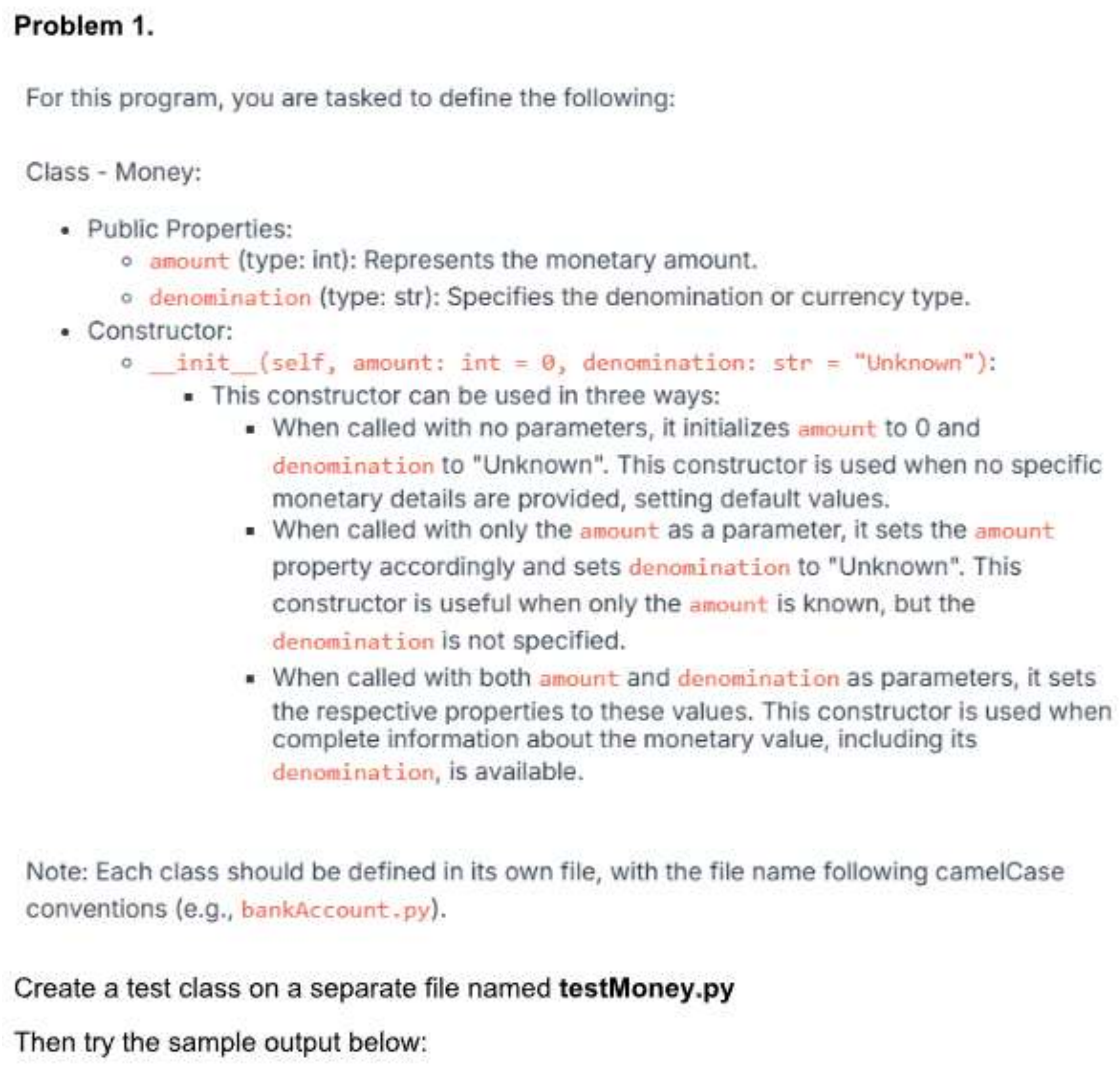
Serrano, Mark Angelo
C203

7OOP1



**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**

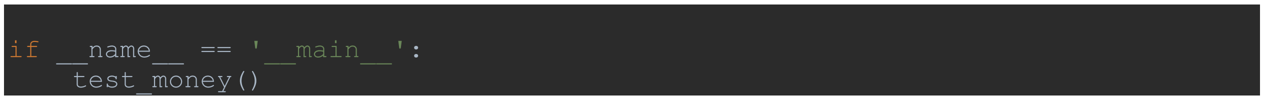
Then try the sample output below:

Problem 1:

```
class Money:

    def __init__(self, amount: int = 0, denomination: str = "Unknown"):
        self.amount = amount
        self.denomination = denomination

    def __str__(self):
        return (f"Action: Invoking the Money class constructor using
Money().\n"
                f"Output:\n"
                f"Amount: {self.amount}\n"
                f"Denomination: {self.denomination}")

def test_money():
    t = Money(0,"Unknown")
    print(t)
```

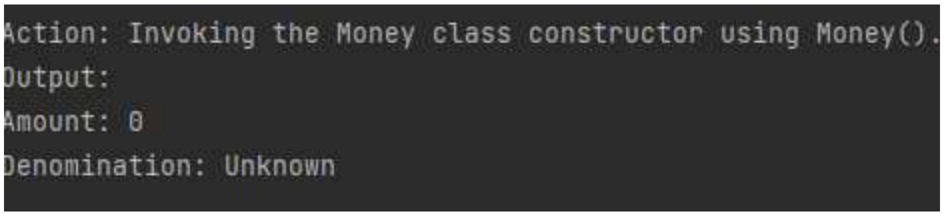
Serrano, Mark Angelo
C203

7OOP1



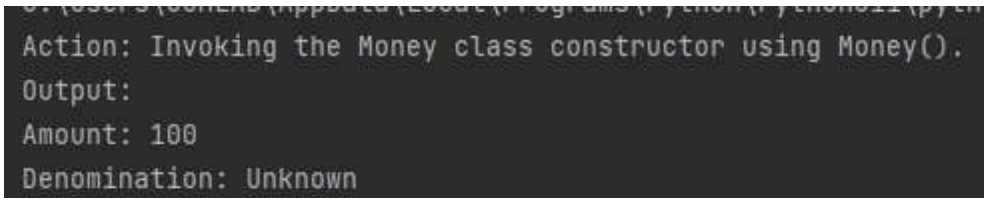
```
if __name__ == '__main__':
    test_money()
```

Sample 1:



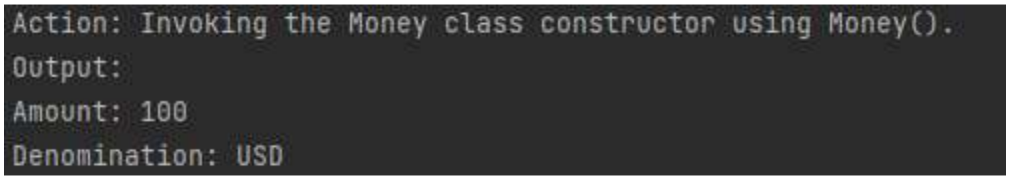
```
Action: Invoking the Money class constructor using Money().
Output:
Amount: 0
Denomination: Unknown
```

Sample 2:



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

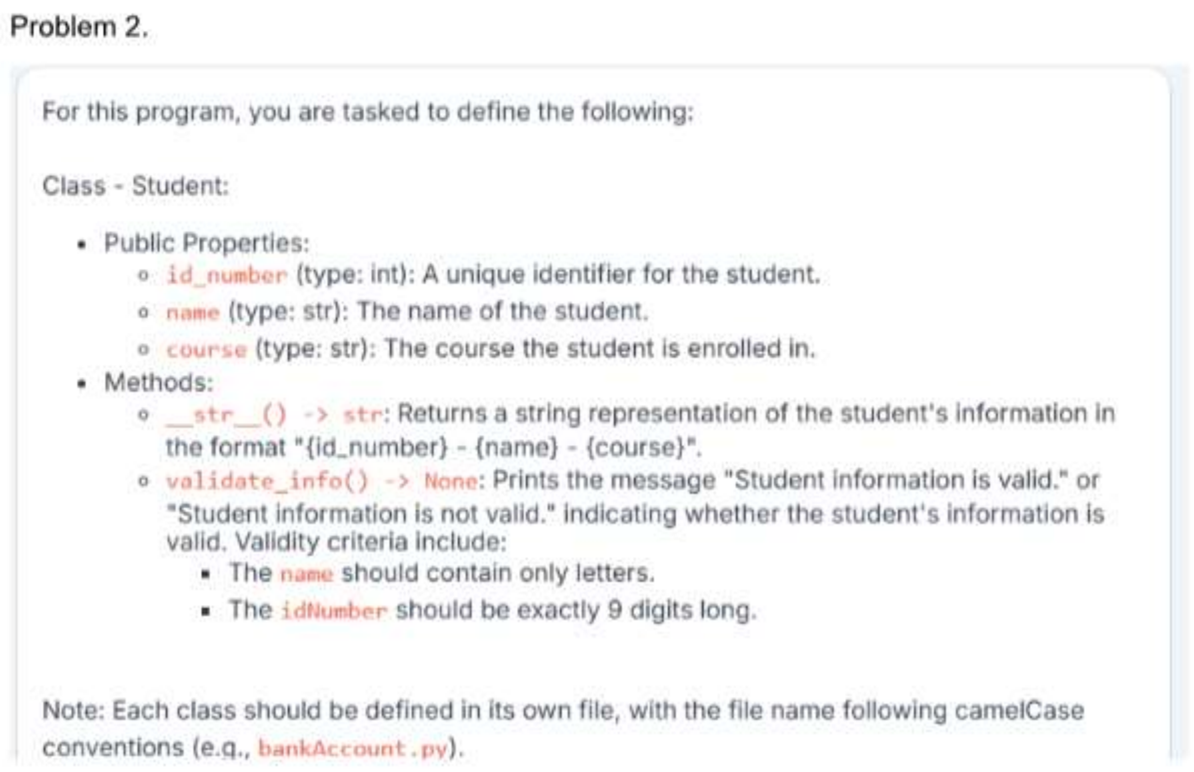
Sample 3:



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

Serrano, Mark Angelo
C203

7OOP1



## 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).

Problem 2:

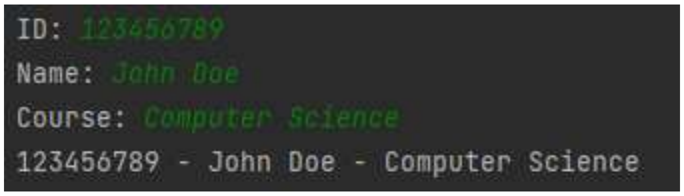
```
class Student:

    def __init__(self, id: int = 0, name: str = "Unknown", course: str =
"Unknown", length: str = 0):
        self.id = id
        self.name = name
        self.course = course
        self.length = length

    def __str__(self):
        if self.id == str:
            self.validate_info()
        elif self.length > 9:
            self.validate_info()
        elif self.name == int:
            self.validate_info()
        elif self.course == int:
            self.validate_info()
        else:
            return (f"{self.id} - {self.name} - {self.course}")

    def validate_info(self):
        print("Student information is not valid.")

def test_student():
    id = int(input("ID: "))
    name = input("Name: ")
    course = input("Course: ")
```

Serrano, Mark Angelo
C203

7OOP1

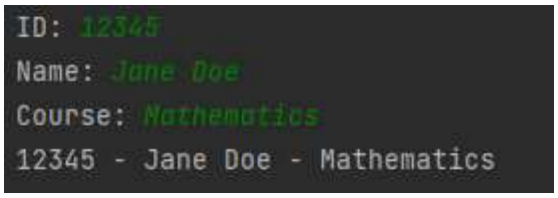
```
    length = len(str(id))
    stud = Student(id,name,course,length)
    print(stud)



if __name__ == '__main__':
    test_student()
```

Sample 1:



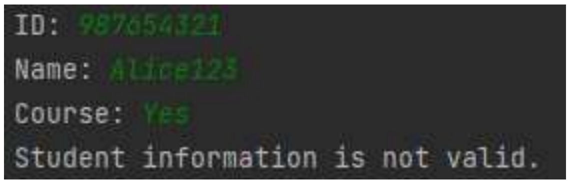
```
ID: 123456789
Name: John Doe
Course: Computer Science
123456789 - John Doe - Computer Science
```

Sample 2:



```
ID: 12345
Name: Jane Doe
Course: Mathematics
12345 - Jane Doe - Mathematics
```

Sample 3:



```
ID: 987654321
Name: Alice123
Course: Yes
Student information is not valid.
```