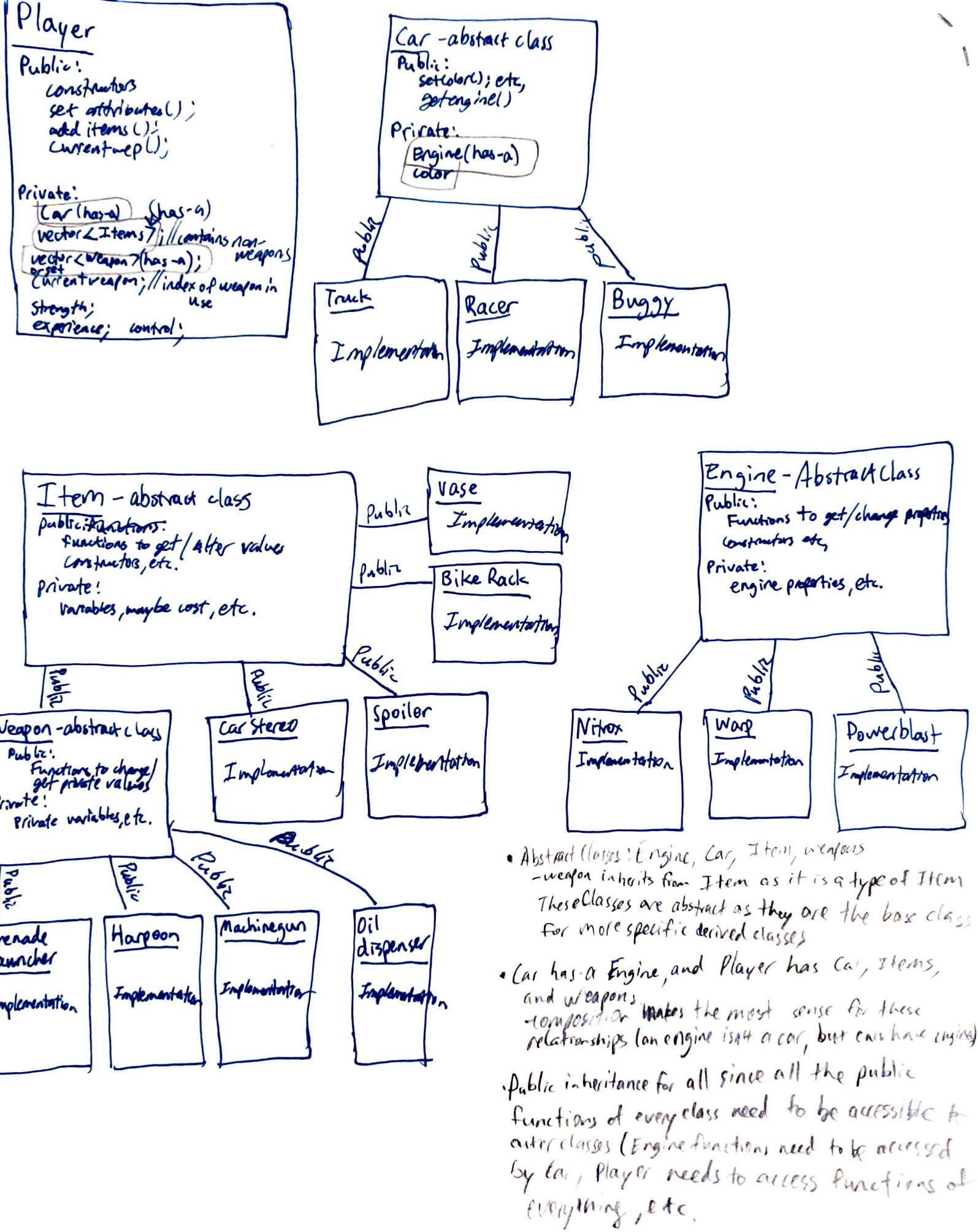

- Abstract classes: Engine, Car, Item, weapons
  - weapon inherits from Item as it is a type of Item
  These classes are abstract as they are the base class for more specific derived classes
- Car has-a Engine, and Player has Car, Items, and weapons
  - composition makes the most sense for these relationships (an engine isn't a car, but cars have engines)
- Public inheritance for all since all the public functions of every class need to be accessible to other classes (Engine functions need to be accessed by Car, Player needs to access functions of everything, etc.