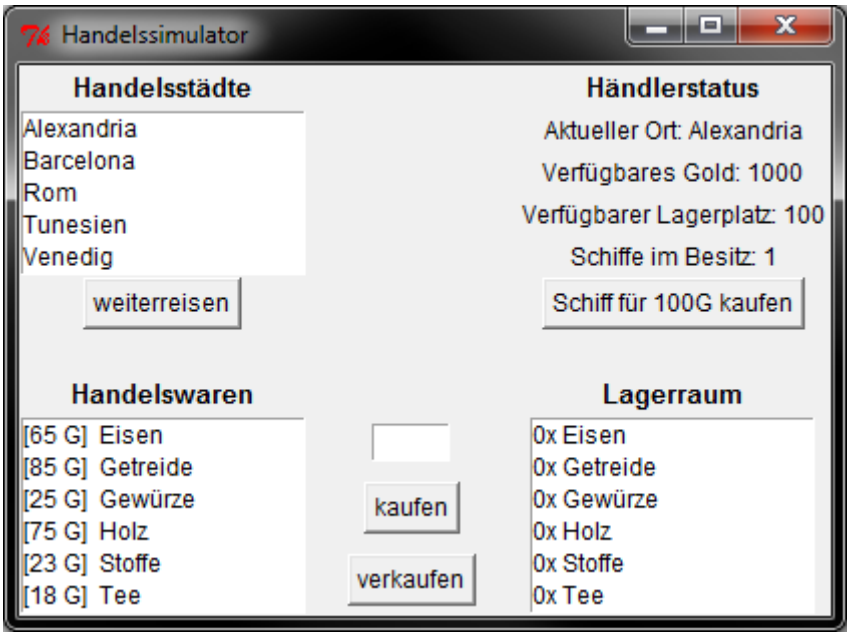
Handelssimulator

Zum Abschluss folgt nun ein etwas größeres Projekt. Bei dem (historischem) Handelssimulator handel es sich um ein recht simples Simulationsspiel. Der Spieler schlüpft in die Rolle eines Händlers und kann zwischen verschiedenen Städten hin und her reisen. Innerhalb der Städte kann der Spieler dann verschiedene Waren ein- und verkaufen.



Aufgabe 1

Als Grundlage für dieses Programm erhälst du eine Vorlage mit MVC Entwurfsmuster. Schau dir daher den Quellcode genau an, bis du den Ablauf des Programms nachvollzogen hast. Die folgenden Übersichten zeigen dir dabei, welche Attribute und Methoden die Klassen besitzen.

Hinweis: Hinter jedem Doppelpunkt steht immer der Datentyp des Rückgabewerte oder des Attributes bzw. Parameters. Void steht dabei für eine Methode ohne Rückgabewert.

Handelssimulator_Controller	
`__model : Handelssimulator_Model`	Beinhaltet ein Objekt der Klasse *Handelssimulator_Model*.
`__view : Handelssimulator_View`	Beinhaltet ein Objekt der Klasse *Handelssimulator_View*.
`__init__() : Void`	Mit dieser Methode wird der Controller initialisiert, das Programm aber noch nicht gestartet.
`kaufen() : Void`	Diese Methode wird ausgeführt, sobald man im View auf den Button *kaufen* klickt.
`reisen() : Void`	Diese Methode wird ausgeführt, sobald man im View auf den Button *weiterreisen* klickt.
`schiff_kaufen() : Void`	Diese Methode wird ausgeführt, sobald man im View auf den Button *Schiff kaufen* klickt.
`starte() : Void`	Mit dieser Methode kann das Programm gestartet werden.
`verkaufen() : Void`	Diese Methode wird ausgeführt, sobald man im View auf den Button *verkaufen* klickt.

Handelssimulator_Model	
__aktuelle_stadt : String	Beinhaltet den Namen einer Stadt und legt damit den aktuellen Standort des Spielers fest.
__anzahl_schiffe : Integer	Beinhaltet die Anzahl der aktuellen Schiffe im Besitz des Spielers.
__gold : Integer	Beinhaltet die Anzahl an Gold im Besitz des Spielers.
__lager : Liste	Beinhaltet eine Liste mit den einzelnen Mengen der Handelswaren im Besitz des Spielers
__lager_pro_schiff : Integer	Beinhaltet die Lagerkapazität eines Schiffes bzw. legt diese fest.
__max_preis : Integer	Beinhaltet den Maximalpreis einer Handelsware bzw. legt diese fest.
__min_preis : Integer	Beinhaltet den Minimalpreis einer Handelsware bzw. legt diese fest.
__preis_pro_schiff : Integer	Beinhaltet den Kaufpreis eines Schiffes bzw. legt diese fest.
__preise : Liste	Beinhaltet eine Liste mit den einzelnen Preisen der Handelswaren für alle Städte.
__staedte : Liste	Beinhaltet eine Liste mit den Namen der einzelnen Städte, welche vom Spieler bereist werden können.
__waren : Liste	Beinhaltet eine Liste mit den Namen der Handelswaren, welche vom Spieler ein- und verkauft werden können.
berechne_lagerplatz() : Integer	Mit dieser Methode kann der freie Lagerplatz berechnet werden. Rückgabewert ist der freie Lagerplatz als Integer.
berechne_neue_preise() : Void	Mit dieser Methode kann die Preisschwankung berechnet werden.
gib_aktuelle_stadt() : String	Gibt den Wert der Variable __ aktuelle_stadt zurück.
gib_anzahl_schiffe() : Integer	Gibt den Wert der Variable __ anzahl_schiffe zurück.
gib_gold() : Integer	Gibt den Wert der Variable __gold zurück.
gib_lager() : Liste	Gibt den Wert der Variable __lager zurück.
gib_preise() : Liste	Gibt den Wert der Variable __preise zurück.
gib_staedte() : Liste	Gibt den Wert der Variable __staedte zurück.
gib_waren() : Liste	Gibt den Wert der Variable __waren zurück.
setze_aktuelle_stadt(stadt : String) : Void	Setzt den Wert der Variable __aktuelle_stadt.
setze_gold(gold : Integer) : Void	Setzt den Wert der Variable __gold.
setze_lager(lager : Liste) : Void	Setzt den Wert der Variable __lager.

Handelssimulator_View	
__fenster : Tk	Beinhaltet ein Objekt der Klasse Tk. Ist damit ein Fenster der grafischen Oberfläche.
__gold_label : Label	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Verfügbares Gold:“.
__handelswaren_auswahl : Listbox	Beinhaltet ein Auswahlfeld und ist ein Element auf dem Fenster. Besitzt als Auswahlelemente die einzelnen Handelswaren inklusiver ihres Preises.

`__handelswaren_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Handelswaren“.
`__kaufen_button : Button`	Beinhaltet einen Button und ist ein Element auf dem Fenster. Beim Betätigen des Buttons soll eine Handelsware gekauft werden.
`__kaufen_methode : Function`	Beinhaltet einen Verweis auf eine Methode der Klasse *Handelssimulator_Controller*.
`__lager_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Verfügbarer Lagerplatz:“.
`__lagerraum_auswahl : Listbox`	Beinhaltet ein Auswahlfeld und ist ein Element auf dem Fenster. Besitzt als Auswahlelemente die einzelnen Handelswaren inklusiver ihrer Anzahl im Lager.
`__lagerraum_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Lagerraum“.
`__menge : Entry`	Beinhaltet ein Eingabefeld und ist ein Element auf dem Fenster. Auf den Wert des Eingabefeldes wird beim Ein- und Verkauf zugegriffen.
`__ort_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Aktueller Ort:“.
`__platzhalter : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Das Label besitzt einen Inhalt und dient nur als Platzhalter.
`__platzhalter2 : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Das Label besitzt einen Inhalt und dient nur als Platzhalter.
`__reisen_button : Button`	Beinhaltet einen Button und ist ein Element auf dem Fenster. Beim Betätigen des Buttons soll zu einer neuen Handelsstadt gereist werden.
`__reisen_methode : Function`	Beinhaltet einen Verweis auf eine Methode der Klasse *Handelssimulator_Controller*.
`__schiff_button : Button`	Beinhaltet einen Button und ist ein Element auf dem Fenster. Beim Betätigen des Buttons soll ein weiteres Schiff gekauft werden.
`__schiff_kaufen_methode : Function`	Beinhaltet einen Verweis auf eine Methode der Klasse *Handelssimulator_Controller*.
`__schiffe_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Schiffe im Besitz:“.
`__staedte_auswahl : Listbox`	Beinhaltet ein Auswahlfeld und ist ein Element auf dem Fenster. Besitzt als Auswahlelemente die einzelnen Handelsstädte.
`__staedte_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Handelsstädte“.
`__status_label : Label`	Beinhaltet ein Label und ist ein Element auf dem Fenster. Besitzt den Text „Händlerstatus“.
`__verkaufen_button : Button`	Beinhaltet einen Button und ist ein Element auf dem Fenster. Beim Betätigen des Buttons soll eine Handelsware verkauft werden.
`__verkaufen_methode : Function`	Beinhaltet einen Verweis auf eine Methode der Klasse *Handelssimulator_Controller*.
`__init__() : Void`	In dieser Methode werden Variablen initialisiert und die GUI des Programms zusammengebaut.
`aktualisiere_handelswaren_auswahl(waren : Liste, preise : Liste) : Void`	Mittels dieser Methode soll die Auswahl der Handelswaren in der GUI aktualisiert werden.

`aktualisiere_lagerraum_auswahl(waren : Liste, lager : Liste) : void`	Mittels dieser Methode soll die Auswahl des Lagerraumes in der GUI aktualisiert werden.
`aktualisiere_staedte_auswahl(staedte : Liste) : Void`	Mittels dieser Methode soll die Auswahl der Städte in der GUI aktualisiert werden.
`aktualisiere_status(aktuelle_stadt : String, gold : Integer, lagerplatz : Integer, anzahl_schiffe : Integer) : void`	Mittels dieser Methode soll der Status-Bereich der GUI aktualisiert werden.
`gib_ausgewaehlte_handelsware() : Liste`	Gibt das ausgewählte Element in __handelswaren_auswahl zurück.
`gib_ausgewaehlte_lagerware() : Liste`	Gibt das ausgewählte Element in __lagerraum_auswahl zurück.
`gib_ausgewaehlte_stadt() : Liste`	Gibt das ausgewählte Element in __staedte_auswahl zurück.
`gib_eingabe_menge : String`	Gibt die Eingabe in __menge zurück.
`kaufen() : Void`	Diese Methode wird ausgeführt, sobald man auf den Button *kaufen* klickt.
`reisen() : Void`	Diese Methode wird ausgeführt, sobald man auf den Button *weiterreisen* klickt.
`schiff_kaufen() : Void`	Diese Methode wird ausgeführt, sobald man auf den Button *Schiff kaufen* klickt.
`setze_kaufen_methode(methode : Function) : Void`	Setzt den Wert der Variable __kaufen_methode.
`setze_schiff_kaufen_methode(methode : Function) : Void`	Setzt den Wert der Variable __schiff_kaufen_methode.
`setze_reisen_methode(methode : Function) : Void`	Setzt den Wert der Variable __reisen_methode.
`setze_verkaufen_methode(methode : Function) : Void`	Setzt den Wert der Variable __verkaufen_methode.
`starte() : Void`	Mit dieser Methode kann die GUI gestartet werden.
`verkaufen() : Void`	Diese Methode wird ausgeführt, sobald man auf den Button *verkaufen* klickt.
`zeige_nachricht(title : String, nachricht : String) : Void`	Mit diese Methode kann eine Nachrichtenbox (für Fehler, Informationen etc.) angezeigt werden.

Aufgabe 2

Beschreibe den Ablauf beim …

a) Starten des Spiel bis zu Anzeige der grafischen Oberfläche.

b) Kauf einer Handelsware. (Beginn bei Drücken des Buttons.)

Erläutere dabei, auf welche Klassen und jeweilige Methoden zugegriffen wird. Nutze im Idealfall ein Sequenzdiagramm zur Beschreibung des Ablaufs.

Aufgabe 3

In der Vorlage sind noch nicht alle Funktionen (Waren aus dem Lagerraum verkaufen; ein weiteres Schiff kaufen) implementiert worden. Füge diese selber in die Vorlage ein und teste sie anschließend. (Du musst dafür lediglich in der Controller-Klasse arbeiten.)

Starten kannst du das Programm von der Datei *Handelssimulator.py* aus.

Aufgabe 4

Nimm dir aus der folgenden Liste weitere Features (Schwierigkeit nimmt absteigend zu) für das Spiel heraus und implementiere diese in das Programm:

- Ein Spielende nach einer bestimtmen Anzahl an Spielrunden mit der Berechnung einer Punktzahl.
- Möglichkeit während der Reise von Piraten, Unwettern etc. Überrascht zu werden und einen Teil der Schiffsladung zu verlieren.
- Preise schwanken nicht global gleichmäßig, sondern für jede Stadt werden die Preise je Ware einzelnt bestimmt. (Jede Stadt besitzt also eine eigene Preisliste für die Waren, welche sich bei jedem Weitereisen verändert.)
- Möglichkeit des Eintritts eines besonderen Ereignisses (z.B. Naturkatastrophen, Kriege etc.), die den Preis einer Ware in allen Städten für mehrere Spielrunden stark an- oder absteigen lassen.
- Einen Computergegner, der ebenfalls verschiedene Städte bereist und Waren ein- und verkauft. Am Ende sollten dessen Punktzahlen mit der des Spielers verglichen werden.
- Gestalte eine neue grafische Oberfläche mit mehr Bildern und Fenstern.