

Zabala, Rhaldynyl Brian F.

C203

Problem 1

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
```

Code:

```
main.py Money.py testMoney.py +
1 class money:
2
3     def __init__(self, amount: int = 0, denomination: str = "Unknown"):
4         self.amount = amount
5         self.denomination = denomination
6
7     def __str__(self):
8         return f"Amount is: {self.amount}\nDenomination: {self.denomination}"
9
10    def __getitem__(self, key):
11        if key == 0:
12            return f"{self.amount}"
13        elif key == 1:
14            return f"{self.denomination}"
15        else:
16            return ""
```

```
main.py Money.py testMoney.py +
1 from Money import money
2
3
4
5 def test(self):
6     self.wallet = money()
7     self.wallet2 = money(100)
8     self.wallet3 = money(100, "USD")
9
10    print("Action: Invoking the Money class constructor using Money()\nOutput:")
11    print(self.wallet)
12    print("\nAction: Invoking the Money class constructor using Money(100)\nOutput:")
13    print(self.wallet2)
14    print("\nAction: Invoking the Money class constructor using Money(100, 'USD')\nOutput:")
15    print(self.wallet3)
16
17
18 if __name__ == '__main__':
19     test(money)
```

Output:

```
Run Share Command Line Arguments
Action: Invoking the Money class constructor using Money()
Output:
Amount is: 0
Denomination: Unknown
> Action: Invoking the Money class constructor using Money(100)
Output:
Amount is: 100
Denomination: Unknown
Action: Invoking the Money class constructor using Money(100, "USD")
Output:
Amount is: 100
Denomination: USD
** Process exited - Return Code: 0 **
```

Problem 2

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.
```

Code

```
student.py x testStudent.py
1 class Student: 4 usages
2     def __init__(self, id_number: int = 0, name: str = "Unknown", course: str = "Unknown"):
3         self.id_number = id_number
4         self.name = name
5         self.course = course
6
7     def __str__(self):
8         return f"{self.id_number} - {self.name} - {self.course}"
9
10    def validate_info(self): 1 usage
11
12        if self.name.replace(__old: " ", __new: "").isalpha() and len(str(self.id_number)) == 9:
13            print("Student information is valid.")
14        else:
15            print("Student information is not valid.")
```

```
student.py x testStudent.py x
1 from student import Student
2
3
4    def main(): 1 usage
5        print("Action: Invoking __str__() method with the following Student information:")
6
7        s1 = Student(id_number: 123456789, name: "John Doe", course: "Computer Science")
8
9        print("ID:", s1.id_number)
10
11        print("Name:", s1.name)
12
13        print("Course:", s1.course)
14
15        print("\nOutput:")
16
17        print(s1)
18
19        print()
20
21        print("Action: Invoking __str__() method with the following Student information:")
22
23        s2 = Student(id_number: 12345, name: "Jane Doe", course: "Mathematics")
24
25        print("ID:", s2.id_number)
26
27        print("Name:", s2.name)
28
29        print("Course:", s2.course)
30
31        print("\nOutput:")
32
33        print(s2)
34
35        print()
```

```

36
37     print("Action: Invoking validate_info() method with the following Student information:")
38
39     s3 = Student(id_number=987654321, name="Alice123", course="Physics")
40
41     print("ID:", s3.id_number)
42
43     print("Name:", s3.name)
44
45     print("Course:", s3.course)
46
47     print("\nOutput:")
48
49     s3.validate_info()
50
51     print()
52
53
54 ▶ if __name__ == '__main__':
55     main()
56

```

Output:

```

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

Output:
123456789 - John Doe - Computer Science

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

Output:
12345 - Jane Doe - Mathematics

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

Output:
Student information is not valid.

Process finished with exit code 0

```

