Start-Guide für „Shogi – The Game“

**Projektstruktur**

In der Dateistruktur des Projektes gibt es zunächst die Ordner „src“ und „WebContent“. Im „src“-Ord­­ner befinden sich alle Java-Dateien geordnet nach ihrer Funktion in der Web Anwendung:

1. „beans“: Beinhaltet die Java Bean die die Daten und Spiellogik speichern.
2. „dataclasses“: Beinhaltet die Java Klassen, die für die Spiellogik relevant sind und diejenigen, die zur objektorientierten Darstellung eines Shogi-Spiels benötigt werden.
3. „enums“: Beinhaltet die Enums, die die zur Unterscheidung von Spielfiguren nötig sind.
4. „Servlets“: Beinhaltet das Servlet, das die Schnittstelle zwischen Java Bean und JSPs ist.
5. „util“: Beinhaltet die Java Klassen, die für bestimmte Funktionalitäten in „dataclasses“ be­nö­tigt werden.

Im „WebContent“-Ordner befinden sich die JSPs, CSS-Dateien sowie der „res“-Ordner mit den Bild­dateien, die für die Darstellungen der Spielfiguren benötigt werden. Zudem findet sich darin ein Ordner mit Bezeichnung „WEB-INF“, der das web.xml-file beinhaltet sowie im Ordner „lib“ die jstl-library.

**Startseite**

Die Startseite der Web Anwendung ist die JSP „index.jsp“. Darauf kann der Spielmodus über ein For­mular ausgewählt werden, welches auf das Servlet „Controller.java“ verweist. Momentan ist nur der Knopf für den Einzelspielermodus sinnvoll nutzbar. Weiterhin gibt es auf dieser, wie auf al­len anderen Webseiten eine Fußzeile, in der Links zum Impressum („Impressum.jsp“) und zur Spiel­­anleitung („Instructions.jsp“) führen. Die Webseite des Impressums besitzt keine eigenen Funk­­tionen und soll deswegen nicht weiter erläutert werden.

**Spielanleitung**

Auf der JSP der Spielanleitung befindet sich eine Erklärung der Regeln des Spiels „Shogi“ in Form von Volltext. Zusätzlich gibt es die Möglichkeit, sich die Zugmöglichkeiten und Ver­bes­ser­ten Zustände aller Spielfigur-Typen anzeigen zu lassen. Dafür wurde für jede Spielfigur eine qua­­dratische Tabelle erstellt, die in der Farbe Weiß die Zugmöglichkeiten dieser Spielfigur an­zeigt. Zu jeder Spielfigur wird eine äquivalente Tabelle für den verbesserten Zustand angezeigt. Nur bei dem König und dem Goldenen General gibt es keinen verbesserten Zustand. Über ein Drop­­down Menü kann eine Figur ausgewählt werden, deren Zugmöglichkeiten und Ver­bes­ser­ten Zustand man sich anzeigen lassen möchte. Dadurch wird eine JavaScript function getriggert, die alle Tabellen bis auf die gewählten über das CSS Attribut „display“ ausgeblendet.

Ei­ne weitere Funktion dieser Webseite ist, dass durch einen Click auf den Button „Ansicht wech­seln“ die Optik der Spielsteine geändert werden kann. Clickt man auf diesen Button wird eine Ja­va­­Script function aufgerufen, die das für die Spielfiguren zuständige Stylesheet der Webseite än­­dert. Diese Funktion wurde auch auf dem Spielbrett eingesetzt.

**Spielsitzung erstellen (playerMenu.jsp)**

Über den Button „Einzelspieler“ auf der Webseite „index.jsp“ gelangt man auf die Webseite „player­­Me­nu.jsp“. Auf dieser Webseite wird zuerst mit dem Jsp-Tag „<jsp:useBean>“ eine neue Ja­va­ Bean für die Spiele Session angelegt. Danach wird der Spieler dazu aufgefordert, einen User­namen für die Sitzung anzugeben und eine Farbe (Team) zu wählen. Diese Eingabewerte wer­den über ein Form aufgenommen und weitergesendet.

**Controller (Servlet.java)**

In diesem Projekt gibt es nur ein Servlet („Controller.java“), welches für alle Webseiten zu­stän­dig ist. Dazu wird anhand eines sog. „referers“ aus dem Header der Request-Instanz he­raus­ge­le­sen, welche Webseite auf den Controller verwiesen hat und darauf basierend entschieden wo­hin weitergeleitet werden muss. Dabei sind alle Weiterleitungen sog. „Redirects“. In der Ge­sam­ten Web Anwendung wird nur mit Http-GET-Requests gearbeitet.

Zu diesem Zeitpunkt kommt zum ersten Mal das Servlet sinnvoll zum Einsatz. Zwar wird das Serv­­let auch schon für den Wechsel von index.jsp auf die Folgeseiten eingesetzt, dabei hat das Serv­­let jedoch keinen echten Wert. Die erste Sinnvolle Aufgabe des Servlets ist es, das Attribut „Game­­Bean“ des Servlets zu initialisieren und dabei die übergebenen Parameter „User­na­me“ und „Farbe (Team)“ in die Java Bean einzutragen. Dazu wird für das gewählte Team eine neue Instanz der Klasse „Player.java“ mit dem gewählten Usernamen erstellt. Dabei darf der User­­name nicht „AI“ sein. Für die andere Farbe wird eine Instanz der Klasse „AI.java“ angelegt, de­­ren Username „AI“ ist.

Bei erfolgreichem Setzen der Spieler in der GameBean wird auf das Gameboard verwiesen, wo das eigentliche Spiel stattfindet. Auf dem Gameboard werden Spielzüge ausgeführt. Diese äh­neln dem Schach (z. B. C1 auf C2). Es gibt außerdem noch die Möglichkeit Figuren aus dem Ge­fäng­nis zurück zu holen und Figuren, die im hinteren Drittel des Spielfelds stehen, zu ver­bes­sern. Wenn ein Spielzug auf der Webseite „board.jsp“ abgeschlossen wurde, so wird eine Http-Re­quest an das Servlet versendet. Darin sind die X- und Y-Werte einer Koordinate für die Po­si­tion an der eine Spielfigur verbessert werden soll, für die Position, von welcher eine Figur aus be­wegt werden soll und die Position, wo die Spielfigur hinbewegt werden soll.

Die­se Http-Request wird von der doGet() Methode des Servlets entgegengenommen. Zuerst wer­den nun die Koordinaten in neue Instanzen der Klasse „Vector2.java“ eingefügt und es wird be­stimmt, welcher Spieler kein Computer-Spieler ist. Diese Werte werden für die folgenden Be­rech­nungen und Überprüfungen benötigt.

Der nächste Code-Block überprüft, anhand der Koordinate der Position, an der eine Spielfigur ver­bessert werden soll, ob eine Figur verbessert werden soll. Der Standard-Wert ist (-1,-1). Wird dieser Wert übergeben, so soll keine Figur verbessert werden.

Da auf der Webseite „board.jsp“ die X-Werte einer Koordinate in Form von Buchstaben an­ge­ge­ben werden, gibt es eine Methode im Servlet, die dafür sorgt, dass alle Buchstaben in einen pas­sen­den Zahlenwert überführt werden. Dadurch wird der Buchstabe „A“ zu einer „0“ und der Buch­stabe „I“ zu einer „8“. Sollte ein Buchstabe alphabetisch hinter dem „I“ oder irgendein an­de­rer invalider Wert eingegeben werden, so wird daraus eine „-1“. Damit keine Spielfigur ver­bes­sert wird, muss also nur ein Buchstabe alphabetisch hinter dem „I“ als X-Wert und als Y-Wert ei­ne „-1“ eingetragen werden.

Nun folgen noch Überprüfungen, ob die Figur im hinteren Drittel des Spielfelds steht, und ob die Figur dem Team des aktuellen Spielers angehört. Abschließend wird überprüft, ob die Figur ei­nen verbesserten Zustand hat und die Figur verbessert. Da man seine Figur zusätzlich zu einer Be­wegung im oder auf das Spielfeld verbessern kann, ist dieser Code-Block unabhängig von dem folgenden Code-Block.

Der nun folgende Code-Block ist für das Bewegen einer Spielfigur im und auf das Spielfeld zu­stän­dig. Zuerst wird die Ausgangsposition der Spielfigur überprüft. Sollte sie sich in dem Spiel­feld befinden, so werden Bewegungen in dem Spielfeld durchgeführt. Sollte die Aus­gangs­po­si­tion der Bewegung sich nicht in Spielbrett befinden, so wird eine Figur aus dem Gefängnis ein­ge­setzt.

Bei Bewegungen im Spielfeld wird nun überprüft, ob sich an der angegebenen ausgehenden Stelle eine Spielfigur befindet und ob diese auch dem Team des aktuellen Spielers angehört. Dann wird überprüft, ob die Zielposition im Spielfeld liegt und ob diese besetzt oder leer ist. Soll­te sie besetzt sein, so darf die Spielfigur an der Zielposition nicht dem gleichen Team an­ge­hö­ren, wie die Figur an der Ausgangsposition. Nun wird überprüft, ob der Zug für den Typ der Spiel­figur an der Ausgangsposition überhaupt möglich ist. Danach wird nun der Zug ausgeführt und die GameBean verändert.

Nach dem Zug des menschlichen Spielers wird nun überprüft, ob ein Spieler „Schachmatt“ ist. Da­­rauf basierend wird an eine von zwei Webseiten weitergeleitet („Lose. jsp“ oder „Win.jsp“). Die­se beiden Webseiten haben keine wichtige oder interessante Funk­tion und sollen deswegen nicht weiter behandelt werden.

Wenn kein Team „Schachmatt“ ist, so führt der Computer-Spieler einen zufälligen, möglichen Zug auf einer zufällig gewählten Figur seines Teams aus. Es besteht jedoch die Präferenz, dass der Computer-Spieler sich immer erst aus dem „Schach-Zustand“ rettet.

Der folgende Code-Block ist für das Bewegen einer Spielfigur auf das Spielfeld zuständig. Hier kommt wieder die oben angesprochene Methode zum Einsatz, die einem Buchstaben einen Zah­len­wert zuordnet. Denn sollte der X-Wert der Ausgangsposition „-1“ sein und der Y-Wert zwi­schen „-1“ und „-7“, so wird eine Figur aus dem Gefängnis geholt. Dazu wird jedem Y-Wert ein „Fi­gure­Type“ zugeordnet (z. B. -6 = Bauer). Für jeden Fall wird in dem Gefängnis des aktuellen Spie­lers nach dem Index einer Figur des übergebene „FigureTypes“ gesucht. Sollte eine pas­sen­de Figur im Gefängnis vorhanden sein, so muss noch die Zielposition überprüft werden. Diese muss sich auf dem Spielbrett befinden und leer sein. Zudem darf nach der Bewegung der geg­ne­rische Spieler nicht im „Schach“ stehen. Dann wird die Figur bewegt und die GameBean ver­än­dert. Dieser Ablauf wird für beide Teams ermöglicht, je nachdem, welchem Team der mensch­li­che Spieler angehört.

Abschließend wird ein Redirect auf die Webseite „board.jsp“ durchgeführt. Dort wird der ak­tu­el­le Stand der GameBean dargestellt und der Prozess, nach Absenden einer Http-Request, er­neut durchlaufen.

Die letzten Zeilen der doGet-Methode sind für den Fall, dass man auf die Jsps „Win.jsp“ oder „Lose.jsp“ gelangt ist. Diese Zielen Code sind jedoch nicht besonders interessant und sollen nicht weiter beleuchtet werden.

**Spielbrett**

Die Webseite „board.jsp“ ist die Webseite, auf der das Spielen einer Partie Shogi stattfindet. Hier finden sich neben den standardisierten Style-Elementen im oberen Teil und in der Fußzeile der Webseite mehrere Eingabefelder und zwei Knöpfe. Direkt darunter sind das Spielbrett und die Gefängnisse zu finden.

Die Eingabefelder sind über eine Tabelle in dem vorliegenden Format angeordnet. Die beiden Knöpfe und die Eingabefelder sind „input-Elemente“ in einem „form“, dass bei einem Klick auf den Knopf mit Bezeichner „Senden“ die eingegebenen Daten an das Servlet sendet. Neben diesem Knopf befindet sich ein weiterer Knopf, der dem Knopf auf der Webseite „Instructions.jsp“ gleicht und das äußere Erscheinungsbild der Spielfiguren ändert. Die versendeten Daten sind zwei oder drei Koordinaten von Feldern auf dem Spielfeld. Ganz links auf der Webseite finden sich zwei optionale Eingabefelder, die dazu da sind, eine Figur zu verbessern. Dazu muss lediglich die Position der Figur auf dem Spielbrett abgelesen und eingetragen werden (z. B. A4).

Rechts davon sind vier obligatorische Eingabe Felder, die dazu da sind, Spielfiguren auf das (Gefängnis 🡪 Spielfeld) oder in dem Spielfeld (Spielfeld 🡪 Spielfeld) zu bewegen. In beiden Fällen muss eine zulässige Zielposition (Felder ganz rechts) gewählt werden. Im ersten Fall heißt das, dass die auf das Feld platzierte Figur den gegnerischen König nicht direkt ins Schach setzen darf. Jede andere, freie Position ist erlaubt. Im zweiten Fall darf die Zielposition von einer gegnerischen Figur besetzt oder leer sein. Wichtig ist außerdem, dass die benutzte Figur sich an diese Stelle bewegen kann („FigureType“). Dafür ist die Ausgangsposition und der FigureType dieser Figur relevant.

Die Ausgangsposition einer Bewegung wird in den mittleren Eingabefeldern eingetragen. Für den Fall, dass eine Figur aus dem Gefängnis geholt werden soll, gibt es eine kurze Erklärung am rechten Rand der Webseite. Denn um eine Figur aus dem Gefängnis zu holen muss die X-Ordinate außerhalb des eigentlich gültigen Bereichs (A - I) liegen (z. B. „Z“). Außerdem muss die Y-Ordinate der Ausgangsposition einen Wert zwischen -1 und -7 einnehmen. Diese Werte sind eindeutig einem „FigureType“ zugeordnet. Am rechten Rand ist die Zuordnung dargestellt. Für den zweiten Fall muss lediglich die Position einer eigenen Figur auf dem Spielfeld eingetragen werden (z. B. B2).

Das darunter liegende Spielfeld ist mit einem „table“ erstellt worden. Darin befinden sich die Spielsteine, die als „div-Elemente“ erzeugt werden. Die nebenliegenden Gefängnisse sind eine Art Liste, die jede Art von Spielstein („FigureType“) beinhaltet und die Anzahl der gleichartigen Spielsteine zählt. Da die Webseite „board.jsp“ die einzige Webseite ist, die für das Spielen einer Partie Shogi benötigt wird, muss sich der Zustand des Spielbretts (Position der Figuren) und der Gefängnisse bei jedem Aufruf dynamisch visualisieren lassen. Das funktioniert über die „jstl-core-Tags“ und Expressionlanguage. Der Zustand des Spielfelds beschreibt die aktuelle Position der Figuren auf dem Spielfeld.

Beim Erzeugen des Spielfelds werden zwei „For-Schleifen“ durchlaufen. So kann das zweidimensionale Array in der GameBean durchlaufen werden, dass die Position der Spielfiguren speichert. Bei jedem Durchlauf der äußeren „For-Schleife“ wird dabei eine Zeile („tr“) der zu Grunde liegenden Tabelle erstellt. Bei jedem Durchlauf der inneren Schleife wird eine Zelle („td“) in der Tabelle erzeugt. Jede Zelle bekommt als „id“ eine eindeutige Zuordnung eines Buchstabens von A – I zu einer Zahl von 0 – 8. Falls sich in diesem Feld in der GameBean eine Figur befindet, so wird ein passendes „div-Element“ erzeugt, dass mit zwei veränderlichen Style-Klassen ausgestattet ist. Die eine gibt an, in welche Richtung eine Spielfigur zeigen muss, also zu welchem Team die Figur gehört. Die andere gibt an, welches Hintergrundbild der Spielstein haben soll, also zu welchem FigureType die Figur gehört. Beim Erzeugen des Gefängnisses wird lediglich die jeweilige Anzahl der gefangenen Spielfiguren eines Typs dynamisch erzeugt.

**Java Beans, Klassen und Enums**

Das Produkt dieses Projekts arbeitet mit einer zentralen Instanz der „GameBean.java“, die auf der Webseite „playerMenu.jsp“ instanziiert und in der Session gespeichert wird. Somit können meh­rere User gleichzeitig die Anwendung unabhängig voneinander nutzten.

Diese GameBean hat mehrere Attribute, von denen die meisten Instanzen von Java Klassen sind, die vom Projekt-Team angelegt wurden. Zusammengenommen stellt die GameBean das gesamte Spiel­brett mit Figuren und Gefängnissen, die beiden Spieler und eine „History“ der vergangenen Zü­ge dar. Letzteres sollte für einen „undo-Button“ genutzt werden, ist jedoch noch nicht rea­li­siert worden.

Zu allen Java Beans, Klassen und Enums liegt eine umfangreiche JavaDoc vor. Zudem sind die Na­men der Methoden immer selbsterklärend gewählt, sodass der Code gut lesbar ist. Deshalb soll hier keine weitere Erläuterung der Java Beans, Klassen und Enums folgen. Auch das Servlet wur­de kommentiert und sollte somit gut verständlich sein.

**CSS**

Das Produkt nutzt mehrere Css-Dateien für die Gestaltung der Web Anwendung zentral sind da­bei die Dokumente „MainStyle.css“, „TraditionalPieceStyle.css“ und „ModernPieceStyle.css“. Das Dokument „MainStyle.css“ ist für die grundlegenden Style-Elemente zuständig, die auf allen Web­seiten eingesetzt werden. Die anderen beiden Style-sheets sind nur für die Darstellung der Spiel­steine auf der Webseite „Instructions.jsp“ und auf der Webseite “board.jsp“ zuständig.

Zu­dem gibt es noch für die einzelnen Webseiten spezielle Style-sheets wie „BoardStyle.css“ oder „In­structionsStyle.css“. Diese haben spezielle Eigenschaften, die für die einzelnen Webseiten wich­tig sind.