Class 08 Assignment - May 05, 2025

✅ Assignment:

One Dollar ($1) Challenge (2nd Week)

Create a unique application by applying OOP principles. It could be anything from a Web App to a CLI app or anything in between.

If there are multiple qualified submissions, there will be two winners: one male and one female.

Simple Budget Planner is my choice because its smart, helpful, and a perfect OOP-based app for my class 08 assignment.

```python
#  simple_budget_planner.py

# A CLI-based Budget Planner app using OOP principles in Python.
# You can add incomes, expenses, and view your savings summary.

# Base class to represent any transaction (income or expense)
class Transaction:
    def __init__(self, amount, description):
        self.amount = amount
        self.description = description

# Subclass for income
class Income(Transaction):
    pass  # Inherits everything from Transaction

# Subclass for expense
class Expense(Transaction):
    pass  # Inherits everything from Transaction

# Manager class to handle budget-related operations
class BudgetPlanner:
    def __init__(self):
        self.incomes = []   # List to store income objects
        self.expenses = []  # List to store expense objects
```

```
    # Add a new income
    def add_income(self, amount, description):
        self.incomes.append(Income(amount, description))
        print(f"✅ Income added: {description} - ${amount:.2f}")

    # Add a new expense
    def add_expense(self, amount, description):
        self.expenses.append(Expense(amount, description))
        print(f"❌ Expense added: {description} - ${amount:.2f}")

    # Calculate total income
    def total_income(self):
        return sum(item.amount for item in self.incomes)

    # Calculate total expense
    def total_expense(self):
        return sum(item.amount for item in self.expenses)

    # Calculate remaining balance
    def balance(self):
        return self.total_income() - self.total_expense()

    # Show a simple summary
    def show_summary(self):
        print("\n--- 💼 Budget Summary ---")
        print(f"Total Income:   ${self.total_income():.2f}")
        print(f"Total Expenses: ${self.total_expense():.2f}")
        print(f"Net Savings:    ${self.balance():.2f}")
        print("--------------------------")


# Main function to run the app in CLI
def main():
    planner = BudgetPlanner()

    while True:
        # Display menu options
        print("\n--- Budget Planner ---")
        print("1. Add Income\n2. Add Expense\n3. View Summary\n4. Exit")
        choice = input("Select (1-4): ")

        # User choices and validations
        if choice == "1":
            try:
                amt = float(input("Income amount: $"))
                desc = input("Description: ")
                planner.add_income(amt, desc)
            except ValueError:
                print("⚠️ Enter a valid number.")
        elif choice == "2":
```

```
            try:
                amt = float(input("Expense amount: $"))
                desc = input("Description: ")
                planner.add_expense(amt, desc)
            except ValueError:
                print("⚠ Enter a valid number.")
        elif choice == "3":
            planner.show_summary()
        elif choice == "4":
            print("👋 Exiting. Stay financially smart!")
            break
        else:
            print("⚠ Invalid option. Choose 1 to 4.")


# Entry point check
if __name__ == "__main__":
    main()
```