DATA 602 Assignment Number 4

John Ferrara

Q1: Create a class called BankAccount that has four attributes: bankname, firstname, lastname, and balance.

The default balance should be set to 0.

In addition, create ...

- A method called deposit() that allows the user to make deposits into their balance.
- A method called withdrawal() that allows the user to withdraw from their balance.
- Withdrawal may not exceed the available balance. Hint: consider a conditional argument in your withdrawal() method.
- Use the **str**() method in order to display the bank name, owner name, and current balance.
- Make a series of deposits and withdrawals to test your class.

In [ ]:
```
class BankAccount:
    def __init__(self,bankname, firstname, lastname, balance):
        self.bankname = bankname
        self.firstname = firstname
        self.lastname = lastname
        self.balance = balance

    def __str__(self):
        return f"BankAccount(bankname={self.bankname}, firstname={self.firstname},

    def deposit(self, deposit_amount):
        print(f"Current Balance: ${self.balance}")
        print(f"Deposit ammount: ${deposit_amount}")
        self.balance = self.balance + deposit_amount
        print(f"Depsit successful! New balance is ${self.balance}.")

    def withdrawal(self, withdrawal_amount):
        print(f"Current Balance: ${self.balance}")
        print(f"Withdrawal Request Amount: ${withdrawal_amount}")
        if self.balance >= withdrawal_amount:
            self.balance = self.balance - withdrawal_amount
            print(f"Withdrawal Request Successful! New Account balance is: ${self.b
        else:
            print("Withdrawal Request Denied due to insufficient funds.")
```

Executing / Testing Code

In [2]:
```
### INitializaing
BA = BankAccount("Community Bank","John", "Smith", 500)
##Testing __str__
print(BA)
#Method Testing
BA.deposit(100)
print('--')
BA.withdrawal(500)
print('--')
BA.withdrawal(400)
print('--')
BA.deposit(100)
```

```
BankAccount(bankname=Community Bank, firstname=John, lastname=Smith, balance=500)
Current Balance: $500
Deposit ammount: $100
Depsit successful! New balance is $600.
--
Current Balance: $600
Withdrawal Request Amount: $500
Withdrawal Request Successful! New Account balance is: $100
--
Current Balance: $100
Withdrawal Request Amount: $400
Withdrawal Request Denied due to insufficient funds.
--
Current Balance: $100
Deposit ammount: $100
Depsit successful! New balance is $200.
```

Q2: Create a class Box that has attributes length and width that takes values for length and width upon construction (instantiation via the constructor).

In addition, create

- A method called render() that prints out to the screen a box made with asterisks of length and width dimensions
- A method called invert() that switches length and width with each other
- Methods get_area() and get_perimeter() that return appropriate geometric calculations
- A method called double() that doubles the size of the box. Hint: Pay attention to return value here.
- Implement **eq** so that two boxes can be compared using ==. Two boxes are equal if their respective lengths and widths are identical.
- A method print_dim() that prints to screen the length and width details of the box
- A method get_dim() that returns a tuple containing the length and width of the box
- A method combine() that takes another box as an argument and increases the length and width by the dimensions of the box passed in
- A method get_hypot() that finds the length of the diagonal that cuts through the middle
- Instantiate 3 boxes of dimensions 5,10 , 3,4 and 5,10 and assign to variables box1, box2 and box3 respectively
- Print dimension info for each using print_dim()
- Evaluate if box1 == box2, and also evaluate if box1 == box3, print True or False to the screen accordingly
- Combine box3 into box1 (i.e. box1.combine())
- Double the size of box2
- Combine box2 into box1

In [ ]:

```
class Box:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def __eq__(self, otherbox):
        if self.width == otherbox.width and self.length == otherbox.length:
            return True
        else:
            return False

    def render(self):
        for i in range(self.length):
            print("*" * self.width)

    def invert(self):
        print(f"Inverting dimensions initial dimensions where length is {self.lengt
        self.width_invert = self.length
        self.length_invert = self.width
        self.width = self.width_invert
        self.length = self.length_invert
        print(f"Dimenstions successfully inverted. Length is {self.length}, width i

    def get_area(self):
        self.area = self.length * self.width
        print(f"The area of this box is {self.area}.")
        return self.area

    def get_permeter(self):
        self.perimeter = (self.length*2) + (self.width *2)
        print(f"The perimeter of this box is {self.perimeter}.")
        return self.perimeter

    def double(self):
        print("Doubling the box width and length!")
        ## Assuming we are just consistently updating the class dimensions.
        self.length = self.length*2
        self.width = self.width*2
        print("The length and widths of the box have been doubled!")


    def print_dim(self):
        print(f"The dimensions of this box are a {self.width} width and a {self.len

    def get_dim(self):
        return (self.length, self.width)

    def combine(self, otherbox):
        self.length = self.length + otherbox.length
        self.width = self.width + otherbox.width

    def get_hypot(self):
        hypotenuse = sqrt(self.width**2+self.length**2)
        print(f"The diagonal line in this box is: {hypotenuse}")
```

Action Items for Code

In [4]:
```
## USE-CASE ACTION ITEMS
box1 = Box(5,10)
box1.print_dim()
box2 = Box(3,4)
box2.print_dim()
box3 = Box(5,10)
box3.print_dim()

print("Is box1 equal to box2")
print(box1==box2)
print("Is box1 equal to box3")
print(box1==box3)

## Box 3 into Box1
print("Box1 Original Dimensions")
box1.print_dim()
box1.combine(box3)
print("Box1 Combined Dimensions")
box1.print_dim()

## Double box 2
print("Box2 Original Dimensions")
box2.print_dim()
box2.double()
print("Box2 with Doubled Dimensions")
box2.print_dim()

## Box2 into Box 1
print("Box1 Original Dimensions")
box1.print_dim()
box1.combine(box2)
print("Box1 Combined Dimensions")
box1.print_dim()
```

```
The dimensions of this box are a 10 width and a 5 length.
The dimensions of this box are a 4 width and a 3 length.
The dimensions of this box are a 10 width and a 5 length.
Is box1 equal to box2
False
Is box1 equal to box3
True
Box1 Original Dimensions
The dimensions of this box are a 10 width and a 5 length.
Box1 Combined Dimensions
The dimensions of this box are a 20 width and a 10 length.
Box2 Original Dimensions
The dimensions of this box are a 4 width and a 3 length.
Doubling the box width and length!
The length and widths of the box have been doubled!
Box2 with Doubled Dimensions
The dimensions of this box are a 8 width and a 6 length.
Box1 Original Dimensions
The dimensions of this box are a 20 width and a 10 length.
Box1 Combined Dimensions
The dimensions of this box are a 28 width and a 16 length.
```