

# WEEK 7 - ASSIGNMENT 2

## OOPS CONCEPTS

### NOTE:

- No need to submit anywhere, just keep track of all the PDF you made in a specific folder.
- Compare your solution with the solution I'll provide, in case of doubts, kindly reach out to me.
- You may get assignment solution in format of PDF or VIDEO solution, depending on the difficulty level.

### MINI PROJECT 1

#### Class Implementation for Student Information

##### Objective:

Your task is to create a Python class to manage student information. The class should have attributes for the student's name, age, gender, and a list of marks. Additionally, you need to implement various methods to manipulate and display this information.

##### Class Structure:

Your class should be named Student and should contain the following attributes:

**name (str):** Name of the student.

**age (int):** Age of the student.

**gender (str):** Gender of the student ('Male', 'Female', or 'Other').

**marks (list of int):** List of marks obtained by the student.

##### Methods to Implement:

- **displayAllInfo():** This method should print all information about the student, including name, age, gender, and marks.
- **displayOnlyMarks():** This method should print only the marks obtained by the student.

- **getTotal():** This method should return the total marks obtained by the student.
- **getMax():** This method should return the maximum marks obtained by the student.
- **getMini():** This method should return the minimum marks obtained by the student.
- **getAvg():** This method should return the average marks obtained by the student.
- **addMark(mark):** This method should add a new mark to the list of marks for the student.
- **removeMark():** This method should remove the last mark from the list of marks for the student.

### Instructions:

- Implement the Student class according to the provided specifications.
- Instantiate the Student class with using a **\_\_init\_\_() method**. Ask the details and 5 marks within **\_\_init\_\_()**.
- Test your class by creating instances of Student and calling various methods on them to ensure they work correctly.
- Use meaningful variable names and include comments where necessary to enhance code readability.

### EXAMPLE USAGE

```
# Creating a student object
student1 = Student()

# Display all information about the student
student1.displayAllInfo()

# Display only the marks obtained by the student
student1.displayOnlyMarks()

# Get total marks obtained by the student
total_marks = student1.getTotal()
print("Total Marks:", total_marks)

# Get maximum marks obtained by the student
max_marks = student1.getMax()
print("Maximum Marks:", max_marks)

# Get minimum marks obtained by the student
min_marks = student1.getMini()
print("Minimum Marks:", min_marks)

# Get average marks obtained by the student
avg_marks = student1.getAvg()
print("Average Marks:", avg_marks)

# Add a new mark to the list
student1.addMark(87)

# Remove the last mark from the list
student1.removeMark()
```

## Class Implementation for Bank Management

### Objective:

Your task is to create a Python class to manage bank accounts. The class should have attributes for the account holder's name, account number, balance, and account type. Additionally, you need to implement various methods to manipulate and display this information.

## Class Structure:

Your class should be named `Bank` and should contain the following attributes:

- **account\_holder\_name (str):** Name of the account holder.
- **account\_number (int):** Account number of the account holder.
- **balance (float):** Current balance in the account.
- **account\_type (str):** Type of account ('Savings', 'Checking', etc.).

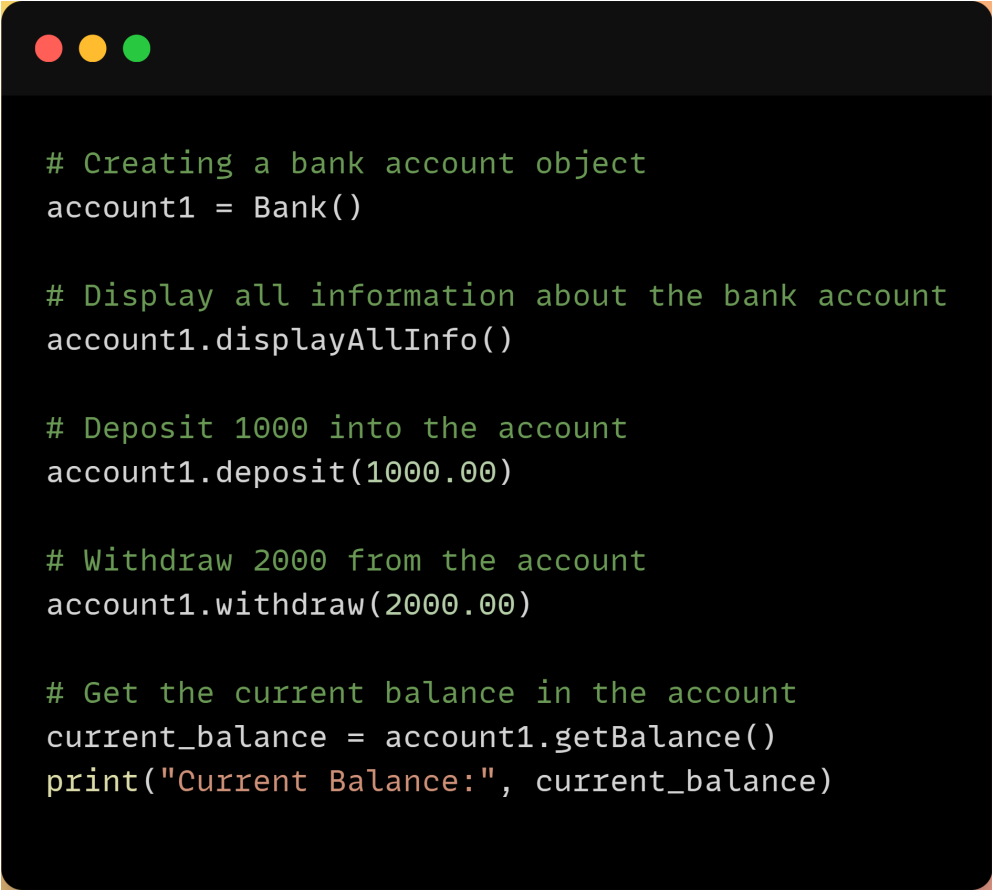
## Methods to Implement:

- **displayAllInfo():** This method should print all information about the bank account, including the account holder's name, account number, balance, and account type.
- **deposit(amount):** This method should add the specified amount to the account balance.
- **withdraw(amount):** This method should deduct the specified amount from the account balance if sufficient funds are available.
- **getBalance():** This method should return the current balance in the account.

## Instructions:

- Keep in mind you just need to ask **account\_holder\_name** and **account\_type** from the user in **\_\_init\_\_()** method. The **account\_number** should be randomly generated between 100000 to 999999 and **balance** should automatically set to **100**.
- Implement the `Bank` class according to the provided specifications.
- Test your class by creating instances of `Bank` and calling various methods on them to ensure they work correctly.
- Use meaningful variable names and include comments where necessary to enhance code readability.

## EXAMPLE USAGE



```
# Creating a bank account object
account1 = Bank()

# Display all information about the bank account
account1.displayAllInfo()

# Deposit 1000 into the account
account1.deposit(1000.00)

# Withdraw 2000 from the account
account1.withdraw(2000.00)

# Get the current balance in the account
current_balance = account1.getBalance()
print("Current Balance:", current_balance)
```