Das Programm Dame soll ein Spiel sein. Es basiert auf einem Brettspiel und befolgt alle seine Regeln. Spielzüge, die nicht Regelkonform sind werden nicht ausgeführt. Es gibt eine Ausnahme! Es wird nicht vorgeschrieben, dass Spieler abwechselnd ihre Züge machen, weil es Situationen gibt, bei dem auch mehrere Züge hintereinander gemacht werden können. Ist es nämlich möglich nach einem schlagen eines Steines noch ein Stein geschlagen werden kann muss dies gemacht werden. Auch die Regen, dass man dazu gezwungen ist einen Stein zu schlagen, wenn man dies auch kann ist nicht dabei.

Das Spiel wird zu zweit gespielt und jeden hat seine eigenen Spielfiguren. In dem Fall sind es X und O. Sie werden im Code çoban genannt. Hinzu kommt, wenn man auf das Ende der gegnerischen Seite gelangt, dass sich diese Figur Weiterentwickelt zu kraliçe. Für diese Figur gelten nun andere Regeln und die Spielfigur wird in Fett dargestellt

Çoban:

1. Darf nur in eine Richtung laufen.
2. Darf nur ein Feld laufen.
3. Darf nur Diagonal laufen.
4. Beim Schlagen wird der Platz hinter dem Geschlagenen Stein eingenommen.
5. Es darf nur geschlagen werden, wenn der Platz dies auch zulässt.

Kraliçe:

1. Sie darf auch zurück Laufen.
2. Die darf so viele Felder laufen wie möglich.
3. Die darf keine Spielsteine beim Laufen überspringen werden.
4. Sie erbt die Regeln von Çoban.

Dar ich die Felder nach A1, A2, B1, B2… benannt habe um x -und y-Achse besser voneinander zu unterscheiden. Gibt es zwei Dictionary, die für Schleifen Chars in Int umwandeln. Dies ist nicht unbedingt nötig, weil Char auch als Zahlenwert dargestellt wird. Allerding ist die bei einem A nicht 0. Also habe ich mich dafür entschieden den Char A als int 0 zu definieren. Ein weiteres Problem ist, dass der Char 0 als int auch nicht 0 ist. Auch wenn man Convert.toInt16(Char 0) benutzt, ist der Wert Chart 0 als int > int 0.

translate4index[Char, int]

Mit der Funktion Controls.Find() kann nach einem Objekt mit einem Namen gesucht werden. Darum habe ich ein weiteren Dictionary benutzt um int wieder in chars zu übersetzen.

Die ersten beiden Schleifen in Meinem Programm benutze ich, um das Spielfeld zu füllen. Dabei ist es Wichtig, dass bei Dame nur eine Farbe der Spielfeder für das Spiel benötigt wird. Ich habe mich für die Farbe Weiß entschieden und die Felder abwechselnd Schwarz und Weiß eingefärbt. Die O Figuren geben bis C und X fangen bei F an. Ich habe mich dafür entschieden, weil so das Basic Spielfeld für mehrere Spiele, Farben genutzt werden kann. Theoretisch kann man so in einem Menü entscheiden, wie das Spielfeld gefüllt werden soll.

Wichtig ist auch zu wissen, dass in Forms mit einem EventHandler jeder Form eine eigene Funktion bekommt, die ausgeführt wird, wenn die Form ausgelöst wird. Die Funktion kann selbst benannt werden. In der Datei Form1.Designer.cs mit this.[Form].[Funktion] += new System.EventHandler(this.[Funktion]); in meinem Fall ist [Form] alle Knöpfe (A0, A1, …) und [Funktion] Click. Wir das Event von z.B. der Form A0 ausgelöst wird die Funktion Click ausgeführt. Die Funktion Click wird in der Datei Form1.cs weiter beschrieben. Dabei hat die Funktion Click() zwei Werte die übergeben werden. In dabei ist objekt sender essenziell. Dieses Objekt beinhaltet den Formnamen z.B. A0.

Jetzt benötige ich nur noch eine Globale Variable, die speichert welche Form als letztes geklickt wurde. Ich habe sie clicked\_button benannt und wird mit dem Rückgabewert von der Funktion Playground\_Click gefüllt.

PLayground\_Click werden zwei Werte übergeben. b.Name und clickes\_button. b.Name ist im Prinzip die Varriable sender von der Formfunktion Click. Es muss nur vorher gesagt werden worum es sich bei sender handelt. Da sender ein objekt ist es sozusagen ein Variable ohne Typ. Mit sender is Button wird überprüft ob es sich auch um ein objekt Button handelt.

In der Funktion Playground\_Click wird als erstes Überprüft, ob clicked\_button schon einen Buttonnamen enthält. Wenn dies nicht der Fall ist wird b.Name aka button zurückgegeben und clicked\_button wird damit gefüllt. So hat nun clicked\_button einen Wert, so dass bei dem nächsten Klick Playground\_click einen schon geklickten Knopfnamen übergeben bekommt und somit Weiß, dass der Stein von clicked\_button nach button verschoben werden soll.

Der nächste Schritt ist zu Überprüfen ob der Zug Regel konform ist. Als erstes werden grobe Regel Überprüft. Welche Regen genau stehen bei den Bedingungen im Programmcode. Danach wird festgestellt in welche Richtung sich der Spielstein bewegen möchte. Da das Spielfeld eine x -und y-Achse besitzt gibt es auch 4 Bewegung Richtungen, die in sing\_letter und sing\_number als 1 oder -1 gespeichert werden. Dies ist extrem wichtig.

Jetzt wo wir festgestellt haben in welche Richtung die Steine unterwegs sind können wir weitere Spielregen überprüfen. Als Dame darf am auf dem Weg zum Ziel keine Steine überspringen. Mit einer Schleife, laufen wir den weg ab, bis das Ziel erreicht ist. Da wir wissen in welche Richtung die Figur unterwegs ist, können wird die Vorzeichen der x –und y-Achse so ändern, dass wird genau den Laufweg der Figur abgehen. Dadurch das die Figur nur Diagonal laufen kann brauchen wird zum Glück nur eine Schleife. Ist auf dem weg schon eine Figur, wird mit goto die Funktion Playground\_Click beendet.

Als nächstes wird überprüft ob an dem Ziel ein Gegner ob nichts auf einen wartet. Wenn dort nichts ist, wird die Figur einfach an die neue Stelle verschoben und aus der Alten gelöscht. Wenn dort ein Gegner ist muss der Gegner übersprungen werden. Wenn dies nicht klappt, weil hinter den Gegner sich noch eine Figur befindet, wird mit goto die Funktion beendet. Wenn es klappt, dann überspringt, die Figur den Gegner und der Gegner wird entfernt. Erreicht die Spielfigur beim Überspringen die Seite des Gegners wird sie zur Dame aka Kraliçe. Diese ganzen Vorgänge befindet sich in try. Wenn sich nämlich der Gegner am Rand befindet wurde man keinen Button dahinter finden und vom Rand fallen. Bei einem Error wird die Funktion beendet.

Bei Einer normalen Bewegung wird noch einmal überprüft, ob sich der Stein jetzt am Rand des Gegners befindet und gegebenen Falles zur Kraliçe.

Am Ende wird clicked\_button geleert, um das Versetzen eines anderen Spielsteines zu ermöglichen.

Es können auch ein Spielstand gespeichert und geladen werden.

Beim Speichern wird jeder Button abgefragt und der Text und Font wird in jeweils einer Zeile gespeichert. Am Ende wird dann noch der Score der Teams gespeichert.

Beim Laden wird jede Zeile in die Buttons geschrieben. Leider wird Font in der Datei als String gespeichert und es gibt keine Methode, um einen String einfach wieder in ein Font Objekt zu verwandeln. Darum Vergleiche ich den Dateien String und den Font String, wenn sie gleich sind, wird der Button Font zu dem Objekt Font der gleich den Dateien String ist. Am Ende wird dann wieder der Score String in die jeweiligen Labes eingefügt.