Implementation for TwitterNethack
Assignment in the course PA1415 Programvarudesign <2017-04-26>
Author: Morgan Lexander, Henrik Nilsson, Magnus Nyqvist och Victor Olsson.

Author Name	Social Security Number	Thinking	Writing
Morgan L.	971107-xxxx	25%	25%
Henrik N.	970416-xxxx	25%	25%
Magnus N.	940429-xxxx	25%	25%
Victor O.	970612-xxxx	25%	25%

System description

The system is going to be based around the classic game NetHack but in 2D graphics. NetHack is a sort of adventure and roleplaying game where you create a character and goes through random generated dungeons. In the dungeons you can meet various different monsters and collect items. The final goal is to find the amulet of Yendor and escape with it alive.

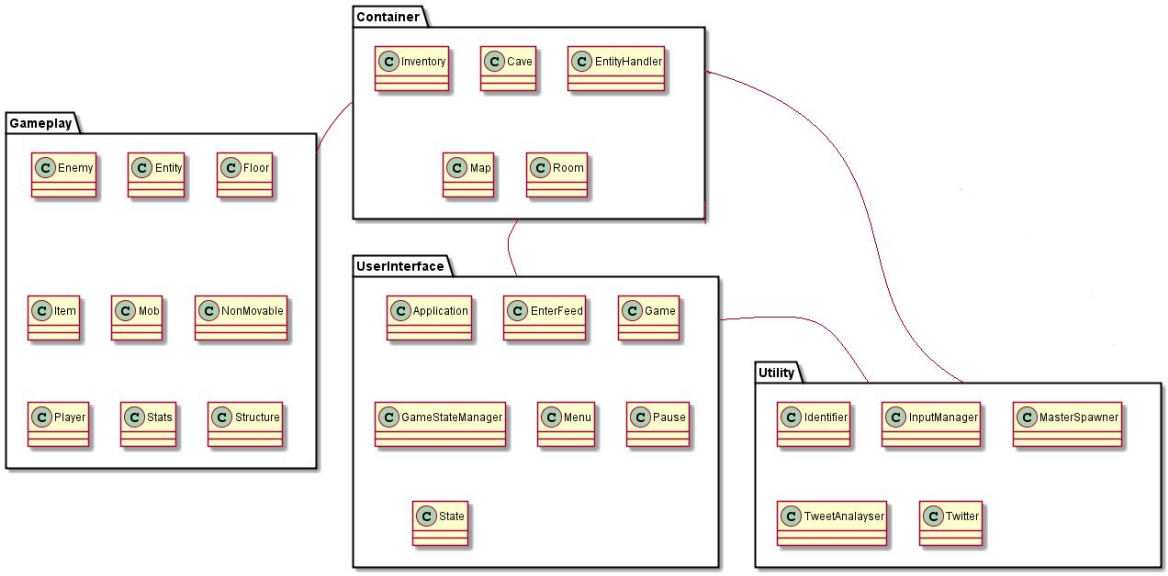
Nethack is a very advanced game and because of this and the limited time we are given, our version of NetHack is not going to include all of the features that the original have. The functionalities we are aiming to implement:

- An opening screen where you can start a new game or join an existing one.
- A screen where you can setup the character.
- Monsters and creatures that freely move around the dungeons.
- Different items to pick up and drop.
- Multiplayer support.
- Generate dungeons with help of twitter.

The dungeons in our NetHack version is going to be generated by the Twitter API. Its job is to read certain keywords in Twitter posts and use these to create specific types of monsters and items for every room in the game. When you enter a new cave, your system shall briefly display information about the “owner” of the cave.

Class and Package Diagram
Present the final UML Class Diagram (divided into Packages).
Briefly describe each package and its responsibilities.

Package Diagram



The UserInterface package is the presentation of the application. This is where the game window is located. When the user is using this application, this is where all the interacting is taking place.

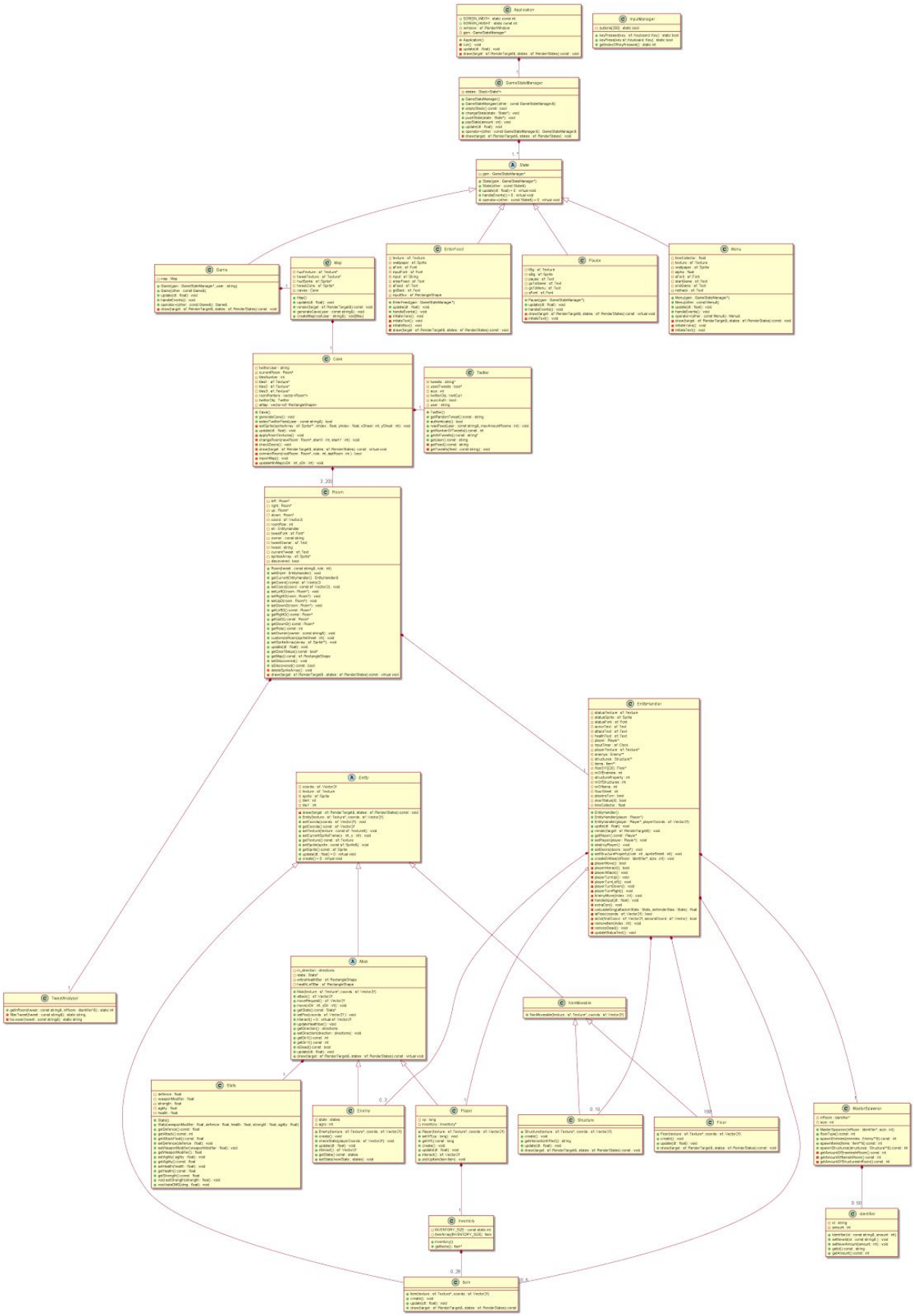
The Container package is where all the game data is stored, like the caves and each caves' rooms. We also find the EntityHandler which as the name implies, handles entites . An Inventory is also included as a container because it stores items the player picks up!

The Utility package is a collection of classes that the container package makes use of. For example the CollisionHandler, which returns whether there was a collision between two objects. The selling point for this game is the Twitter integration. We use Twitter as an utility to help the game generate rooms based on tweets, gathered from Twitter.

The Gameplay package is containing all the gameplay elements you find in the game, like the enemies, the player, floors and structures.

Differences from our pre-implementation package diagram:
There is no huge difference in the implementation compared with our pre-implementation package diagram. The only thing which is different is that the Userinterface-package is now using classes from the Utility package. This is because the InputHandler is in the Utility-package and we needs input in the Userinterface-Package.

Class diagram



Differences from our pre-implementation class diagram:
Some of the classes, functions and variables has been renamed. This is because the new names makes more sense in the context of the design.
Some of the more noticeable changes which have been done are:

- CollisionHandler has been removed.
  This is because we designed the game with a grid system, in which we don't need to check if two textures are colliding. We can easily find and compare coordinates in our current system.
- Room is no longer holding Identifier objects. This is because in our current system, we instead let the MasterSpawner deal with the Identifiers. Room talks with TwitterAnalyzer and passes the Identifier-pointer to EntityHalndler which then do the magic with MasterSpawner.
- Cave do not use a 2D-array anymore to hold the rooms. We went with the approach that each room points towards one another like a linked list. Cave then holds a vector with pointers towards all the rooms to deal with the memory deallocation.
- We have decided not to use an arraylist to hold the caves. Instead we only have one cave and use the Room-pointers to create different “levels” in the cave. At the moment we only have a 2-dimensional cave. We will later implement a 3-dimensional room-system to simulate depth.
- We have renamed Client to GameStateManager and also added an Application class which has the game-loop. This is the heart of the system.
- During the implementation we noticed that our system had a lack of awesome features which will impress the customer. Therefore, when we were done with our first iteration we took the freedom to add stuff from our second iteration.