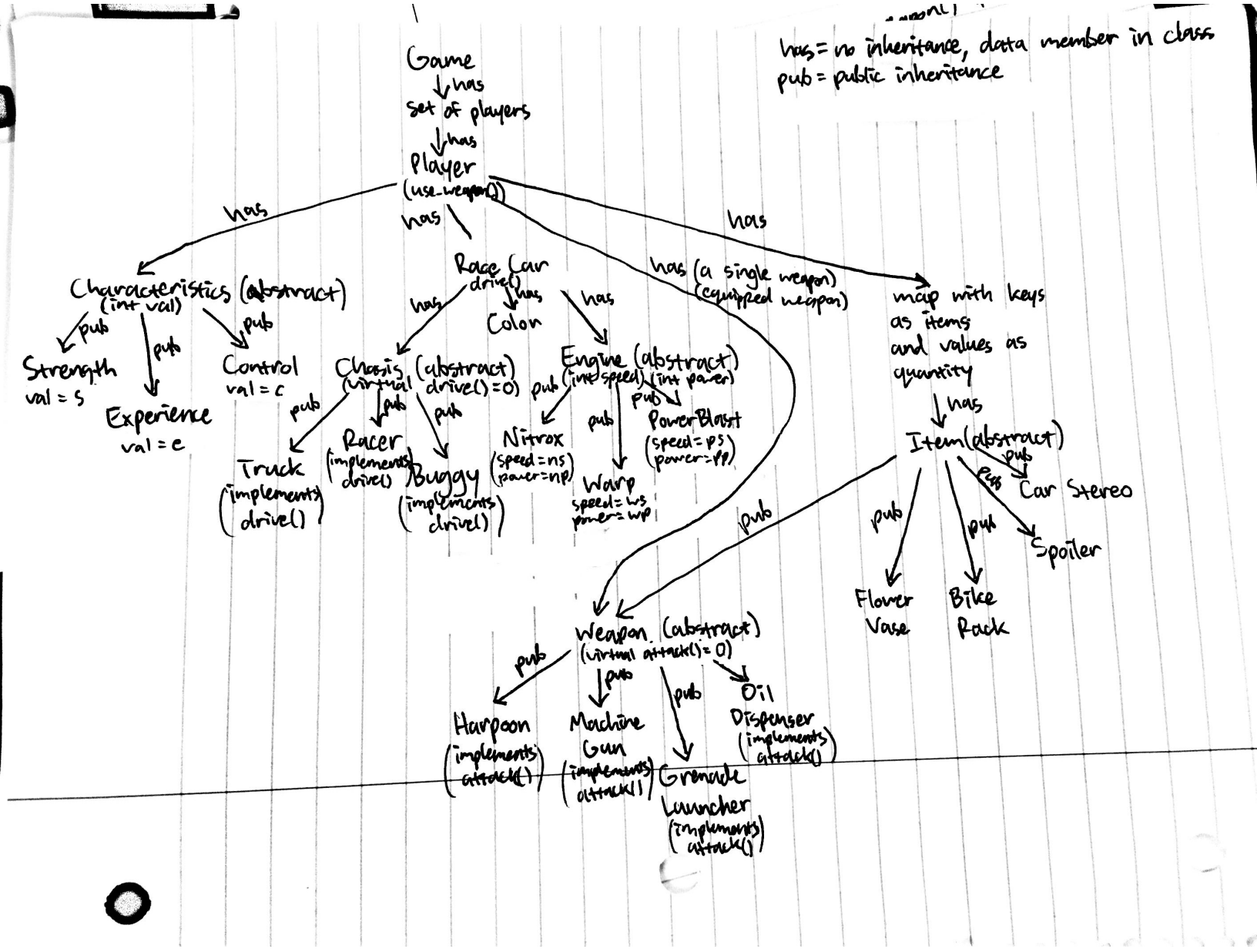
has = no inheritance, data member in class
pub = public inheritance
Game
has
set of players
has
Player
(use_weapon())
has
has
has
has (a single weapon)
(equipped weapon)
Characteristics (abstract)
(int val)
pub
pub
pub
Strength
val = s
Experience
val = e
Control
val = c
Race Car
drive()
has
has
has
Color
Chassis (abstract)
(virtual drive() = 0)
pub
pub
pub
Truck
(implements drive())
Racer
(implements drive())
Buggy
(implements drive())
Engine (abstract)
(int speed)
(int power)
pub
pub
pub
Nitrox
(speed = ns)
(power = np)
Warp
speed = ws
power = wp
PowerBlast
(speed = ps)
(power = pp)
map with keys
as items
and values as
quantity
has
Item (abstract)
pub
pub
pub
pub
Car Stereo
Spoiler
Flower
Vase
Bike
Rack
pub
pub
Weapon (abstract)
(virtual attack() = 0)
pub
pub
pub
Harpoon
(implements attack())
Machine
Gun
(implements attack())
Grenade
Launcher
(implements attack())
Oil
Dispenser
(implements attack())

My game class has a set of players as a data member. Each player has certain characteristics - which I chose to be abstract because we have to choose which characteristics they have - and the different characteristics inherit from the abstract class characteristics. Each Player also has a race car, which itself has a chassis, color, and engine. Chassis and engine are abstract because we have to specify what type of chassis and engine the car has, and therefore the different chassises and engines inherit from their respective abstract classes. Player has a map of items - the key is the item and the value is the number of that item that the player has. Item is an abstract class so all items inherit from it. Weapons is also an abstract class, because there are many different types of weapons. I also gave the player an equipped weapon ^ data member because the player can only use one weapon at a time.

I gave the player class a use_weapon() function, which calls attack() for the players currently equipped weapon. I also gave the race car class a function drive(). This calls the chasis's drive() function, which is a pure virtual and is implemented depending on the type of chasis it is. I also gave the abstract class weapon a pure virtual attack() function, which calls the appropriate attack() function depending on the type of weapon it is. I gave the characteristics strength, experience, and control classes each an int value to show how much of that characteristic the player has. The int was declared in the abstract class characteristic and I give the int a value in the concrete classes. I did the same with Engine: I declared the variables speed and power in Engine and give the values in the concrete classes that inherit from Engine.