

## Chaps Challenge Plugins

**Levels as plugins** were implemented by packaging classes, assets (.png's) and json level descriptions into zips that can be dropped into the levels folder. The program had to be refactored in certain areas to the correct level of coupling and cohesion to remove dependencies and hard coding of Tile/Mob types.

The use of polymorphism and inheritance allowed me to use the parent classes of Mob and Tile to define the interactions the game would have with the new objects, while allowing the new classes to define their behaviour. Json frameworks, specifically Javax.json-1.1 were used to simplify the conversion of objects to and from json string.

**Methods** to create a tile from JSON or convert a tile to JSON had to be moved from the persistence package into the tiles themselves so that new tiles that are added in plugins can describe their own json strings. The same had to be done to Mob JSON methods.

**Level descriptions** are extracted from zips present in the /levels package and stored in the level manager.

**Classes** are loaded by a URL class loader that scans the zip file for files ending in .class. These classes are stored in a class set for easy access later and to prevent package naming confusion.

**Assets** are loaded from the zips in the /levels package and passed to the asset manager to be stored and scaled.

When a level is loaded, the level manager parses the JSON description retrieved from the zip.

**Standard elements** such as the time allowed for a level are read into the game object. More diverse objects such as tiles must be built from their class methods. The class needed is determined by reflection, allowing the use of classes from zips.

This system allows the addition of new types of Tiles and Mobs in level plugins. In our program, no mobs are present in level 1. Level 2 adds 3 types of mobs with unique behaviour, and the water tile which requires flippers to enter.

Zac Durant.