Gruppe 21 – Bomberman Island

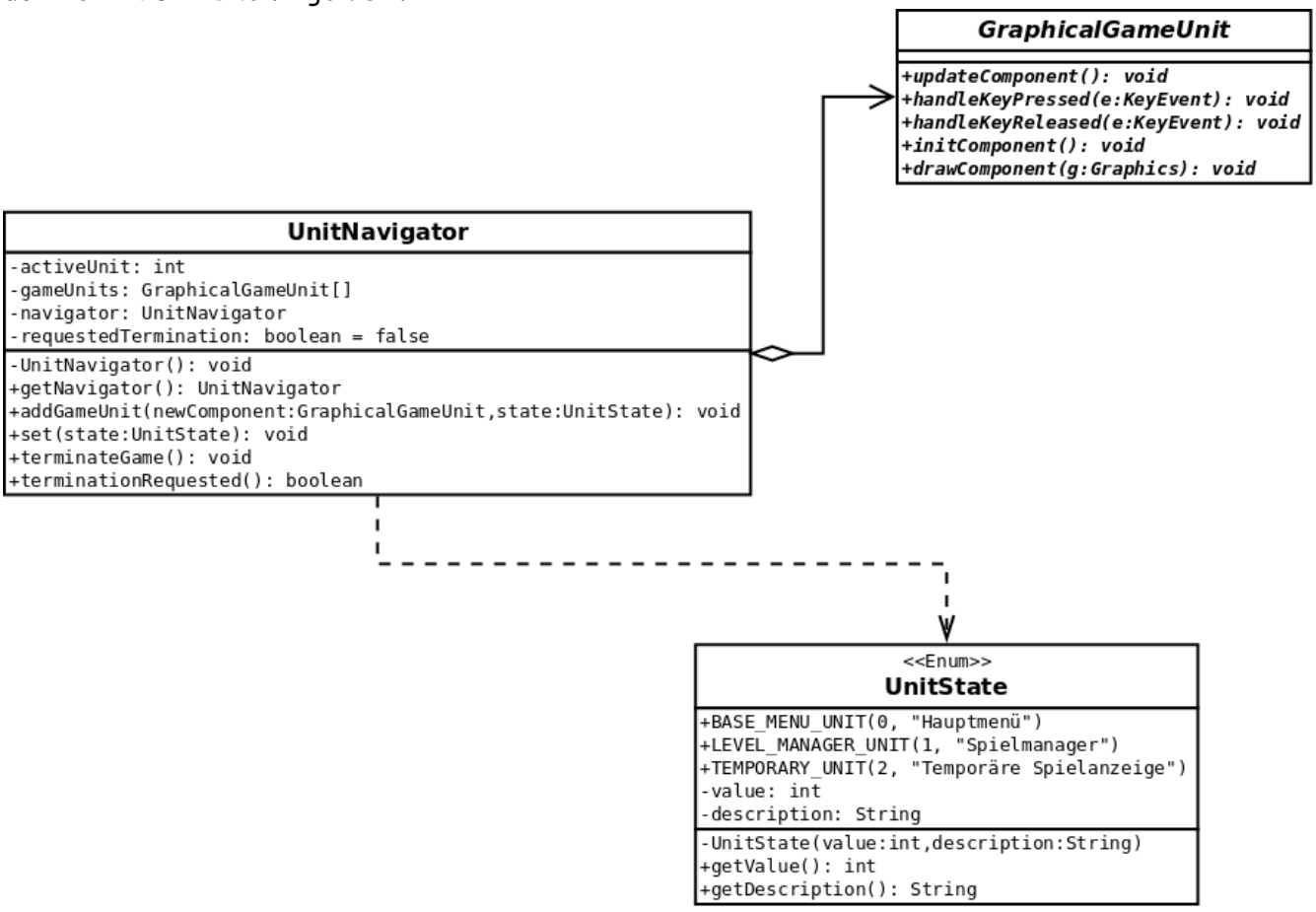
# Aufbau und Projektstruktur I – Framework und Navigation

**Details: Spielmodule, Darstellung und Navigation (main Package)**

Im Rahmen unseres Projekts nutzen wir lediglich zwei `javax.swing` Komponenten:

- Ein `javax.swing.JPanel` (`main.MainPanel.java`), das wir wie eine Leinwand zum Zeichnen aller Spielinhalte verwenden. Hier wird User Input in Form von `java.awt.KeyEvents` entgegen genommen und weitergeleitet; hier findet sich der Hauptthread, der in möglichst regelmäßigen Intervallen interne Daten aktualisiert und neu darstellt
- Ein umgebendes `javax.swing.JFrame`, welches sich als Instanzvariablen der Klasse `main.BMGame` wiederfinden lässt

Um dennoch unterschiedliche Spielkomponenten zu unterscheiden und zwischen diesen zu navigieren, nutzen wir die abstrakte Klasse `main.GraphicalGameUnit`, das Enum `main.UnitState` und den `main.UnitNavigator`:



Eine Subklasse der `GraphicalGameUnit` stellt ein Modul im Programm dar, welches individuell entscheiden kann, für welche `java.awt.KeyEvents` es sich interessiert, wie auf einen Update Aufruf reagiert werden soll und was bei einem Aufruf von `drawComponent()` geschieht.

Der `UnitNavigator` wurde unter Verwendung des Singleton Entwurfsmusters realisiert, um mehrere Instanzen dieser Klasse innerhalb des Programms zu vermeiden und um ihn in jedem Modul bequem zu erreichen. Dabei verwaltet der `UnitNavigator` zum einen maximal drei verschiedene Spielmodule in einem Array. Diese werden in die folgenden Kategorien eingeteilt:

- BASE_MENU_UNIT – Das Hauptmenü (als häufig verwendetes Modul nach der Erstellung fest eingespeichert)
- LEVEL_MANAGER_UNIT – Module in denen der eigentliche Spielablauf geregelt wird
  - `LevelManagerUnit` (Einzelspieler)
  - `LocalMultiplayerUnit` (Zwei Spieler an einer Tastatur)
  - `MultiplayerUnit` (Mehrspieler Netzwerkmodus)
- TEMPORARY_UNIT – Spielmodule für Übergänge (`TransitionUnit`), Untermenüs oder weniger wichtige Teilkomponenten wie die `WorldMapUnit` und die `MPLoungeUnit` (Multiplayer Lounge)

Das Enum `UnitState` definiert zu diesem Zweck verschiedenen Spielzustände. Zu einem `UnitState` gehört dabei eine eindeutige Zahl (wird zur Einordnung einer `GraphicalGameUnit` in das `gameUnits` Array des `UnitNavigators` verwendet) und eine Beschreibung. Die Definition weiterer Zustände wird durch den privaten Konstruktor unterbunden.

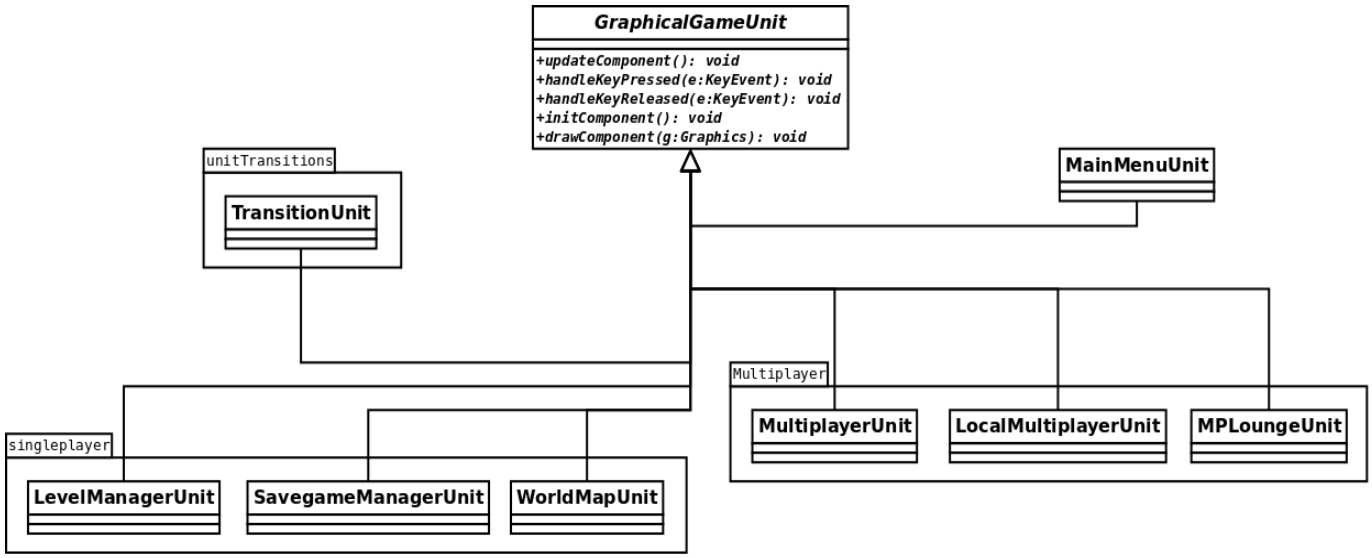
Die zweite Aufgabe des `UnitNavigators` liegt in der Navigation zwischen den verschiedenen Spielmodulen. Ein typischer Aufruf sieht dann in etwa so aus:

```
UnitNavigator.getNavigator().addGameUnit(
          new SavegameManagerUnit(true),
          UnitState.TEMPORARY_UNIT);
UnitNavigator.getNavigator().set(UnitState.TEMPORARY_UNIT);
```

Es wird ein Spielmodul zum Speichern von Spielständen erstellt und zu dem Array der zurzeit bekannten Module als temporäre Unit hinzugefügt (`addGameUnit`). Danach wird der Spielzustand TEMPORARY_UNIT als aktiver Zustand ausgewählt.

Innerhalb des `MainPanel` läuft eine Art Endlos-Schleife, welche in regelmäßigen Abständen die aktive Unit vom UnitNavigator erfragt, diese aktualisiert (`updateComponent`()) und unter Übergabe des eigenen Graphics Objektes neu zeichnet. Dabei wird geprüft, ob ein Modul um die Terminierung des Spiels erbeten hat. In diesem Fall wird die Schleife verlassen und durch `System.exit(0)` das Programm beendet. Die über einen KeyListener in der Klasse `MainPanel` empfangenen `KeyEvents` werden ebenfalls an die aktive `GraphicalGameUnit` weitergeleitet.

**Überblick über die wichtigsten Module**




- **TransitionUnit** – eine flexible Unit zur Darstellung von Übergängen („Zoom“-Effekt bei Map-Beendigung), Info-Nachrichten („You win“ / „You lose“), Hilfenachrichten (z.B. nach Druck von F1 im Einzelspielermodus/WorldMap), etc.
- **LevelManagerUnit** – arbeitet mit einem Kampagnen Objekt, analysiert Map-Beendigung (Aufruf einer TransitionUnit), fordert bei Bedarf eine neue Map an, verwaltet einen ImageLoader (indem die Grafiken eines Levels gespeichert werden), aktualisiert Map-Objekt und leitet Benutzereingaben an ein Player Objekt weiter. Hier finden sich auch die Methoden zur Positionierung der Map in Abhängigkeit der Spielerposition („Insight View“)
- **SavegameManagerUnit** – Modul zum Speichern und Laden von Spielzuständen (Levelfortschritt, aufgesammelte Upgrades)
- **WorldMapUnit** – stellt ein WorldMap Objekt dar, setzt Variablen für den ausgewählten Level nach Benutzereingabe
- **MultiplayerUnit** – Mehrspielerpendant zur LevelManagerUnit: Wandelt Spielereignisse (Spielertod, Upgradespawn) und Benutzereingaben in Nachrichten um, die über das Netzwerk versand werden. Bietet Methoden um eingehende Nachrichten zu analysieren und deren Inhalt auszuführen
- **MPLoungeUnit** – Spielmodul zu dem man nach einem Serverbeitritt gelangt. Stellt Informationen über den Status anderer Spieler dar (verbunden/bereit/nicht verfügbar) und erlaubt dem Host das Starten des Multiplayermatches

Weitere Informationen zu den Methoden und Aufgaben der einzelnen Module finden sich im Javadoc.