Lone Star
section 4

Classes Part 2 quiz

1. (12.1)

   (a) Write a class for a Song with the below instance variables and methods.
   The Song object should have the ability to be passed both initial values.
   You may pick anything you like for the string representation of the object.
   A Song has an artist and can be added to a playlist.
   As part of your class, write a method called play, that displays the song information.
   Hint: for this method, you should print the title and artist of the song.

Song
title artist
get_artist set_artist play __str__

   (b) Write a class for a Playlist.
   The class should start (instantiate) with a playlist_name, and no Songs in it.
   Write a method to add a Song.
   Write a method that plays all songs one after another.
   You may pick anything you like for the string representation of the object.

Playlist
playlist_name songs
add_song play_all __str__

   (c) Create an instance of the Playlist class and add two Songs to it.
   Call the method to play all songs in your playlist (play_all).
   You can make up any titles and artists for Songs and a playlist_name for a Playlist.

2. (12.2)

(a) Write a class for an *Employee* with the below instance variables and methods.
An *Employee* should start (be initialized) with both a name and position.
You may pick anything you like for the string representation of the object.

Employee
name position
get_position set_position __str__

(b) Write a class for a *Department* within a company. The *Department* should start with a name and a budget, but with no employees. That is, it should not have the ability to be initialized with employees in it. However, the *Department* should have the ability to add employees as well as show all of the employees in the *Department*.
Additionally, a department is large if it has 10 or more employees. Write a method called *is_large* which returns True is the department has 10 or more employees and False otherwise.
You may pick anything you like for the string representation of the object.

Department
dept_name budget employees
get_budget set_budget add_employee show_staff_list __str__

(c) Create an instance of the *Department* class and add 2 *Employee*s to it.

Dot Matrix
section 5

Classes Part 2 quiz

1. (12.1)

   (a) Write a class for a Duck with the below instance variables and methods.
   The Duck object should have the ability to be passed both initial values.
   You may pick anything you like for the string representation of the object.
   A Duck says, "Quack."
   As part of your class, write a method called speak, that makes a duck quack!
   Hint: for this method, you can just print the word Quack!

Duck
name color
get_color set_color speak __str__

   (b) Write a class for a Pond.
   The class should start (instantiate) with a name, and no Ducks in it.
   Write a method to add a Duck.
   Write a method that makes all of the ducks in the pond quack one time each.
   You may pick anything you like for the string representation of the object.

Pond
name ducks
add_duck ducks_quack __str__

   (c) Create an instance of the Pond class and add two Ducks to it.
   Call the method to make all ducks in your pond quack (ducks_quack).
   You can make up any names or colors for Ducks and a Pond.

2. (12.2)

(a) Write a class for a Lion with the below instance variables and methods.
The Lion object should have the ability to be passed both initial values.
You may pick anything you like for the string representation of the object.
A Lion says, "Roar."
As part of your class, write a method called roar, that makes a lion roar!
Hint: for this method, you can just print the word Roar!

Lion
name gender
get_name set_name roar __str__

(b) Write a class for a Zoo.
The class should start (instantiate) with a name, and no Lions in it.
Write a method to add a Lion.
Write a method that makes all of the lions in the zoo roar one time each.
Write a method called *count_lions* which reports the number of male lions and female lions at the zoo. eg. Your code could print: *1 male, 4 female.*
You may pick anything you like for the string representation of the object.

Zoo
location lions
add_lion lions_roar count_lions __str__

(c) Create an instance of the Zoo class and add two Lions to it.
Call the method to make all lions in your zoo roar (lions_roar).
You can make up any names or genders for Lions and a location for a Zoo.

Dark Helmet
section 4

Classes Part 2 quiz

1. (12.1)

   (a) Write a class for a Duck with the below instance variables and methods.
   The Duck object should have the ability to be passed both initial values.
   You may pick anything you like for the string representation of the object.
   A Duck says, "Quack."
   As part of your class, write a method called speak, that makes a duck quack!
   Hint: for this method, you can just print the word Quack!

Duck
name color
get_color set_color speak __str__

   (b) Write a class for a Pond.
   The class should start (instantiate) with a name, and no Ducks in it.
   Write a method to add a Duck.
   Write a method that makes all of the ducks in the pond quack one time each.
   You may pick anything you like for the string representation of the object.

Pond
name ducks
add_duck ducks_quack __str__

   (c) Create an instance of the Pond class and add two Ducks to it.
   Call the method to make all ducks in your pond quack (ducks_quack).
   You can make up any names or colors for Ducks and a Pond.

2. (12.2)

(a) Write a class for a Product with the below instance variables and methods.
The Product object should have the ability to be passed both initial values.
You may pick anything you like for the string representation of the object.
A Product has a price and can be added to a cart.
As part of your class, write a method called display_details, that displays the product information.
Hint: for this method, you should print the name and price of the product.

Product
name price
get_price set_price display_details __str__

(b) Write a class for a ShoppingCart.
The class should start (instantiate) with a customer_id, and no Products in it.
Write a method to add a Product.
Write a method that calculates the total price of all products in the cart.
You may pick anything you like for the string representation of the object.

ShoppingCart
customer_id products
add_product calculate_total __str__

(c) Create an instance of the ShoppingCart class and add two Products to it.
Call the method to calculate the total price of all products in your cart (calculate_total).
You can make up any names and prices for Products and a customer_id for a ShoppingCart.

President Skroob Classes Part 2 quiz
section 1

1. (12.1)

   (a) Write a class for a Song with the below instance variables and methods.
   The Song object should have the ability to be passed both initial values.
   You may pick anything you like for the string representation of the object.
   A Song has an artist and can be added to a playlist.
   As part of your class, write a method called play, that displays the song information.
   Hint: for this method, you should print the title and artist of the song.

Song
title artist
get_artist set_artist play __str__

   (b) Write a class for a Playlist.
   The class should start (instantiate) with a playlist_name, and no Songs in it.
   Write a method to add a Song.
   Write a method that plays all songs one after another.
   You may pick anything you like for the string representation of the object.

Playlist
playlist_name songs
add_song play_all __str__

   (c) Create an instance of the Playlist class and add two Songs to it.
   Call the method to play all songs in your playlist (play_all).
   You can make up any titles and artists for Songs and a playlist_name for a Playlist.

2. (12.2)

(a) Write a class for a Product with the below instance variables and methods.
The Product object should have the ability to be passed both initial values.
You may pick anything you like for the string representation of the object.
A Product has a price and can be added to a cart.
As part of your class, write a method called display_details, that displays the product information.
Hint: for this method, you should print the name and price of the product.

Product
name price
get_price set_price display_details __str__

(b) Write a class for a ShoppingCart.
The class should start (instantiate) with a customer_id, and no Products in it.
Write a method to add a Product.
Write a method that calculates the total price of all products in the cart.
You may pick anything you like for the string representation of the object.

ShoppingCart
customer_id products
add_product calculate_total __str__

(c) Create an instance of the ShoppingCart class and add two Products to it.
Call the method to calculate the total price of all products in your cart (calculate_total).
You can make up any names and prices for Products and a customer_id for a ShoppingCart.