# Abbottabad University Of Science And

**Technology**

# Department Of Computer Science



ASSIGNMENT NO 4

## Name : M Hassan Ashraf

**Roll No : 2023132**

## Semester : 2nd

**Subject : OOP**

## Submitted To : Sir Jamal Abdul Ahad

**Q1: Write a program in python Create a BankAccount class with public attributes for account\_number and balance. Add private attributes for \_\_owner\_name and \_\_pin. Implement methods to get and set these attributes securely**

class BankAccount:

def \_\_init\_\_(self, account\_number, balance, owner\_name, pin):

self.account\_number = account\_number

self.balance = balance

self.\_\_owner\_name = owner\_name

self.\_\_pin = pin

# Getter method for owner\_name

def get\_owner\_name(self):

return self.\_\_owner\_name

# Setter method for owner\_name with validation

def set\_owner\_name(self, new\_owner\_name):

if isinstance(new\_owner\_name, str):

self.\_\_owner\_name = new\_owner\_name

print("Owner name has been updated.")

else:

print("Invalid input. Owner name must be a string.")

# Getter method for pin

def get\_pin(self):

return "\*\*\*\*" # For security reasons, only return a masked version of the PIN

# Setter method for pin with validation

def set\_pin(self, new\_pin):

if isinstance(new\_pin, int) and 1000 <= new\_pin <= 9999:

self.\_\_pin = new\_pin

print("PIN has been updated.")

else:

print("Invalid input. PIN must be a 4-digit integer.")

# Example usage:

account = BankAccount(account\_number=123456789, balance=1000, owner\_name="John Doe", pin=1234)

# Accessing public attributes

print("Account Number:", account.account\_number)

print("Balance:", account.balance)

# Accessing private attributes using getter methods

print("Owner Name:", account.get\_owner\_name())

print("PIN:", account.get\_pin())

# Updating private attributes using setter methods

account.set\_owner\_name("Jane Doe")

account.set\_pin(5678)

# Checking updated attributes

print("Updated Owner Name:", account.get\_owner\_name())

print("Updated PIN:", account.get\_pin())

Q2: Design a UserProfile in python class with public attributes for username and email. Add private attributes for \_\_password and \_\_security\_question. Implement methods to get and set these attributes securely.

class UserProfile:

def \_\_init\_\_(self, username, email, password, security\_question):

self.username = username

self.email = email

self.\_\_password = password

self.\_\_security\_question = security\_question

# Getter method for password

def get\_password(self):

return "\*\*\*\*" # For security reasons, only return a masked version of the password

# Setter method for password with validation

def set\_password(self, new\_password):

if isinstance(new\_password, str) and len(new\_password) >= 8:

self.\_\_password = new\_password

print("Password has been updated.")

else:

print("Invalid input. Password must be a string with at least 8 characters.")

# Getter method for security question

def get\_security\_question(self):

return self.\_\_security\_question

# Setter method for security question with validation

def set\_security\_question(self, new\_security\_question):

if isinstance(new\_security\_question, str) and len(new\_security\_question) > 0:

self.\_\_security\_question = new\_security\_question

print("Security question has been updated.")

else:

print("Invalid input. Security question must be a non-empty string.")

# Example usage:

user\_profile = UserProfile(username="john\_doe", email="john@example.com", password="secure\_password", security\_question="Favorite color?")

# Accessing public attributes

print("Username:", user\_profile.username)

print("Email:", user\_profile.email)

# Accessing private attributes using getter methods

print("Password:", user\_profile.get\_password())

print("Security Question:", user\_profile.get\_security\_question())

# Updating private attributes using setter methods

user\_profile.set\_password("new\_secure\_password")

user\_profile.set\_security\_question("Favorite animal?")

# Checking updated attributes

print("Updated Password:", user\_profile.get\_password())

print("Updated Security Question:", user\_profile.get\_security\_question())

Q3: Create an online quiz system with classes like Quiz, Question, and User. Allow users to take quizzes, track their scores, and provide feedback on their performance.

class Question:

def \_\_init\_\_(self, text, options, correct\_option):

self.text = text

self.options = options

self.correct\_option = correct\_option

def is\_correct(self, user\_answer):

return user\_answer == self.correct\_option

class Quiz:

def \_\_init\_\_(self, name, questions):

self.name = name

self.questions = questions

def take\_quiz(self, user):

score = 0

user\_answers = {}

print(f"\n--- {self.name} Quiz ---")

for index, question in enumerate(self.questions, start=1):

print(f"\nQuestion {index}: {question.text}")

for i, option in enumerate(question.options, start=1):

print(f"{i}. {option}")

user\_answer = int(input("Your answer (enter the number of your choice): "))

user\_answers[question.text] = user\_answer

if question.is\_correct(user\_answer):

score += 1

user.record\_quiz\_result(self.name, score, user\_answers)

print(f"\nQuiz completed! Your score: {score}/{len(self.questions)}")

user.provide\_feedback(self.name, score)

class User:

def \_\_init\_\_(self, username):

self.username = username

self.quiz\_results = {}

def record\_quiz\_result(self, quiz\_name, score, user\_answers):

self.quiz\_results[quiz\_name] = {"score": score, "user\_answers": user\_answers}

def provide\_feedback(self, quiz\_name, score):

if score == len(quizzes[quiz\_name].questions):

print("Congratulations! You answered all questions correctly.")

elif score >= len(quizzes[quiz\_name].questions) / 2:

print("Good job! You passed the quiz.")

else:

print("You need to review the quiz. Try again for a better score.")

# Sample questions

question1 = Question("What is the capital of France?", ["Paris", "Berlin", "Rome", "Madrid"], 1)

question2 = Question("Which planet is known as the Red Planet?", ["Earth", "Mars", "Jupiter", "Venus"], 2)

question3 = Question("Who wrote 'Romeo and Juliet'?", ["William Shakespeare", "Jane Austen", "Charles Dickens", "Mark Twain"], 1)

# Sample quiz

quiz1 = Quiz("General Knowledge", [question1, question2, question3])

# Sample user

user1 = User("JohnDoe")

# Running the quiz

quizzes = {"General Knowledge": quiz1}

quiz\_name\_to\_take = "General Knowledge"

quizzes[quiz\_name\_to\_take].take\_quiz(user1)