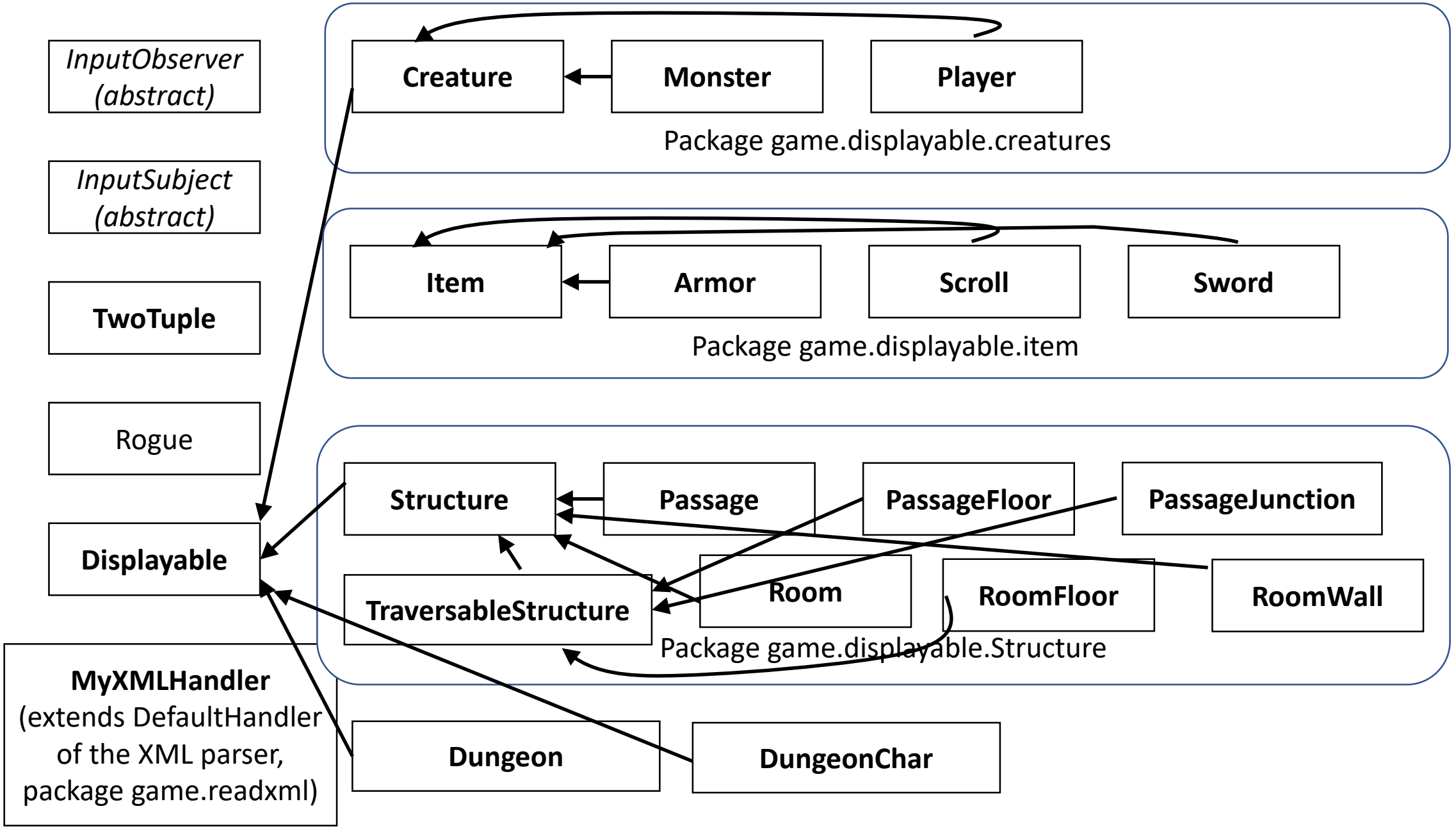
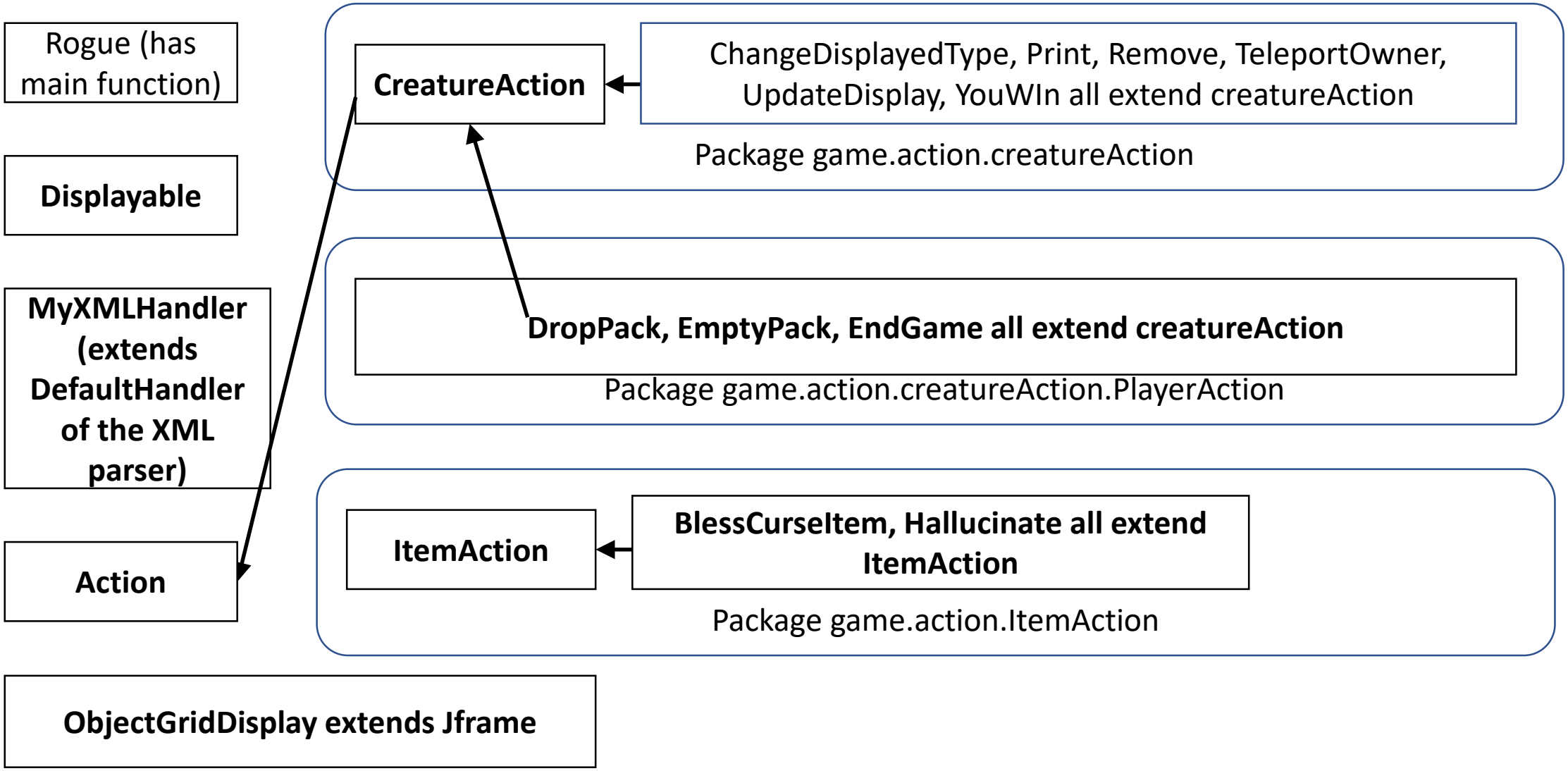
Description of Dungeon classes needed by the parser

Feel free to add extra methods or to change the structure of these. This is intended to be helpful, not to be requirements.

Overall structure in my dungeon ()



Overall structure in my dungeon ()



Dungeon
getDungeon(string name, int width, int gameHeight**)**
addRoom(Room)
addCreature(Creature)
addPassage(Passage passage)
addItem(Item)

ObjectDisplayGrid
getObjectDisplayGrid(int gameHeight, int width, int topHeight)
setTopMessageHeight(int topHeight**);**

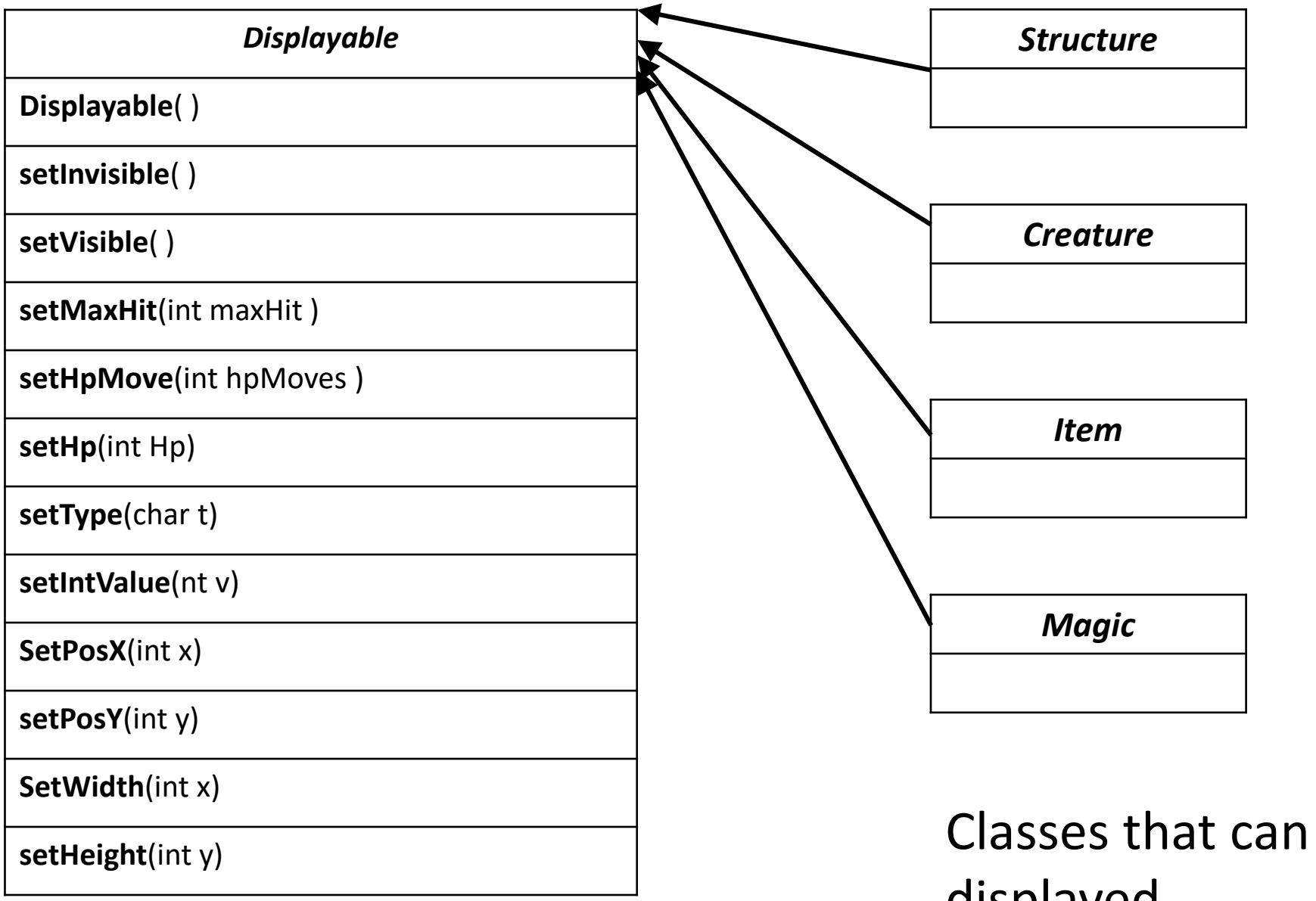
Stand alone classes (for now, at least)

Note that the getObjectDisplayGrid(it, int, int) is called to both to initially create an ObjectDisplayGrid, or to return a reference to the created ObjectDisplayGrid. This is done because there must only be one instance the logical display for the entire game. This is important because should different parts of the game update different logical displays, chaos may ensue. How do we ensure this? We do so by implementing the *Singleton* pattern, which enforces only a instance of an object being created. This is done as follows:

We create a *static* variable called *objectDisplayGrid* that is of type ObjectDisplayGrid, and initialize it to **null**. We make our ObjectDisplayGrid constructor private, and create a public static function called *getObjectDisplayGrid* which checks to see if objectDisplayGrid is **null**, and if it is initializes it by creating a new ObjectDisplayGrid object using the private constructor. If objectDisplayGrid is not null, simply return the reference to the objectDisplayGrid object that was created earlier.

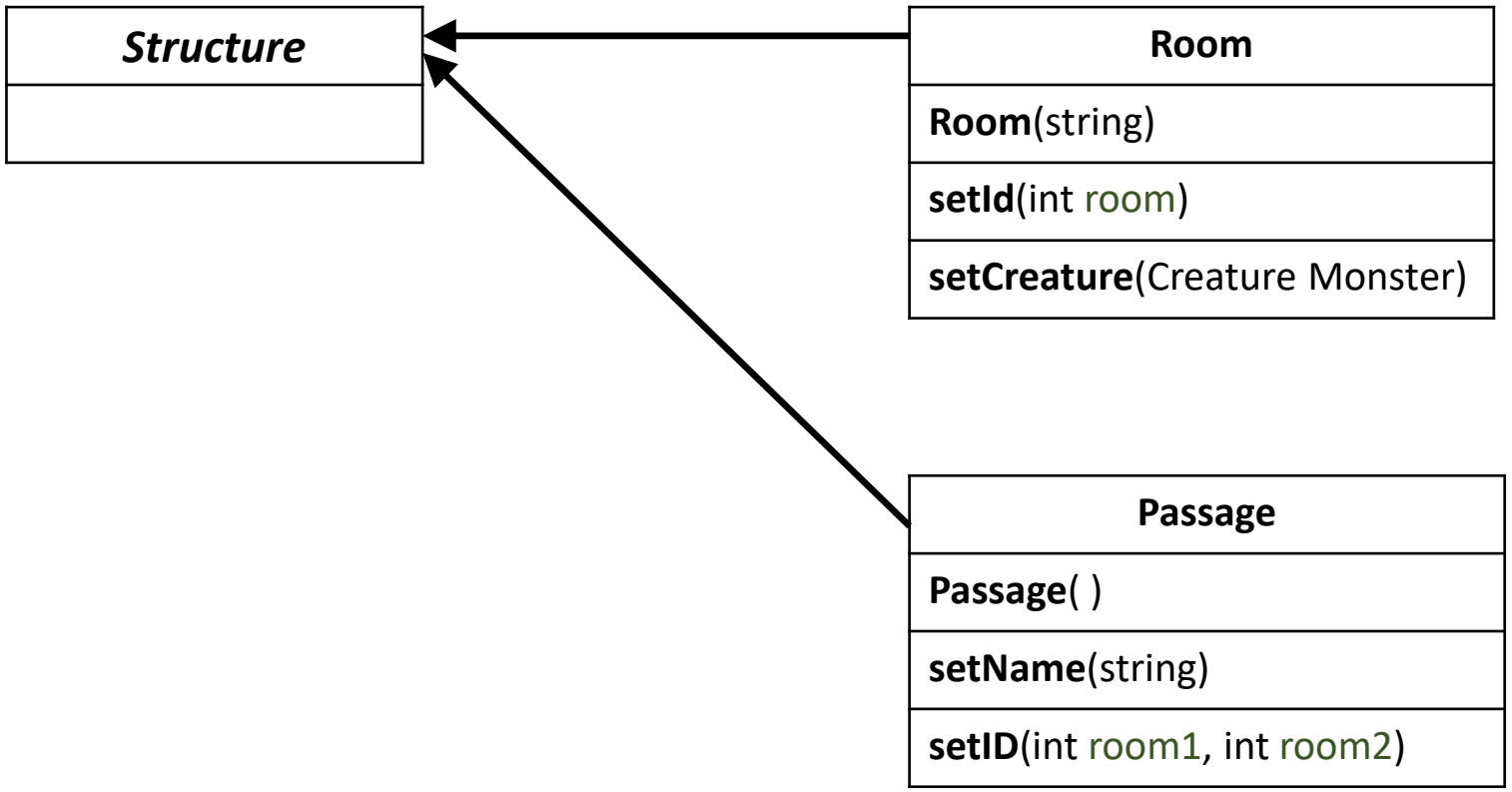
Because the constructor is private, other code cannot call it and create additional instances of ObjectDisplayGrid. And if we only call the constructor within the getObjectDisplay grid function, one and only one instance of ObjectDisplayGrid will ever be completed.

This pattern may be useful in other parts of your program where you want to only allow a single instance of some class to be created.

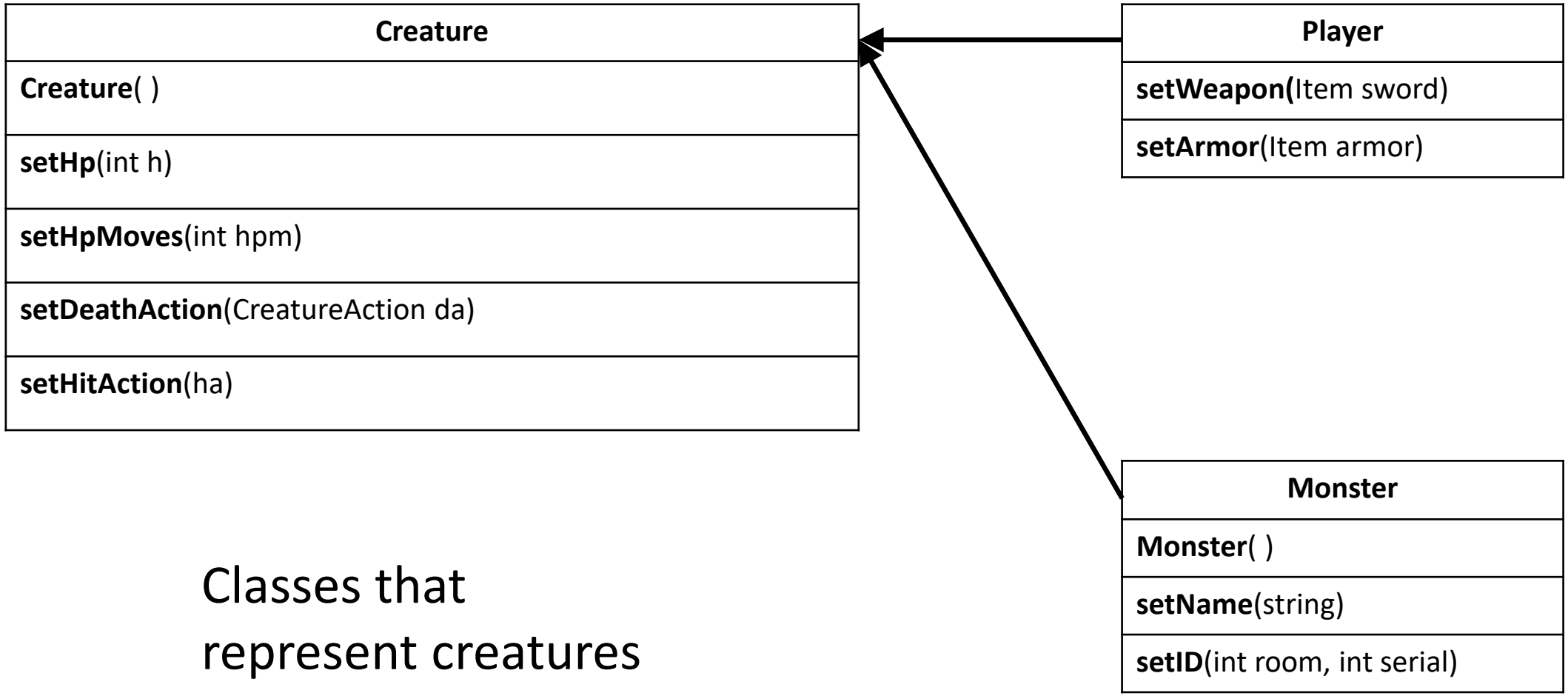


The Displayable base class and immediate subclasses. All objects that are displayable on the game inherit, directly or indirectly, from Displayable.

Classes that can be displayed

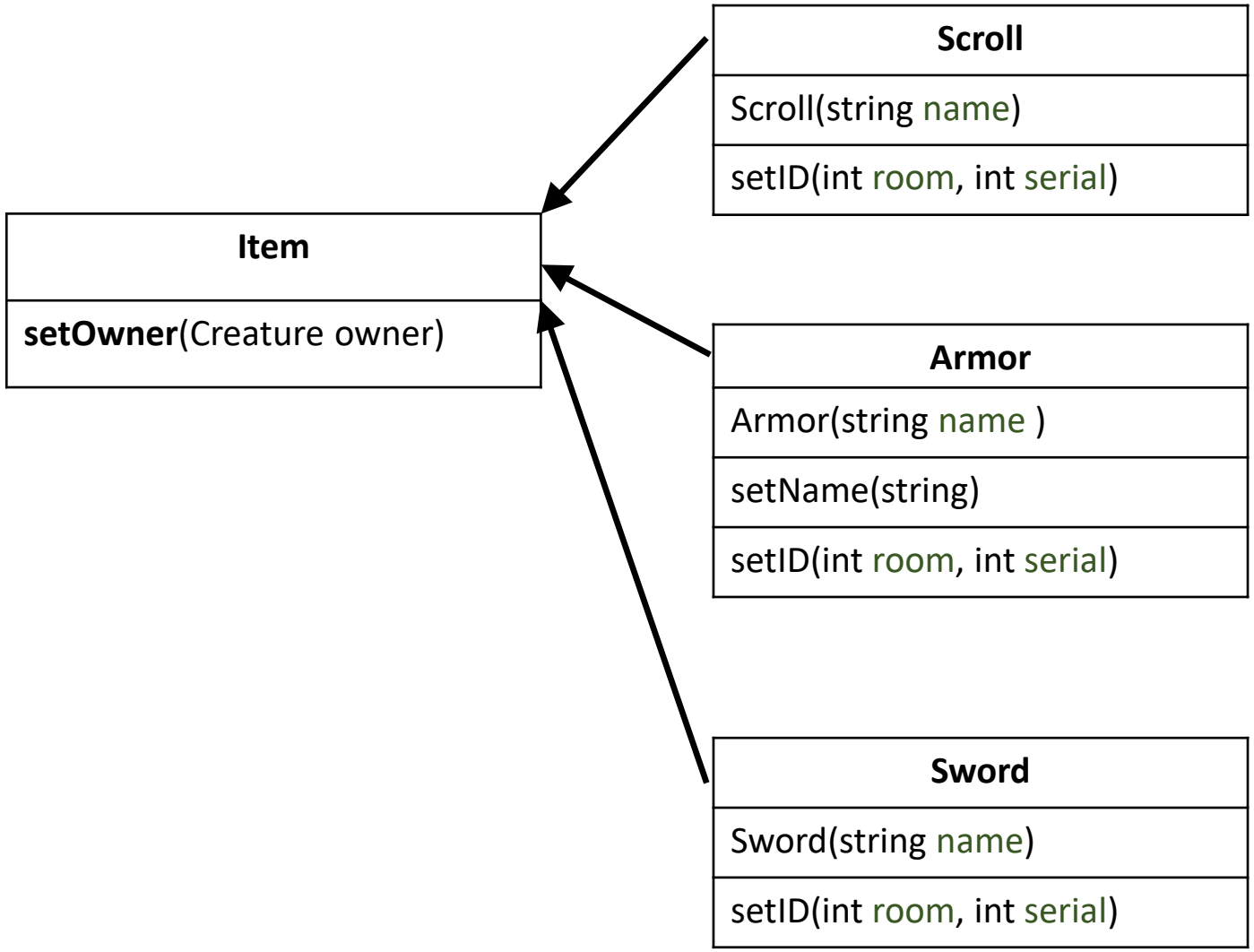


Classes that represent structures. Note as shown in the first slides there may be others



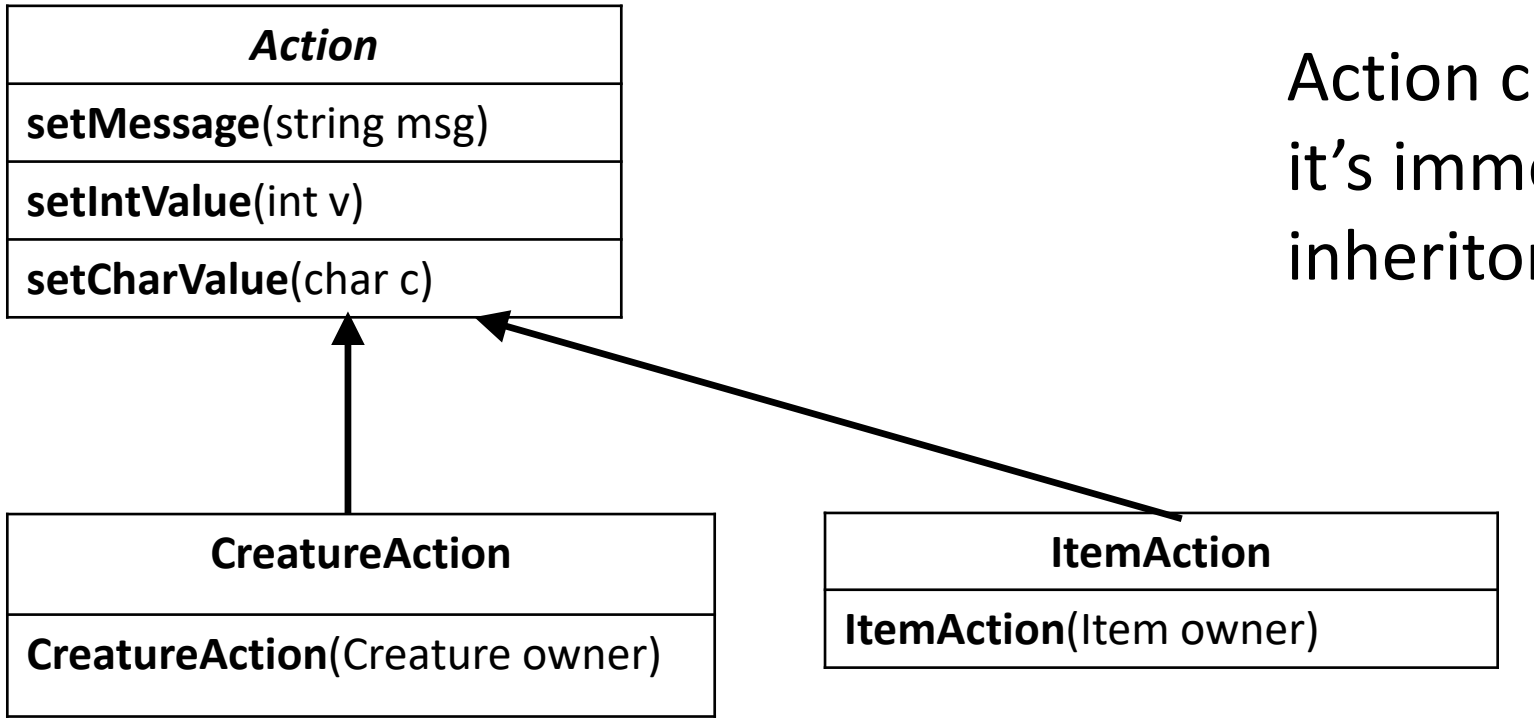
Classes that
represent creatures

Items



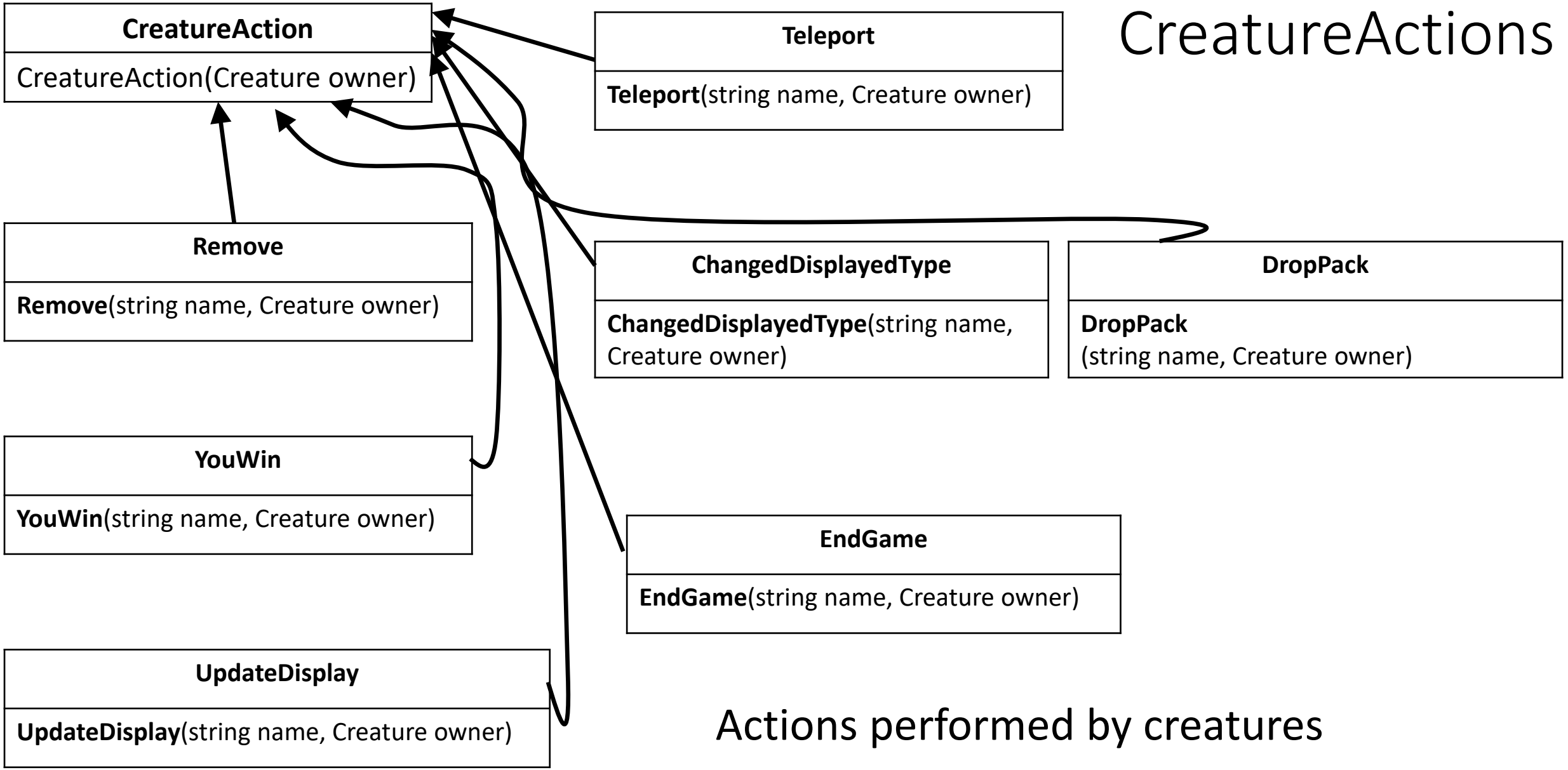
Classes that represent items

Action



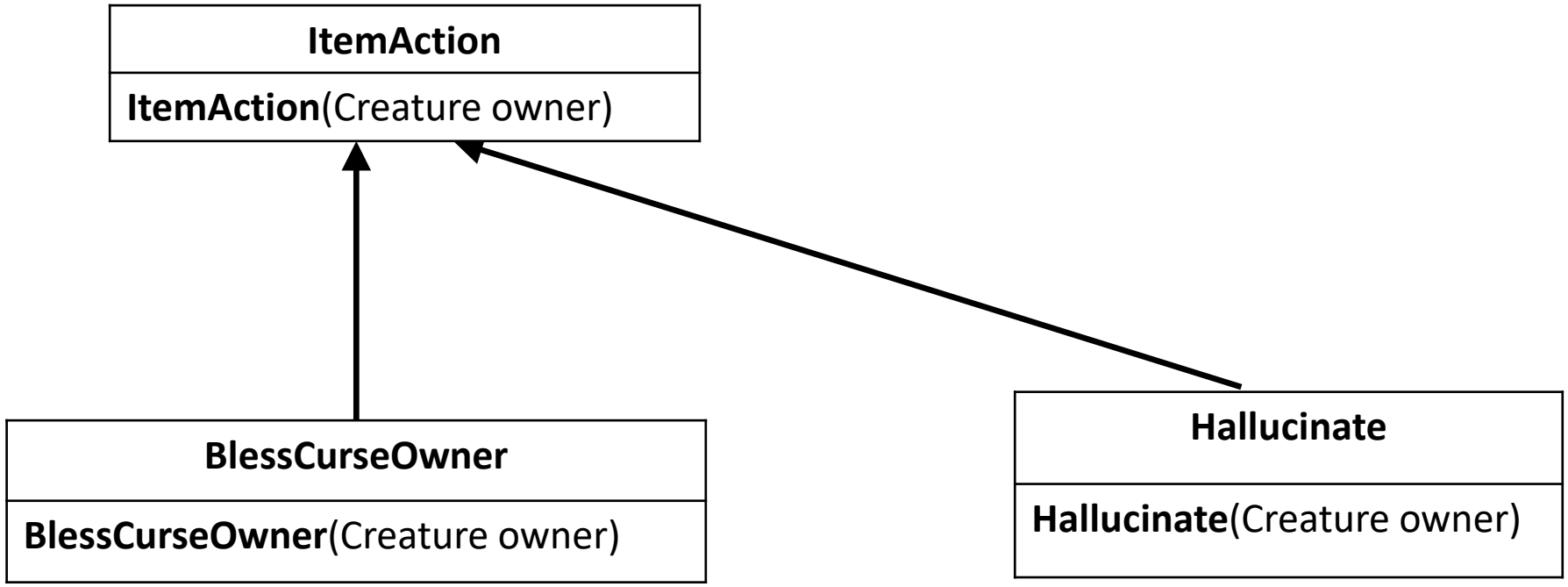
Action classes and it's immediate inheritors

CreatureActions



Actions performed by creatures

ItemActions



Actions performed by items