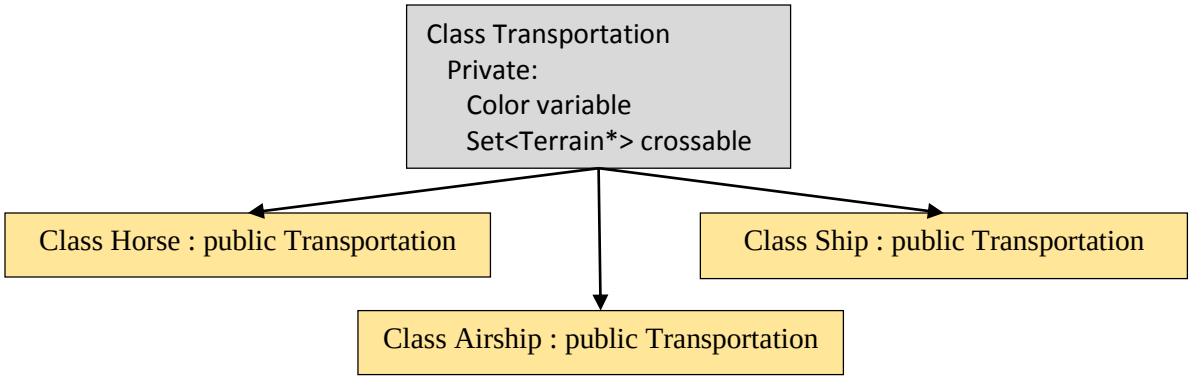
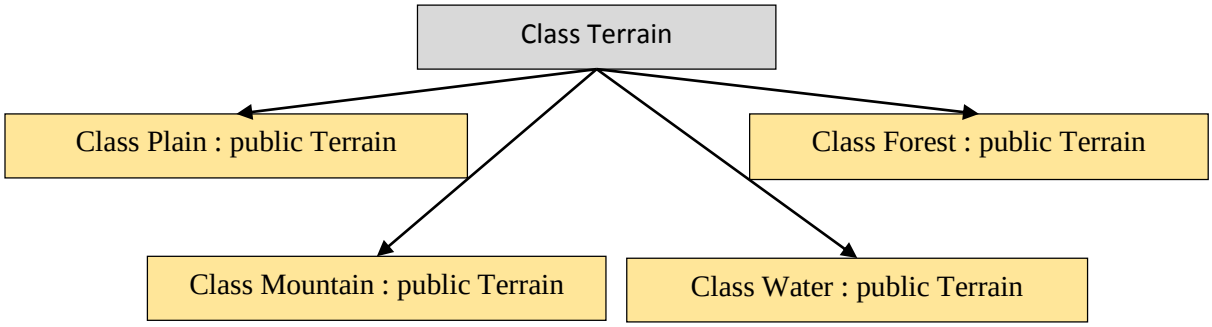
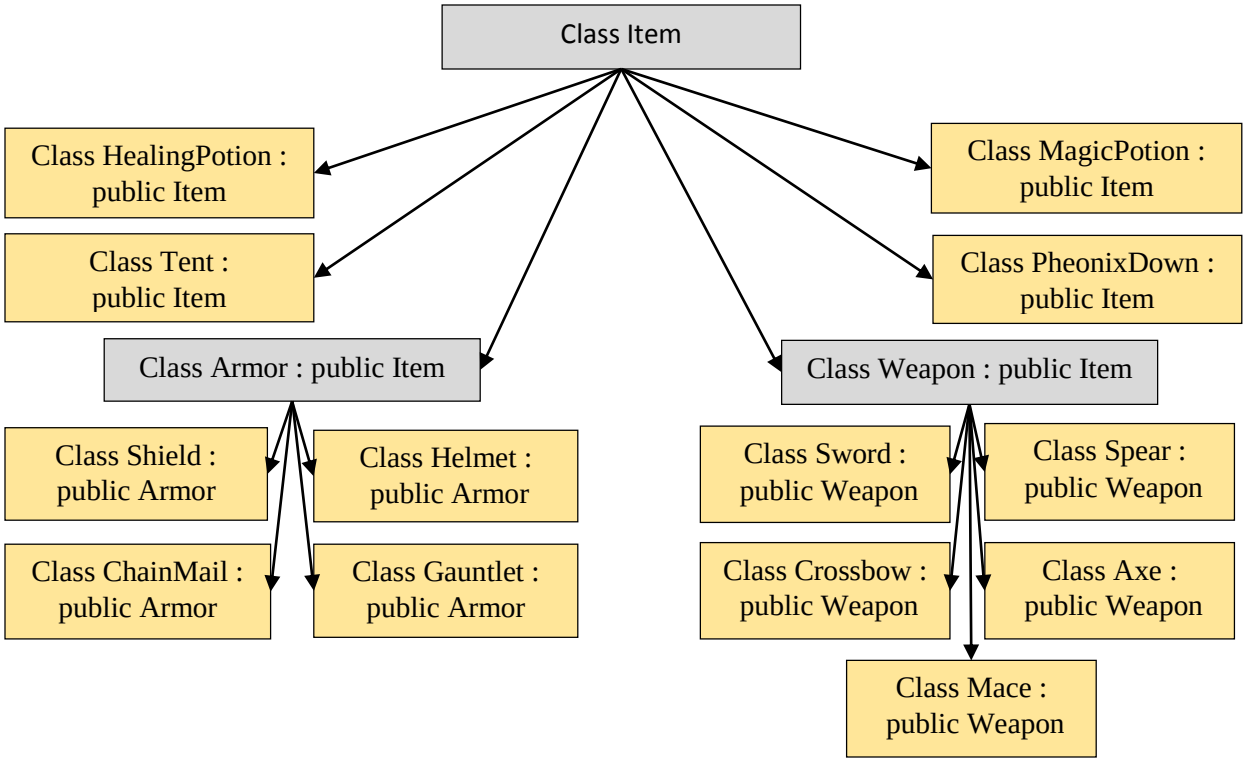


The three non-abstract classes Horse, Ship, and Airship inherit from the abstract class Transportation. They are all a type of transportation with a color the player can set and types of terrain they can cross. The crossable terrains are stored in a set for uniqueness and fast look-up. Transportation is abstract because it is a concept and not a game object.



The four non-abstract classes Plain, Forest, Mountain, and Water inherit from the abstract class Terrain because they are all a type of terrain. Terrain is abstract because it is a concept and the game world must be made up of specific types.



The four non-abstract classes HealingPotion, MagicPotion, Tent, and PheonixDown inherit from the abstract class Item because they are all a type of item. Item is abstract because it is a concept and game objects must be specific things. The abstract classes Armor and Weapon inherit from Item because they have additional combat-related properties. The non-abstract classes Shield, Helmet, ChainMail, and Gauntlet inherit from Armor and Sword, Spear, Crossbow, Axe, and Mace inherit from Weapon. Their parent classes Armor and Weapon are abstract because they are concepts but game objects must be specific things.

```
Class Player
  Private:
    Characteristic variable(s)
    map<Item*, int count> inventory
    set<Transportation*> mounts
    Weapon* currentWeapon
```

Player is a non-abstract class so it can be instantiated. In addition to variables for player characteristics, it has an inventory map, mount set, and Weapon*. The map stores an Item as a key with the number owned as the value for quick look-up. Mounts is a set for uniqueness and quick look-up. Note that depending upon how a mount is accessed, a list or map might be better. This implementation assumes a Transportation* is available.