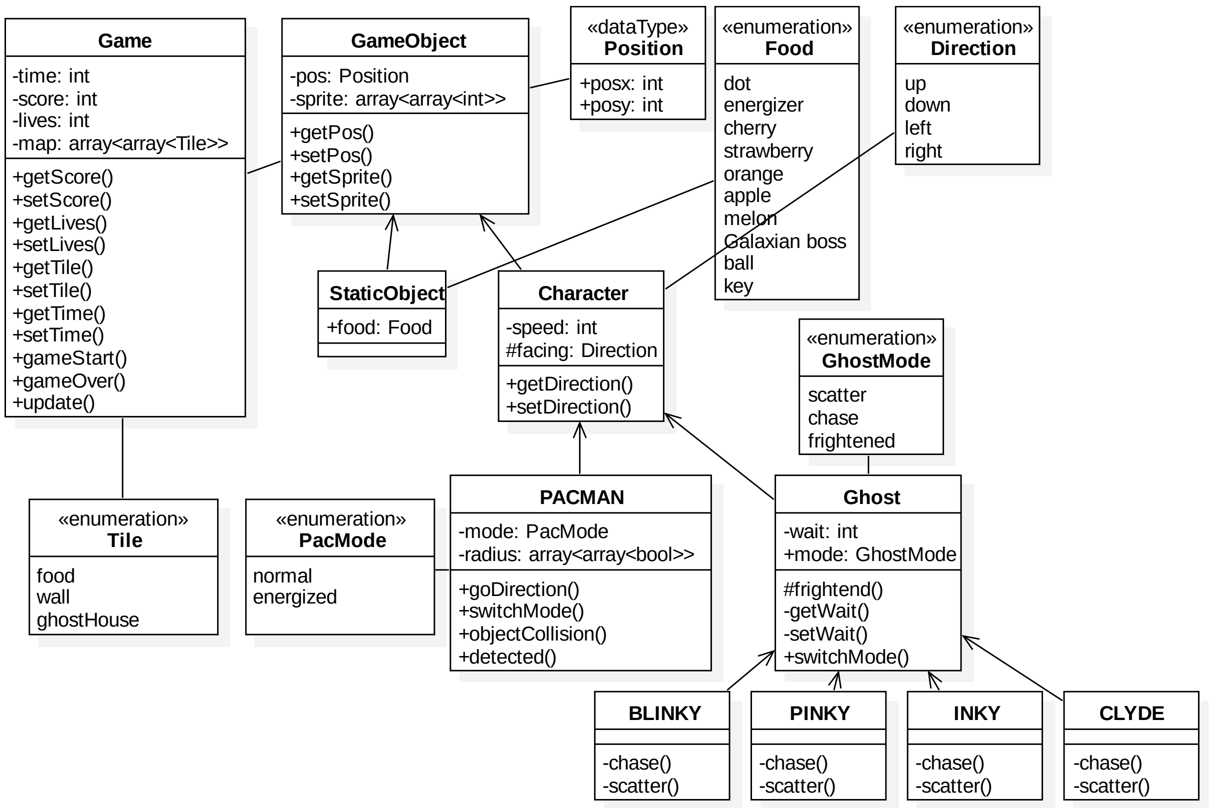
Alexander Green & Qiao Ren
Group 15
Assignment 5a: PAC-MAN

UML Diagram



Design decisions

The game draws and updates the field with `SetTile()` placing the static object food, walls, and the ghost house area.
PAC-MAN eating (items or ghosts) or getting killed is controlled by **`PACMAN`**`::objectCollision()`. It can be contained in **`PACMAN`**, because these events rely on PAC-MAN's location and facing direction. However, it is dependent on the mode of the ghost (frightened or not), which is why **`Ghost`**`::mode` is public.
The ghosts have three modes: starting with *scatter*, followed by *chase* once PAC-MAN comes within a certain radius of that ghost, and *frightened* once PAC-MAN is in *energized* mode. While frightened is globaled among the ghosts, scatter and chase are not. Therefore **`Ghost`**`::Mode mode` is an attribute of **`Ghost`**, not **`Game`**. To check whether PAC-MAN is within the detection radius of the ghost, it was chosen to implement a single radius in **`PACMAN`** instead of one for each ghost, cutting down the variables to update by four. The radius is implemented as an 2D boolean array the size of the maze map.
The attribute `wait` in **`Ghost`** keeps track of whether the ghost is alive or not. While `wait > 0`, the ghost will be stuck in the ghost house. The get and set functions for this attribute are private as they are set by `switchMode()`, which is public as it is called through **`PACMAN`**`::objectCollision()` and **`PACMAN`**`::detected()`. The function `frightened()` is protected as the instances of the separate ghost classes need access to it.
Since all four ghosts have unique scatter and chase patterns, they have their own classes with their specific pattern functions.
The movement of the characters is controlled by calling the `setPos()` function for each object through **`Maze`**`::update()`.