Strukturdefinition „PaperboyClone“

Projekt PaperboyClone im Fach "Pattern and Frameworks" an der FH Brandenburg der Studenten Martin Freudenberg und Timo Kramer

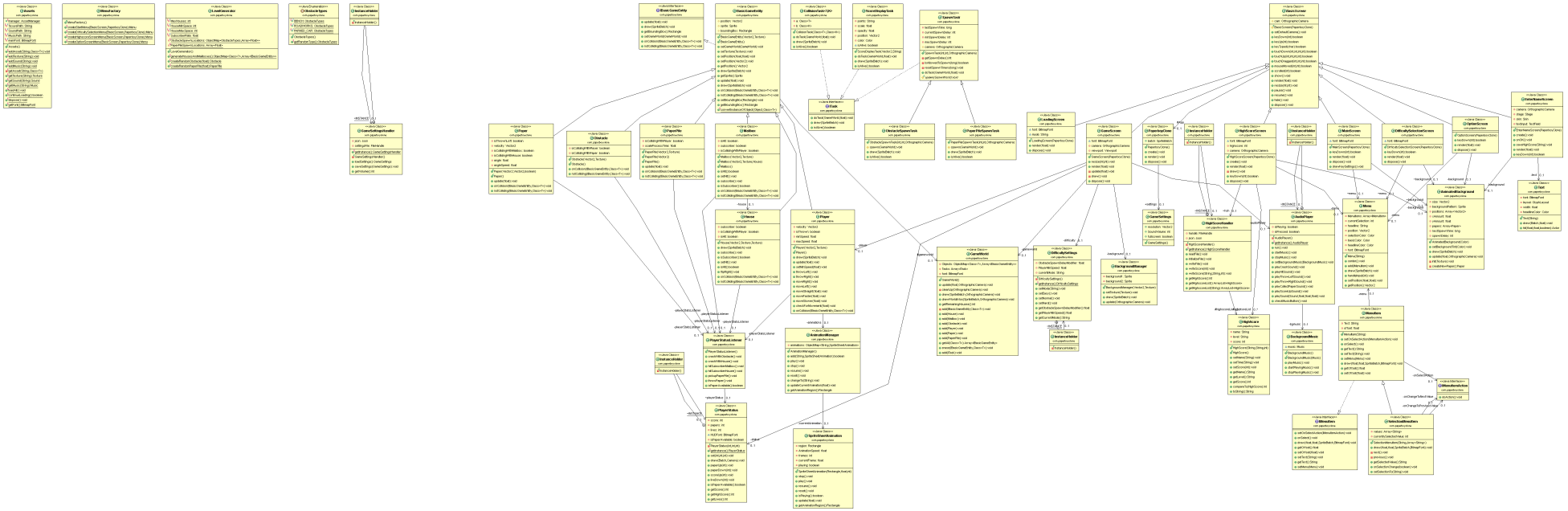
Vorwort

Auf den folgenden Seiten finden sie UML Klassendiagrame zum Projekt und im speziellen den Einsatz verschiedener Design Patterns mit kurzer Erläuterung.

Da die Darstellung von größeren Grafiken in Textverarbeitungsprogrammen nicht optimal ist,
finden sie außerdem die Diagramme einzeln als Bild-Datei im Ordner "Diagramme" .

Klassendiagramm komplett:

siehe Klassendiagramm.gif

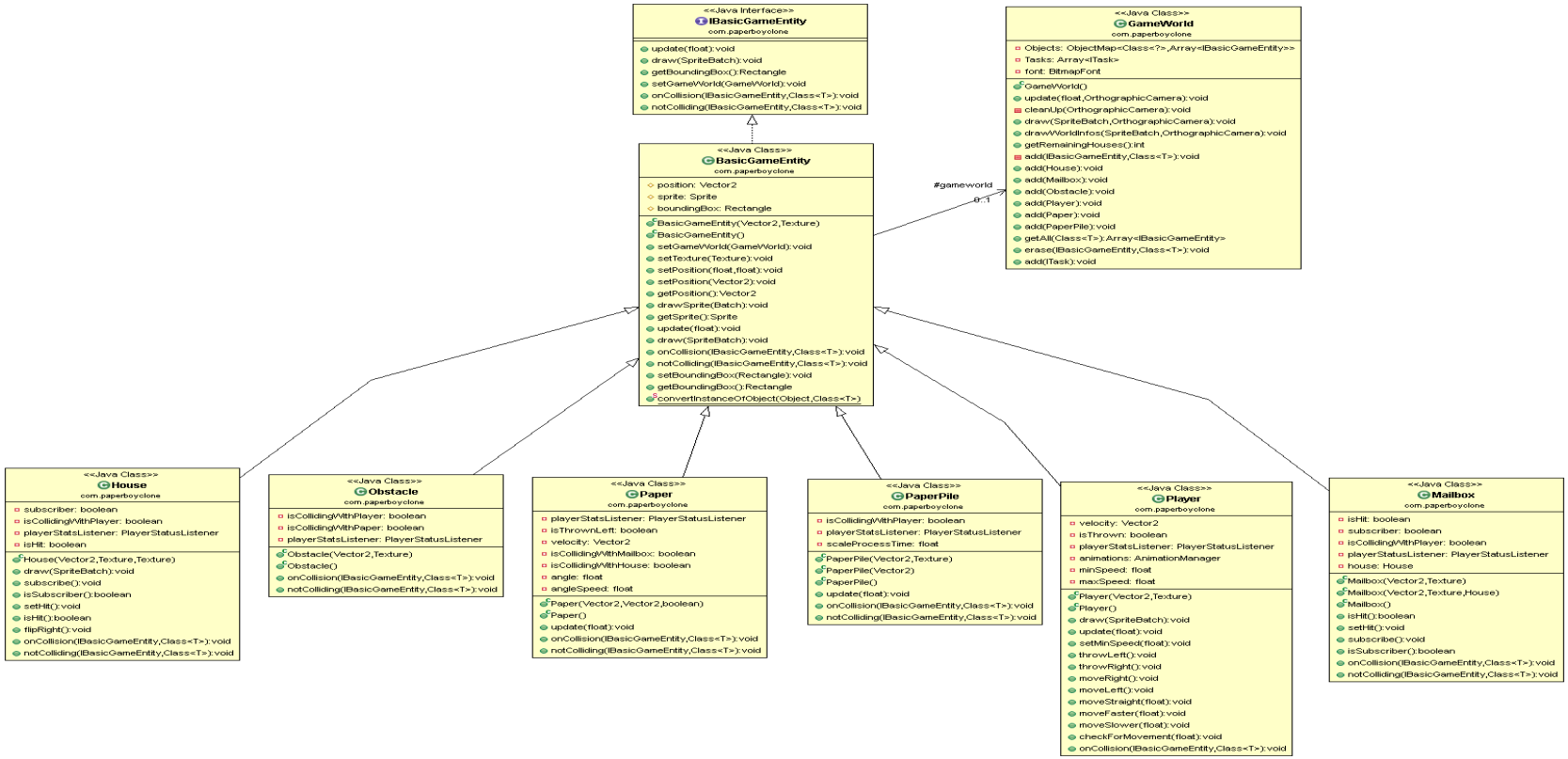


Game Entity Vererbungshierachie und Verwendung in der GameWorld Klasse

Alle Entities implementieren das IBasicGameEntity Interface um so in der GameWorld Klasse einheitlich behandelt werden zu können. Um Coderedundanz zu reduzieren erben alle von der BasicGameEntity Klasse die das grundsätzliche Verhalten der Enities beschreibt. Durch weitere Vererbung erhalten die konkreten Klassen mittels überschreiben der draw() und update() Methode ihr eigenes Verhalten.

In der GameWorld Klasse liegen die Entities in einer Map geordnet nach Klassentyp. Angesprochen werden sie nur über die Schnittstellen des IBasicGameEntity Interface innerhalb der GameWorld Klasse. In der update() und draw() Methode der GameWorld Klasse wird durch die Map iteriert und die jeweilige update() bzw. draw() Methode der Entity aufgerufen.

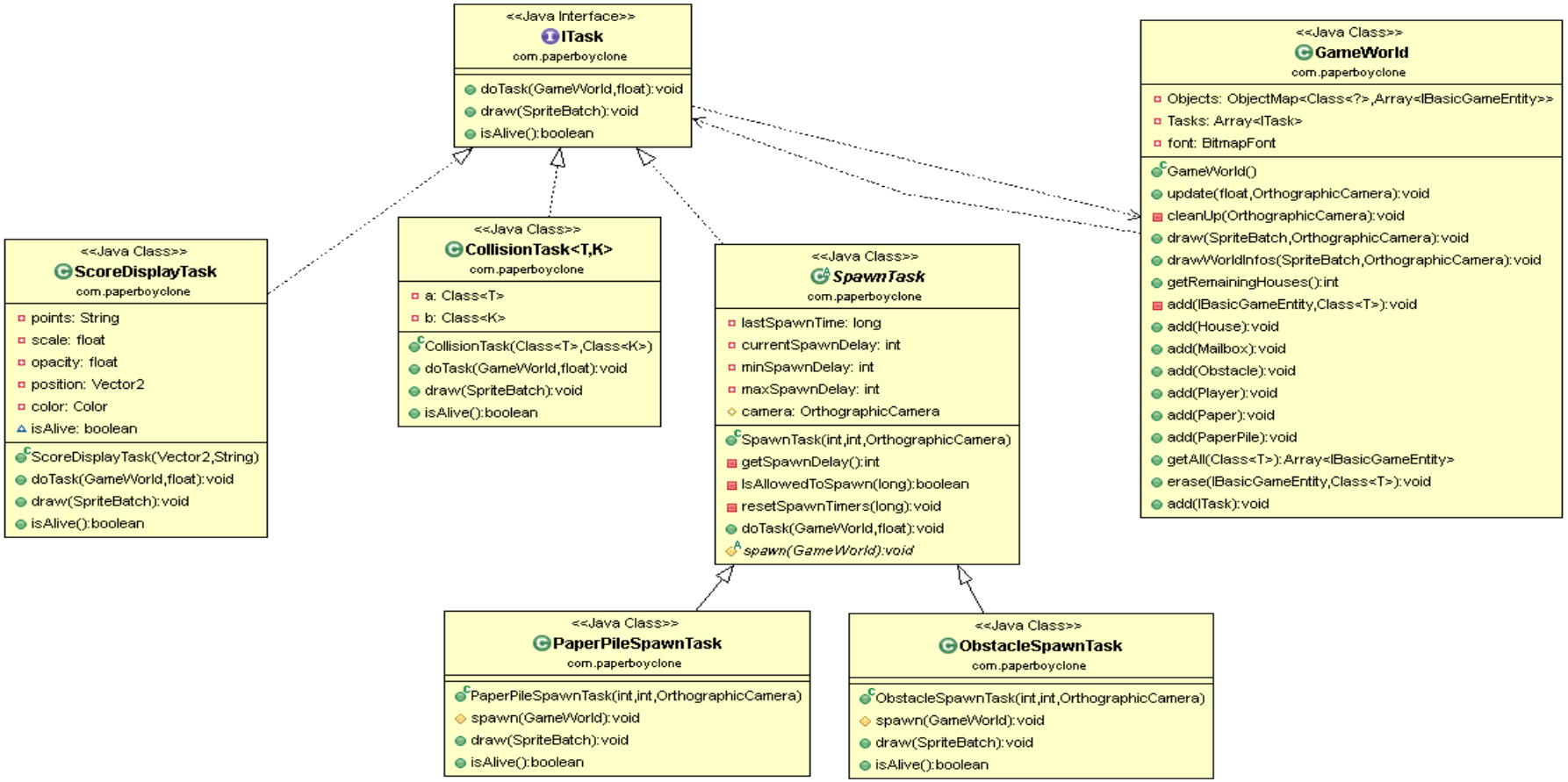
Siehe BasicGameEntityHierarchy.gif



Task System, Strategy Pattern

Um verschiedene Aktionen auszuführen die während des gesamten Spielablaufs regelmäßig oder nach bestimmten Ereignissen durchgeführt werden müssen haben wir ein Task System implementiert. Die Tasks implementieren alle das ITask Interface und definieren ihr eigenes Verhalten in der doTask() bzw. draw() Methode. Zur Laufzeit können Tasks hinzugefügt oder entfernt werden. Falls ein Task fertig mit seiner Ausführung ist kann er dies mittels des Rückgabewerts der isAlive() Methode anzeigen, daraufhin wird die GameWorld Klasse ihn aus der Liste der aktiven Tasks löschen. Bspw. wird ein ScoreDisplayTask erzeugt wenn der Spieler einen Briefkasten trifft und ist so lang aktiv wie die Punktezahl die über dem Briefkasten erscheint zu sehen ist. Aufgabe dieses Tasks ist es die Farbe, Skalierung und den Transparenzwert der Punkte Anzeige zu verändern.

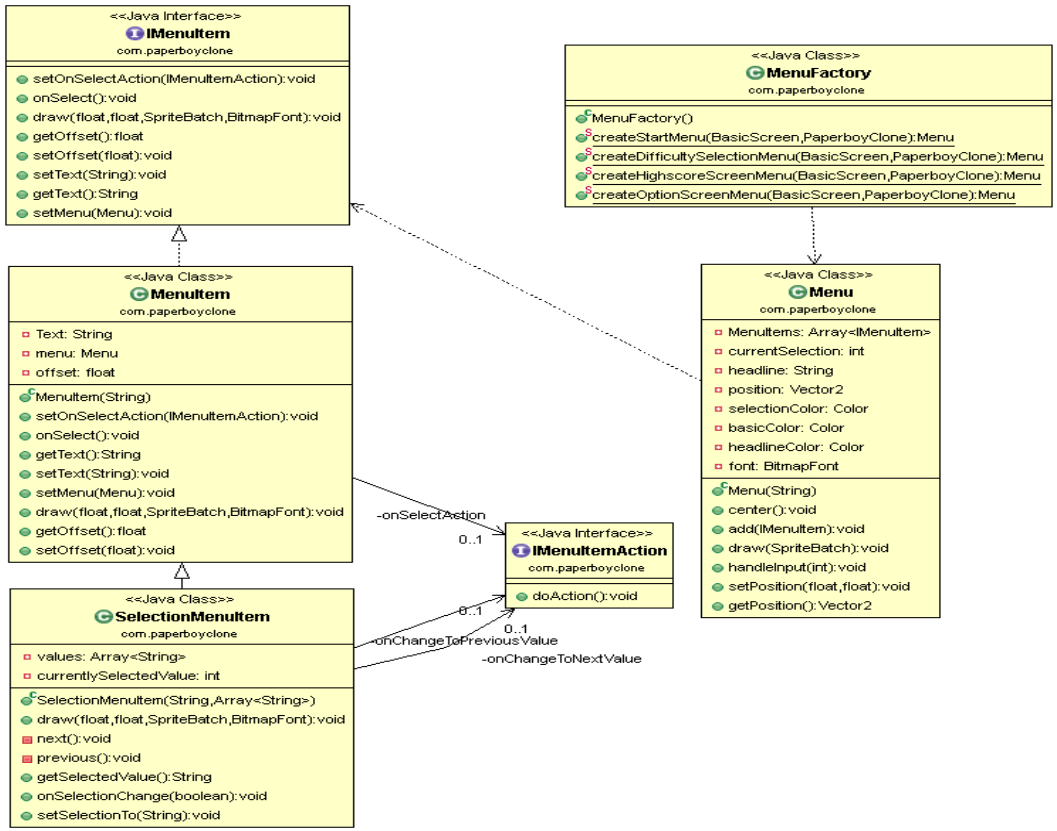
Siehe TaskSystemStrategyPattern.gif



Menu, MenuItems, Command Pattern, Factory

Mittels des Command Pattern hab wir die Auslöser bei selektieren eines Menü Punktes implementiert. Jedem MenuItem kann ein anonym implementiertes IMenuAction Interface übergeben werden. Die Implementiereung der doAction() Methode bestimmt dann was bei Auswahl eines Menü Punktes passiert. Aufgerufen wird von der Menu Klasse welche alle MenuItems in einem Array speichert und überwacht welcher Menü Punkt gerade angewählt ist, die onSelect() Methode sobald der Spieler Enter drückt. Daraufhin ruft das MenuItem die doAction() Methode des anonymen Interfaces auf. Menus werden über die MenuFactory komplett erzeugt und über die verschiedenen statischen Methoden zurückgegeben.

Siehe MenuCommandPattern.gif



Singleton Beispiel

Mehrere Klassen wurden als Singleton implementiert und sind für kleinere Aufgaben zuständig, überwiegend um Daten abrufbar zu halten. Alle Singleton Klassen verfügen über einen sogenannten InstanceHolder welcher sicherstellt das zur Laufzeit immer nur eine Instanz dieser Klasse erzeugt werden kann. Über getInstance() wird diese abgerufen.

Siehe ExampleSingletonPattern.gif