Just two problems this week, and I've pushed the deadline back a day since I'm slow getting this posted.

Get Assignment link: https://classroom.github.com/a/Y1EnDhYf

1. Define a `Card` class suitable for representing a standard playing card, which is identified by two components: a *rank* and a *suit*. The rank is stored internally as an integer between 1 and 13, where an ace is a 1, a jack is an 11, a queen is a 12, a king is a 13, and the numbers 2-10 represent themselves. The suits can also be represented as an integer between 0 and 3, corresponding to clubs, diamonds, hearts and spades, respectively. Your `Card` class should include the following methods:

   - A constructor that takes two arguments in the form of the numeric values of the rank and suit. If `Card` is called as `Card(10,1)` for instance, it should create a 10 of diamonds.
   - A `__str__` method that converts the card to a string representation, with one character representing the rank (either the number or the first letter of the name) and another corresponding to the first letter of the suit. In order to keep all strings to two characters, you can let the rank 10 be represented with a `"T"`. So a card with a rank of 7 and a suit of spades (3) should have the string representation `"7S"`.
   - Getter methods called `get_rank` and `get_suit` which return the numeric values.

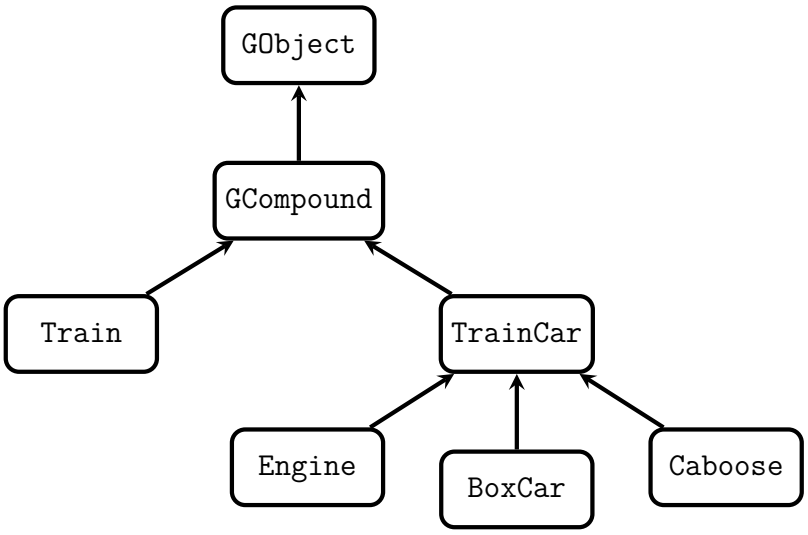
   Some examples of testing might look like:

```
>>> C1 = Card(3,0)
>>> print(C1.get_rank())
3
>>> print(C1.get_suit())
0
>>> print(C1)
3C
>>> C2 = Card(12,3)
>>> print(C2)
QS
```

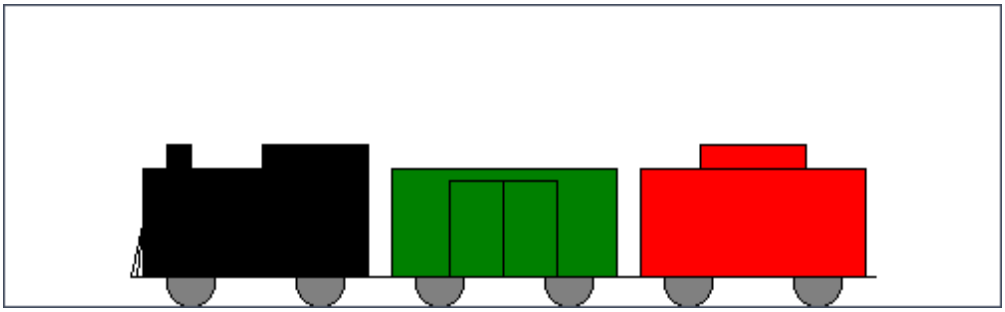
   In addition, you should write a test function creates and then prints the string representation of every card in a standard deck, with each suit appearing on a separate line. The output of the test program should thus look like:

```
AC, 2C, 3C, 4C, 5C, 6C, 7C, 8C, 9C, TC, JC, QC, KC
AD, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D, TD, JD, QD, KD
AH, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, TH, JH, QH, KH
AS, 2S, 3S, 4S, 5S, 6S, 7S, 8S, 9S, TS, JS, QS, KS
```

2. The file `Prob2.py` in the repository contains a partial implementation of a class hierarchy for drawing a train. The complete hierarchy has the form shown below, but the current implementation only includes the code for the `BoxCar` subclass.



Complete the implementation by defining the subclasses `Engine` and `Caboose` and then use these classes to create a train that looks like this:



The starting code in the repository already includes definitions of `click_action` and `step` that will make the entire train move forward until it disappears off the left edge of the window.

Hints: While the actual code you need to write here isn't very complicated, it *absolutely does* require you to be able to work out what each piece of the current code is doing. In particular, some questions to ask yourself will involve:

- How can new cars be easily added to the train object?
- Where in the `GCompound` does `TrainCar` define the origin (0,0) point? Knowing this will be very important for placing the new parts!
- Can you sketch out what distance each given constant is representing?