Read all instructions carefully. Not following instructions will result in you not earning the credit you want for the assignment.

Objectives

- Create an interface
- Implement an interface
- Define a type with a specific set of enumerated values

Problem

In Part 7, suppose you want to add functionality to your Dessert Shop application to identify what type of payment the customer will use. A line indicating the payment type will be added to the bottom of the receipt.

To do this, you will add an interface to your Dessert Shop application and update your receipt output as described below. Possible ways to pay are CASH, CARD, PHONE.

Payable interface

Create a Protocol class `Payable`. Define a type `PayType` whose only legal values are `{CASH, CARD, PHONE}`.

Payable will include two methods:

- get_pay_type(): PayType
- set_pay_type(payment_method: PayType)

Raise a ValueError if `get_pay_type` returns anything other than one of these values. Similarly, raise an error if `set_pay_type` attempts to set a value that is not of of these values.

Changes to Order class

- Implement the Payment interface. You decide how this is to be implemented and tested.
- The default value for pay method in the constructor should be `CASH`.
- Modify `__str__` method to include payment type in the order as shown in the example run.

Changes DessertShop class

- Add a method that
    - prompts the user for payment type as shown in the example run
    - validates the input
    - returns the payment type to the calling code to set the payment type before an order is printed.

Test Cases

Add pytest test cases to test the payment type of an Order. Create a new test file `test_order.py` to hold these test cases. You only need 5 test cases: 3 to check for each valid payment type, one to check for trying to set an invalid value and one for trying to get (return) an invalid value.

We are not testing the user interface with automated test cases, just Order objects with new payment types.

Key Program Requirements

1. You have added a new Payable interface to your system.
2. The PDF receipt output has been modified to include the payment method as shown.
3. Your workspace includes all source files so far and two new files:
    - `test_order.py` with cases for testing payment method
    - Payable interface is defined in file `payable.py`.

Example Run

1: Candy

2: Cookie

3: Ice Cream

4: Sunday

What would you like to add to the order? (1-4, Enter for done): 2

Enter the type of cookie: Oatmeal Raisin

Enter the quantity purchased: 4

Enter the price per dozen: 3.45

1: Candy

2: Cookie

3: Ice Cream

4: Sunday

What would you like to add to the order? (1-4, Enter for done):

1: Cash

2: Card

3: Phone

Enter payment method: 2

```
-----------------------------------------------Receipt----------------------------
Oatmeal Raisin Cookies (Box)
      4 cookies @ $3.45/dozen:                          $1.15           [Tax: $0.08]
Candy Corn (Bag)
      1.50 lbs. @ $0.25/lb.:                            $0.38           [Tax: $0.03]
Gummy Bears (Bag)
      0.25 lbs. @ $0.35/lb.:                            $0.09           [Tax: $0.01]
Chocolate Chip Cookies (Box)
      6 cookies @ $3.99/dozen:                          $2.00           [Tax: $0.14]
Pistachio Ice Cream (Bowl)
      2 scoops @ $0.79/scoop:                           $1.58           [Tax: $0.11]
Vanilla Sundae (Boat)
      3 scoops of Vanilla ice cream @ $0.69/scoop
      Hot Fudge topping @ $1.29:                        $3.36           [Tax: $0.24]
Oatmeal Raisin Cookies (Box)
      2 cookies @ $3.45/dozen:                          $0.58           [Tax: $0.04]
------------------------------------------------------------------------------------
Total number of items in order: 7
Order Subtotals:                                        $9.12           [Tax: $0.66]
Order Total:                                                                $9.78
------------------------------------------------------------------------------------
Paid with CARD
```

Correctness

From your terminal, run `ruff check` on all of your .py source files and tesst files, like:

```
ruff check dessert.py
```

This will check for syntax errors, violations and many issues that could lead to bugs in your code. Code will be maually graded, so any score received are partial.

Style

From your terminal, run `ruff format` on all of your .py source files above to check the format of your code, like:

```
ruff format dessert.py
```

How to Submit

From your GitHub assignment repository page, click Submit and enter a nontrivial commit message.

Grading

This project is manually graded. Use the following rubric.

Criteria	Mastery (100)	Developing (85)	Beginning (70)	Low (50)
Implementation of Payable Interface	The Payable Interface is correctly implemented with the `get_pay_type()` and `set_pay_type(payment_method: PayType)` methods. Raises ValueError correctly.	The Payable Interface is implemented but with minor errors in methods or error handling.	The Payable Interface is partially implemented with significant errors in methods or error handling.	The Payable Interface is either not implemented or incorrectly implemented.
Changes to Order class	The Order class correctly implements the Payment Interface. Default value for pay method is correctly set. `__str__` method correctly modified to include payment type.	The Order class implements the Payment Interface, but there are minor errors in handling default value, interface implementation or modifying `__str__` method.	The Order class attempts to implement the Payment Interface, but with significant errors or omissions.	The Order class does not implement the Payment Interface or has serious implementation errors.
Changes to DessertShop Class	The DessertShop class correctly adds a method that prompts for user input, validates it, and returns the payment type.	The DessertShop class adds the required method but with minor errors in prompting user input or validation.	The DessertShop class attempts to add the required method but with significant errors or omissions.	The DessertShop class does not add the required method or has serious errors.
Test Cases	All required test cases are implemented correctly and pass.	Most required test cases are implemented and pass, but there may be a few missing or failing tests.	Some required test cases are implemented and pass, but there are significant gaps in test coverage.	Few or no required test cases are implemented, or most tests fail.

Criteria	Mastery (100)	Developing (85)	Beginning (70)	Low (50)
Receipt Output	The receipt correctly includes payment information for each item.	The receipt includes payment information for most items, but there may be a few errors or omissions.	The receipt includes payment information for some items, but there are significant errors or omissions.	The receipt does not include correct payment information for the items or has serious errors.
Use of Payable Interface	Payable Interface is implemented and used correctly across the system.	Payable Interface is mostly implemented and used correctly, but there are minor errors or omissions.	Payable Interface is used in some parts of the system, but there are significant errors or omissions.	Payable Interface is either not used or incorrectly used across the system.

Students should aim for the Mastery level in all categories to ensure they have fully understood and implemented the concepts covered in the project. The overall project score is the average of the individual criteria scores.