**VEER**

Here at Veneer we strive to provide the best marketplace experience to connect vetted art dealers with premium galleries. Create a marketplace for people and organizations to buy and sell pieces of art!

In this project we’ll be developing classes and objects that represent the various responsibilities of art dealership software.

The Thin Veneer of Viability

2.Here at Veneer, our main concern is the buying and selling of priceless art works. Let’s start out by building a model for these works of art.

Define a class called Art.

3.Give Art a constructor that takes self and four additional parameters: artist, title, medium, and year.

Assign these values to self.artist, self.title, self.medium and self.year.

4.Give your Art class a string representation method. This method should return a representation of the artwork in as close to standard citation format as we can manage (without italics). It should state:

The artist’s name

The name of the artwork in quotes

The year the work was created

The medium

For instance:

Monet, Claude. "Vétheuil in the Fog". 1879, oil on canvas.

5.Let’s see how our Art class looks, create a new work of art. Our first client wants to list a particular Picasso painting to make more space for her new fascination with Italian Futurism, so let’s see if we can use our data model for this piece:

The artist is “Picasso, Pablo”.

The work’s title is “Girl with a Mandolin (Fanny Tellier)”.

The artwork was created in 1910.

The medium is “oil on canvas”.

Save this work of art into the variable girl\_with\_mandolin.

6.Print out girl\_with\_mandolin and run your code, does your code produce this output?

Picasso, Pablo. "Girl with a Mandolin (Fanny Tellier)". 1910, oil on canvas.

The Marketplace of Artistic Ideas

7.In order to buy and sell works of art, we need a marketplace that will maintain the responsibilities of buying, selling, listing, and delisting of those artworks.

Create a new class called Marketplace.

8.Give your Marketplace class a constructor. In Marketplace‘s constructor, define self.listings as a new list.

9.Create an .add\_listing() method for your Marketplace class. This should take two arguments: self and new\_listing. Have .add\_listing() add the new listing to Marketplace‘s listings attribute.

10.Since we’ll need a way to remove listings when they expire or are acted upon, let’s implement a .remove\_listing() method for our Marketplace

11.The main usage of our application will be the perusal of a marketplace’s listings. Let’s include that functionality as well.

Add a .show\_listings() method to your Marketplace class that iterates through each listing in self.listings and prints them all out.

12.Create our main marketplace by instantiating Marketplace and saving it into the variable veneer.

13.Print out the results of veneer.show\_listings(). This should be empty for now so won’t print anything, but it’s good to test if your code has any errors!

We Need Clients!

14.Now for the most important aspect of a marketplace, clients! Create a new class called Client.

15.Give our Client class a constructor. A client should have the following data:

self.name the name of the person or institution.

self.location is the name of the location of the museum or “Private Collection” if the client is a collector.

self.is\_museum, a boolean value representing whether the client is a museum (if True) or a collector (if False).

name, location, and is\_museum should all be passed as arguments to the constructor.

16.Now instantiate our first Client: her name is Edytta Halpirt and she is a collector and not a museum.

Save our new Client to a variable called edytta.

17.Every purchase requires a buyer and a seller, let’s create a second Client with the following information:

It’s name is “The MOMA”

It is located in “New York”

It is a museum.

Save this Client to a variable called moma.

18.We need to get the MOMA to purchase a Picasso from Edytta, but for now try running your code to make sure our Client class doesn’t produce any errors.

Updating Our Art Class

19.Now that we have Clients our works of Art can have owners! Let’s update our Art class constructor to take an additional parameter, owner, and assign that to self.owner.

20.A full citation of a work of art necessarily includes the name of the person/entity whose collection it includes, as well as the location if that place is a museum.

Because the work of art has an owner property, we can retrieve some of that information from self.owner.

Let’s update Art‘s string representation method to add the self.owner.name at the very end, followed by a comma, followed by self.owner.location, followed by a period.

21.Move our instantiation of girl\_with\_mandolin to after our instantiation of edytta. When creating the Art object for girl\_with\_mandolin, pass in edytta as the owner.

22.Move our print statement printing out girl\_with\_mandolin to after its new instantiation. Does it print out the following?

Picasso, Pablo. "Girl with a Mandolin (Fanny Tellier)". 1910, oil on canvas. Edytta Halpirt, Private Collection.

Don't Be Listless

23.Now that we have a marketplace to facilitate the buying and selling, let’s create our class to list works of art!

Create a new class called Listing. We’ll use these as listings for our Marketplace.

24.Our Listing class needs a constructor! It should define the following instance variables:

self.art, the work of art being listed

self.price, the price of the work

self.seller an instance of Client.

Each instance variable should be set equal to an argument passed to the constructor.

25.Give the Listing a string representation which should print out the following:

The name of the work of art

The price of the work of art

Remember not to use the string representation of Art for this, it’s too verbose and our clientele will want to parse the listings without reading all of the metadata unless they’re interested in purchasing in the artwork.

26.Update our Client class to have a new method, .sell\_artwork(). This method should take three parameters: self, artwork, and price.

.sell\_artwork() should do the following:

Check that artwork.owner is the same (==) as self (i.e., make sure the client owns the art they’re trying to sell).

Create a new Listing with the given art, price, and client.

Add the listing to the marketplace using veneer.add\_listing().

27.Use edytta.sell\_artwork() to create a listing for girl\_with\_mandolin. Edytta wants to sell it for $6M (USD).

28.Call veneer.show\_listings(), is our newly listed work of art on the market?

Buy Low, Sell High

29.There’s one last piece of functionality before we’re ready to hit the market (so to speak), our clients need to be able to buy art!

Create a .buy\_artwork() method for the Client class. .buy\_artwork() should take two arguments: self and artwork.

Start by having .buy\_artwork() check that the artwork is not owned by the client.

30.The next thing .buy\_artwork() should do is make sure that the artwork is listed in veneer.listings. Save the appropriate listing as art\_listing, we’ll need to remove it later.

31.If the art is not currently owned and is listed then go through with the transaction! .buy\_artwork() should do the following:

Change the artwork.owner to the client doing the purchasing.

Remove the listing from the marketplace using veneer.remove\_listing()

32.The MOMA is very interested in purchasing girl\_with\_mandolin. Call .buy\_artwork() from the moma instance with girl\_with\_mandolin as an argument.

33.Finally, print out girl\_with\_mandolin one last time. It should display the following:

Monet, Claude. "Vétheuil in the Fog". 1879, oil on canvas. The MOMA, New York.

34.Also call veneer.show\_listings(). There shouldn’t be any listings left! Congrats on one purchase successfully made!

35.Amazing! We built out a whole marketplace with buyers, sellers, art, and listings!

Here are some more things you could try:

Add a wallet instance variable to clients, update the buying and selling of artworks to also exchange dollar amounts.

Create a wishlist for your clients, things that are listed but they’re not sure if they should purchase just yet.

Create expiration dates for listings! Have out of date listings automatically removed from the marketplace.