und mit möglichst wenig Rateversuchen das Wort herausfinden. Die Wörter sollten aus einer Textdatei geladen werden können.



### Quiz

Der Benutzer soll ein Multiple Choice-Quiz erstellen, speichern und zum Ausfüllen gegeben werden können.

1. <https://github.com/haf/DotNetZip.Semverd> [↑](#footnote-ref-1)
2. <https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials> [↑](#footnote-ref-2)
3. <https://github.com/danielmiessler/SecLists/tree/master/Passwords> [↑](#footnote-ref-3)
4. <https://www.spielewiki.org/wiki/Hauptseite> [↑](#footnote-ref-4)
5. [https://de.wikipedia.org/wiki/Bataille\_royale](https://www.spielewiki.org/wiki/Hauptseite) [↑](#footnote-ref-5)