Eric Roberts and Jed Rembold
CS 151

Handout #16
November 4, 2022

Problem Set #6— Classes and Objects

Due: 5:00 P.M., Friday, November 11

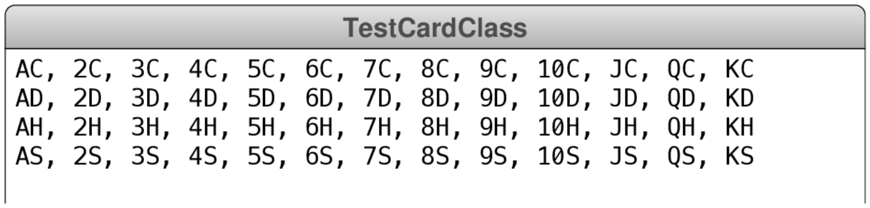
Problem 1 (Chapter 9, exercise 4, page 351)

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 as an integer between 1 and 13 in which an ace is a 1, a jack is an 11, a queen is a 12, and a king is a 13. For the convenience of clients, the `Card` class exports constants named `Card.ACE`, `Card.JACK`, `Card.QUEEN`, and `Card.KING`. The suit is also represented as an integer between 0 and 3, which are exported as the constants `Card.CLUBS`, `Card.DIAMONDS`, `Card.HEARTS`, and `Card.SPADES`, respectively.

In addition to these constants, the `Card` class should export the following methods:

- A constructor that takes either of two forms. If `Card` is called with two arguments, as in `Card(10, Card.DIAMONDS)`, it should create a card from those components. If `Card` is called with one argument, it should interpret the argument as a string composed of a rank (either a number or the first letter of a symbolic name) and the first letter of the suit, as in `"10D"` or `"QS"`.
- A `__str__` method that converts the card to a string as described in the outline of the constructor. The card `Card(Card.QUEEN, Card.SPADES)`, for example, should have the string representation `"QS"`.
- The getter methods `get_rank` and `get_suit`.

You should also write a test program that displays the string representation of every card, with each suit appearing on a separate line. The output of the test program should look like this:

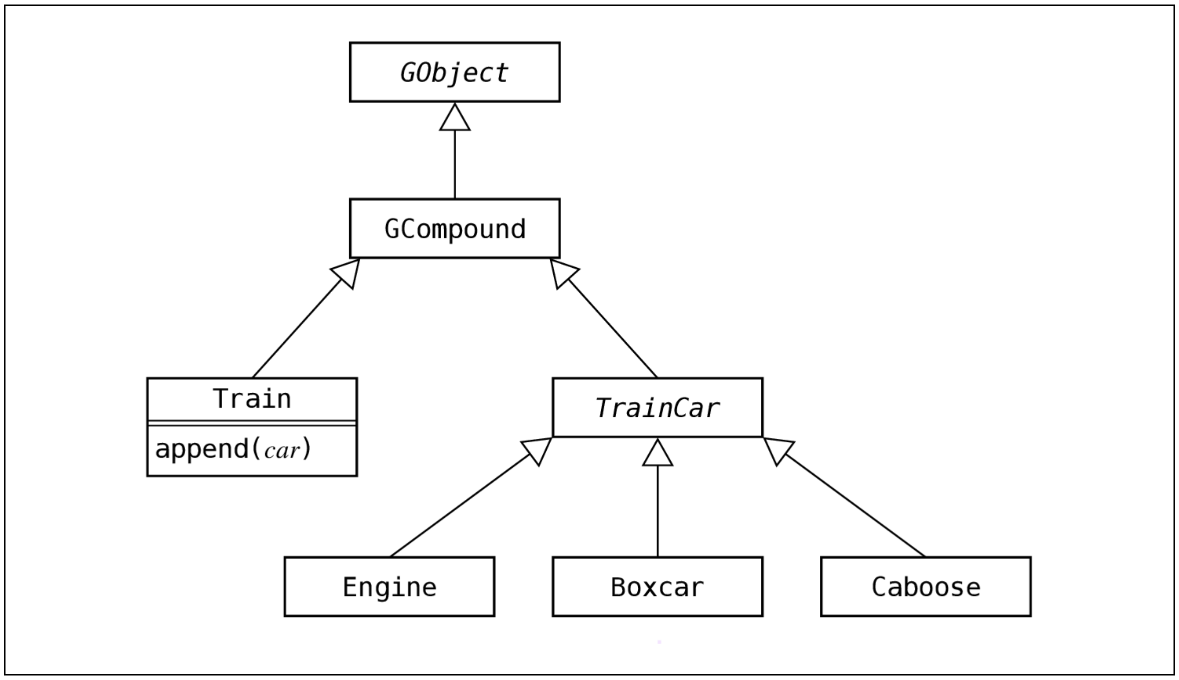


```
AC, 2C, 3C, 4C, 5C, 6C, 7C, 8C, 9C, 10C, JC, QC, KC
AD, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D, 10D, JD, QD, KD
AH, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, 10H, JH, QH, KH
AS, 2S, 3S, 4S, 5S, 6S, 7S, 8S, 9S, 10S, JS, QS, KS
```

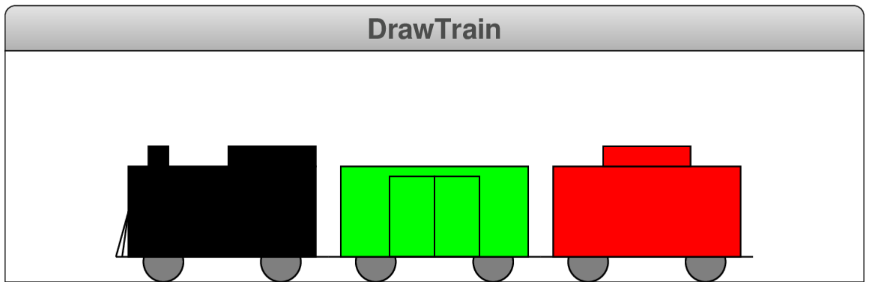
Problem 2

The file `DrawTrain.py` in the starter folder contains a partial implementation of a class hierarchy for drawing a train. The complete hierarchy has the form shown in Figure 1 at the top of the next page. The code in `DrawTrain.py`, however, includes only the code for the `BoxCar` subclass.

Figure 1. Inheritance hierarchy for train cars



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



The code in the starter file 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.