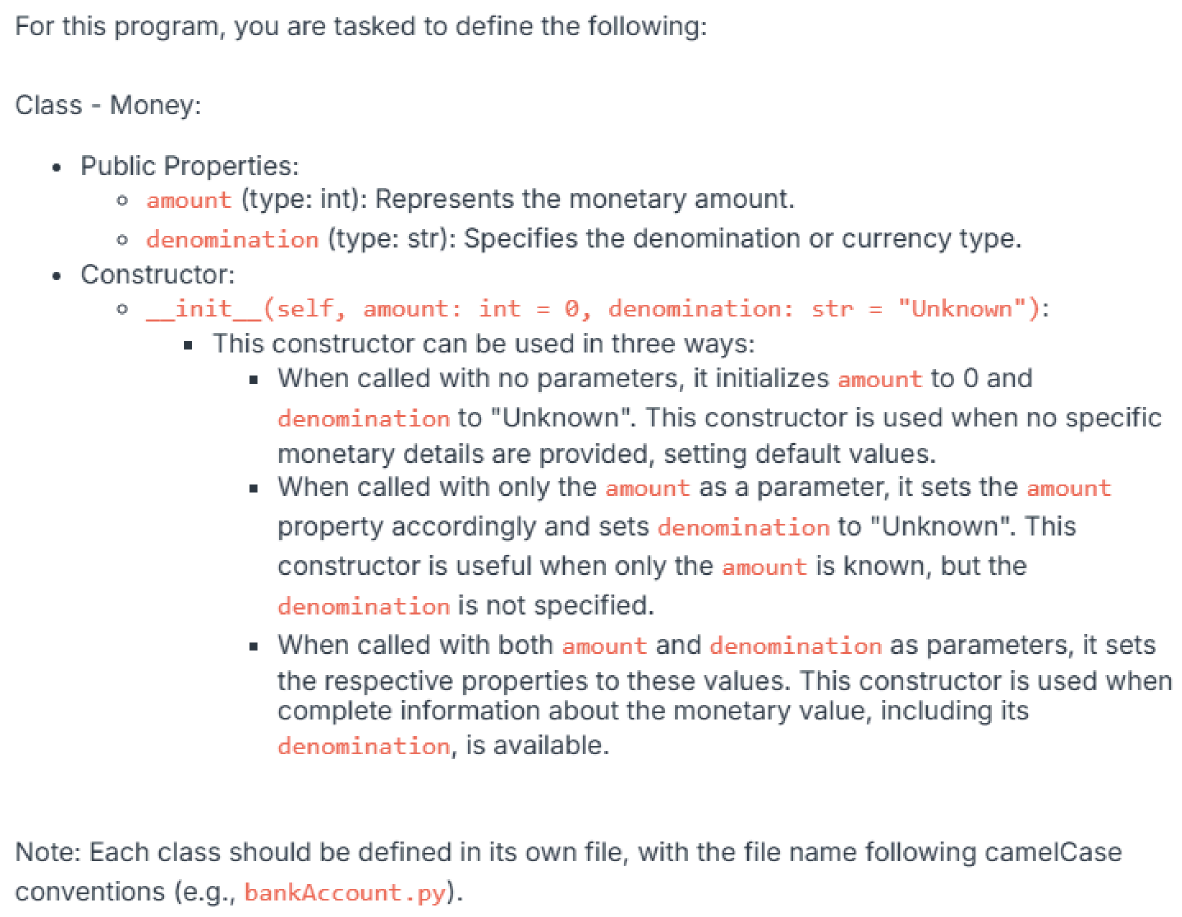
Midterm Lab Task 6.
Constructor Activity

Problem 1.



For this program, you are tasked to define the following:

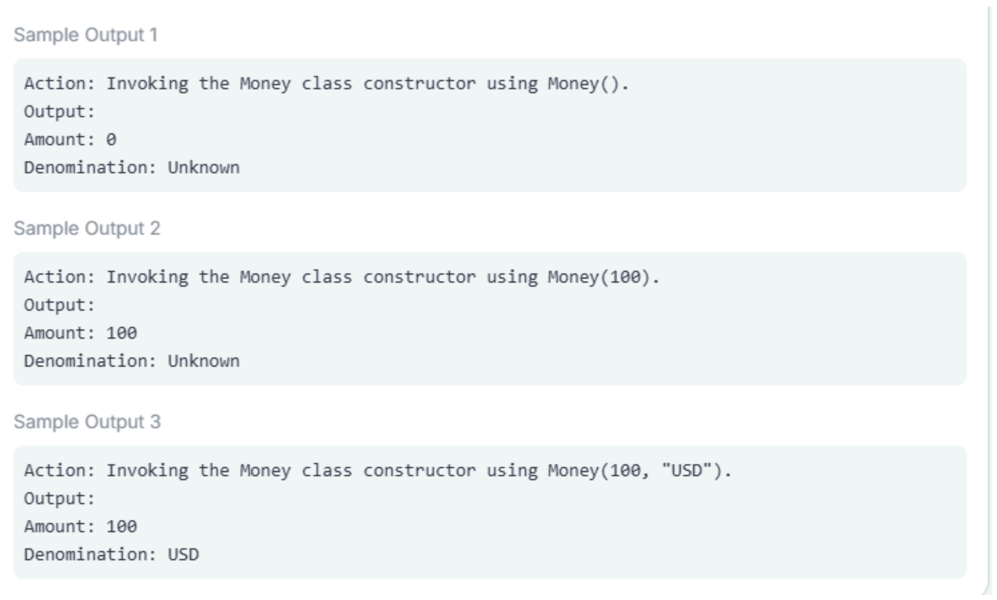
Class - Money:

- Public Properties:
    - `amount` (type: int): Represents the monetary amount.
    - `denomination` (type: str): Specifies the denomination or currency type.
- Constructor:
    - `__init__(self, amount: int = 0, denomination: str = "Unknown")`:
        - This constructor can be used in three ways:
            - When called with no parameters, it initializes `amount` to 0 and `denomination` to "Unknown". This constructor is used when no specific monetary details are provided, setting default values.
            - When called with only the `amount` as a parameter, it sets the `amount` property accordingly and sets `denomination` to "Unknown". This constructor is useful when only the `amount` is known, but the `denomination` is not specified.
            - When called with both `amount` and `denomination` as parameters, it sets the respective properties to these values. This constructor is used when complete information about the monetary value, including its `denomination`, is available.

Note: Each class should be defined in its own file, with the file name following camelCase conventions (e.g., `bankAccount.py`).

Create a test class on a separate file named **testMoney.py**

Then try the sample output below:



Sample Output 1

```
Action: Invoking the Money class constructor using Money().
Output:
Amount: 0
Denomination: Unknown
```

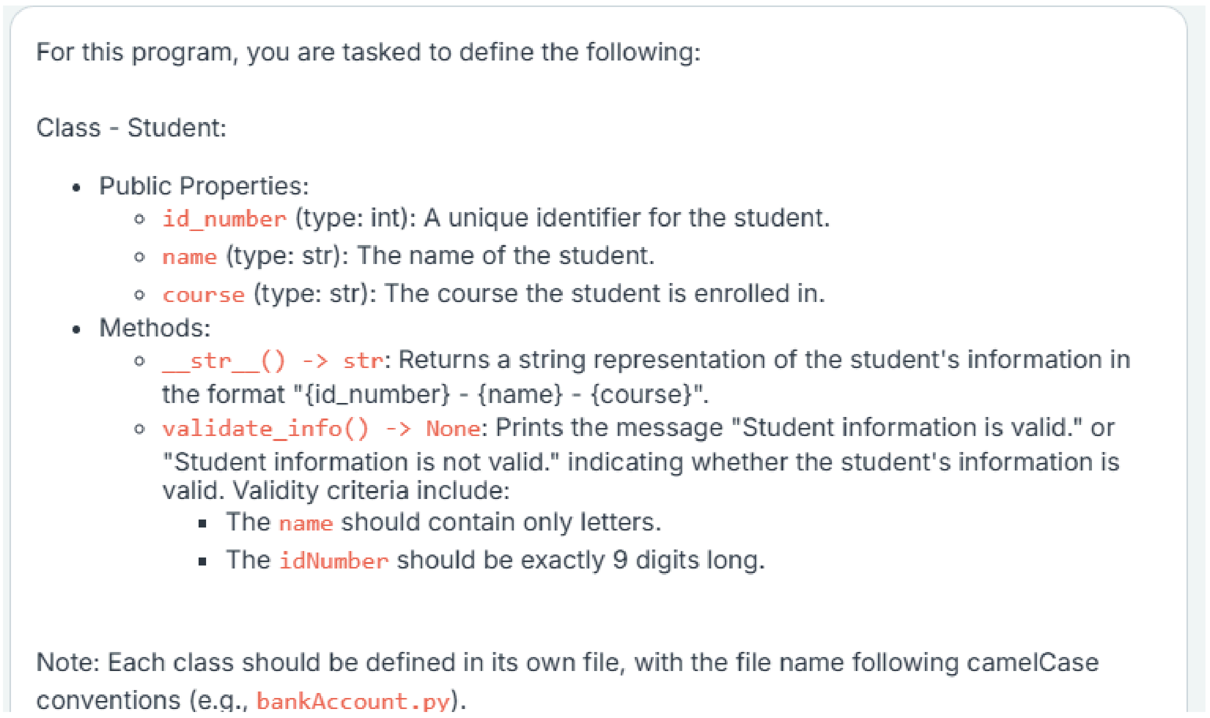
Sample Output 2

```
Action: Invoking the Money class constructor using Money(100).
Output:
Amount: 100
Denomination: Unknown
```

Sample Output 3

```
Action: Invoking the Money class constructor using Money(100, "USD").
Output:
Amount: 100
Denomination: USD
```

Problem 2.



For this program, you are tasked to define the following:

Class - Student:

- Public Properties:
  - `id_number` (type: int): A unique identifier for the student.
  - `name` (type: str): The name of the student.
  - `course` (type: str): The course the student is enrolled in.
- Methods:
  - `__str__() -> str`: Returns a string representation of the student's information in the format "{id_number} - {name} - {course}".
  - `validate_info() -> None`: Prints the message "Student information is valid." or "Student information is not valid." indicating whether the student's information is valid. Validity criteria include:
    - The `name` should contain only letters.
    - The `idNumber` should be exactly 9 digits long.

Note: Each class should be defined in its own file, with the file name following camelCase conventions (e.g., `bankAccount.py`).

Create a test class on a separate file named **testStudent.py**

Sample Output 1

```
Action: Invoking __str__() method with the following Student information:
ID: 123456789
Name: John Doe
Course: Computer Science

Output:
123456789 - John Doe - Computer Science
```

Sample Output 2

```
Action: Invoking __str__() method with the following Student information:
ID: 12345
Name: Jane Doe
Course: Mathematics

Output:
12345 - Jane Doe - Mathematics
```

Sample Output 3

```
Action: Invoking validate_info() method with the following Student information:
ID: 987654321
Name: Alice123
Course: Physics

Output:
Student information is not valid.
```