CP1890 – Assignment 1

Handed Out: 04 Feb 2024
Due Date: 18 Feb 2024

Objectives

This problem set will introduce you to Pycharm environment, Github usage, proper versioning and code contribution skills in a professional environment while building up on what you learnt on last semester with the introduction of classes, number formatting and proper handling of datetime objects.

Collaboration

You will be working with your assigned group. Each group will designate a leader and the members will make commits to the leader's fork of the class by submitting a pull request and going through with the review process. You are free to divide the assignment objectives among your group members. Please note that each pull request has to be reviewed with the two other members of your group.

Once the assignment is complete, **the leader of the group** will raise a pull request to merge the code with the main class repository and will assign the **other group members as reviewers for that pull request.**

Submission

You will create one folder for your group under assignments with your group name. In this folder, you will have one python file for each question. You need to ensure that suitable comments are present in the code and the code has gone through the pull request and review process with your group.. Once done, the leader should ensure the assignment link is posted to the drop box for the group that is provided on Brightspace **before the due date (no commits after the due date will be considered for grading)**.

Late submissions

Penalties for late submissions is as follows:

- Up to 24 hours after the due date: Flat 25% penalty
- Beyond 24 hours up to 48 hours: Flat 50% penalty
- Beyond 48 hours: No points for assignment

If for some reason you are unable to submit the assignment on time, please reach out to me to make alternate arrangements – which will be handled based on the merits of the case. **No extensions will be provided.** Please ensure you stay on top of the requirements and manage your time well.

Question 1

Create an object-oriented program that performs calculations on a rectangle.

Console

```
Rectangle Calculator

Height:    10
Width:     20
Perimeter: 60
Area:      200
* * * * * * * * * * * * * * * * * * * *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
* * * * * * * * * * * * * * * * * * * *

Continue? (y/n): y

Height:    5
Width:     10
Perimeter: 30
Area:      50
* * * * * * * * * *
*                 *
*                 *
*                 *
* * * * * * * * * *

Continue? (y/n): n

Bye!
```

Specifications

- Use a Rectangle class that provides attributes to store the height and width of a rectangle. This class should also provide methods that calculate the perimeter and area of the rectangle. In addition, it should provide a method that gets a string representation of the rectangle.
- When the program starts, it should prompt the user for height and width. Then, it should create a Rectangle object from the height and width and use the methods of that object to get the perimeter, area, and string representation of the object.
- Assume that the user will enter valid data.

Question 2

Create an object-oriented program that creates a deck of cards, shuffles them, and deals the specified number of cards to the player.

Console

```
Card Dealer

I have shuffled a deck of 52 cards.

How many cards would you like?: 7

Here are your cards:
Jack of Hearts
Jack of Diamonds
2 of Diamonds
6 of Spades
Jack of Spades
6 of Hearts
King of Diamonds

There are 45 cards left in the deck.

Good luck!
```

Specifications

- Use a Card class to store the rank and suit for each card. In addition, use a method to get a string representation for each card such as "Ace of Spades", "2 of Spades", etc.
- Use a Deck class to store the 52 cards in a standard playing deck (one card for each rank and suit):

  ```
  ranks: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace
  suits: Clubs, Diamonds, Hearts, Spades
  ```

- The Deck class should include a method that shuffles the deck, a method that counts the number of cards in the deck, and a method that deals a card from the deck, which should reduce the count of the cards in the deck by 1.
- When the program starts, it should get a new deck of cards, shuffle them, and display a message that indicates the total number of cards in the deck. To shuffle the cards, you can use the shuffle function of the random module described in chapter 6.
- The program should prompt the user for the desired number of cards. Then, it should deal the user the desired number of cards and display a message that indicates the number of cards left in the deck.

Question 3

Create an object-oriented program that reads a list of Customer objects from a CSV file and allows the user to display the data for a customer by specifying the customer's ID.

Console

```
Customer Viewer

Enter customer ID: 103

Art Venere
8 W Cerritos Ave #54
Bridgeport, NJ 08014

Continue? (y/n): y

Enter customer ID: 104

Lenna Paprocki
Feltz Printing
639 Main St
Anchorage, AK 99501

Continue? (y/n): y

Enter customer ID: 99

No customer with that ID.

Continue? (y/n): n

Bye!
```

Specifications

- Your instructor should provide a CSV file named customers.csv that contains customer data.
- Use a Customer class to store the customer data. This class should include these attributes: id, firstName, lastName, company, address, city, state, zip.
- In addition, this class should include a property or method that returns the full name and the full address. The address should have two lines if the company attribute is empty or three lines if the company attribute is not empty.
- Create a function that reads the customer data from the CSV file and creates Customer objects from it.
- To find the customer with the specified ID, you need to loop through each Customer object in the list of Customer objects and check whether the specified ID matches the ID stored in the Customer object.
- If the specified ID isn't found, display an appropriate message.