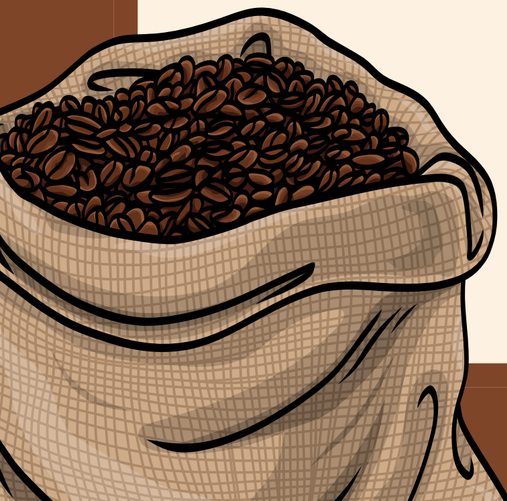
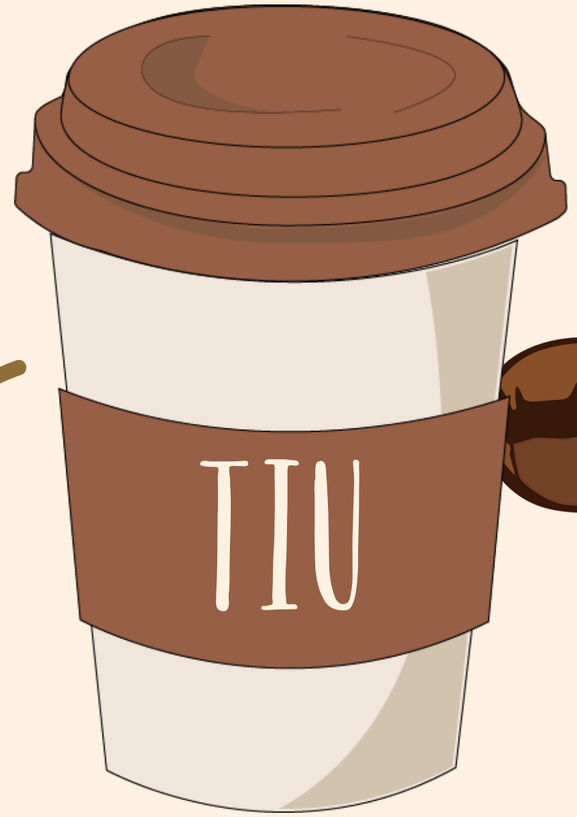
TIU
Coffee Express

A BILLING SYSTEM BY YESHANI BHAGYA





PROBLEM

MANUAL ORDERING AND BILLING ARE SLOW AND ERROR-PRONE DURING BUSY EVENTS. THERE IS NO AUTOMATED RECORD-KEEPING.

OBJECTIVE 1

IMPLEMENT A FLEXIBLE, MENU-BASED ORDERING SYSTEM USING DICTIONARIES.

OBJECTIVE 2

PERFORM ACCURATE FINANCIAL CALCULATIONS (SUBTOTAL, OPTIONAL DISCOUNT, AND FINAL TOTAL).

OBJECTIVE 3

GENERATE A FORMATTED BILL RECEIPT AND SAVE A PERMANENT RECORD TO A FILE.

CLASSES & OBJECTS

THE COFFEEORDER CLASS IS OUR ORDER MANAGER. EVERY TIME A NEW CUSTOMER STARTS AN ORDER, WE CREATE A NEW OBJECT FROM THIS CLASS. THIS ENSURES THAT EACH BILL IS HANDLED COMPLETELY SEPARATELY, KEEPING OUR DATA CLEAN AND ORGANIZED.

THE CONSTRUCTOR

THE __INIT__ METHOD IS LIKE THE CASHIER OPENING A NEW TICKET. IT RUNS AUTOMATICALLY WHEN AN ORDER BEGINS, SETTING UP ALL THE STARTING ATTRIBUTES (VARIABLES) LIKE THE EMPTY ITEMS LIST AND THE SUBTOTAL TO ZERO. THIS MAKES SURE EVERY ORDER STARTS FRESH.

```
import datetime
import random
MENU = {
  "1": {"item": "Espresso", "price": 2.50},
  "2": {"item": "Latte", "price": 4.00},
  "3": {"item": "Cappuccino", "price": 4.50},
  "4": {"item": "Muffin", "price": 3.00},
  "5": {"item": "Croissant", "price": 2.00}
}
PROMO_CODE = "TIUSTUDENT"
DISCOUNT_RATE = 0.15
class CoffeeOrder:
  def __init__(self, customer_name):
    self.customer_name = customer_name
    self.items = []
    self.subtotal = 0.0
    self.discount_applied = 0.0
    self.final_total = 0.0
    self.receipt_id = random.randint(1000, 9999)

def add_item(self, item_code, quantity):
    item_data = MENU.get(item_code)

    if item_data:
      price = item_data["price"]
      item_name = item_data["item"]
      line_total = price * quantity

      self.items.append({
        "name": item_name,
        "qty": quantity,
        "price": price,
        "line_total": line_total
      })

      self.subtotal += line_total
      print(f"Added {quantity}x {item_name} to the order. Current subtotal: ${self.subtotal:.2f}")
    else:
      print(" Invalid item code. Please choose from the menu.")
```

METHODS

I MOVED ALL THE IMPORTANT ACTIONS—LIKE CALCULATE_TOTAL(), ADD_ITEM(), AND APPLY_DISCOUNT()—INSIDE THE CLASS AS SPECIALIZED FUNCTIONS CALLED METHODS. THEY ALL USE THE SPECIAL SELF ARGUMENT TO ACCESS AND UPDATE THAT SPECIFIC ORDER'S DATA.

```
def apply_discount_prompt(order_object):
    has_code = get_valid_input("Do you have a promo code? (Y/N): ", str)
    if has_code == 'Y':
     code = input("Enter your promo code: ").strip().upper()
     order_object.apply_discount(code)
    else:
     print("No discount applied.")
def main():
  print("=" * 50)
  print("WELCOME TO THE TIU COFFEE EXPRESS BILLING SYSTEM")
  print("=" * 50)
  customer_name = input("Enter Customer Name/ID: ").strip()
  current_order = CoffeeOrder(customer_name)
  display_menu(MENU)
  take_order(current_order)

  if not current_order.items:
    print("\nOrder was empty. Exiting system.")
    return
  apply_discount_prompt(current_order)
  final_total = current_order.calculate_total()
  receipt_output = current_order.generate_receipt()
  print("\n" + "=" * 50)
  print("     ")
  print(receipt_output)
  current_order.save_receipt(receipt_output)
if __name__ == "__main__":
  main()
```

DATA ENCAPSULATION

BY PUTTING THE DATA (SELF.ITEMS) AND THE LOGIC (CALCULATE_TOTAL()) TOGETHER IN ONE CLASS, I CREATE A SECURE, SELF-CONTAINED UNIT. THIS MAKES THE ENTIRE BILLING SYSTEM RELIABLE AND EASY TO MAINTAIN.

```
DEF APPLY_DISCOUNT_PROMPT(ORDER_OBJECT):
    HAS_CODE = GET_VALID_INPUT("DO YOU HAVE A PROMO CODE? (Y/N): ", STR)
    IF HAS_CODE == 'Y':
      CODE = INPUT("ENTER YOUR PROMO CODE: ").STRIP().UPPER()
      ORDER_OBJECT.APPLY_DISCOUNT(CODE)
    ELSE:
      PRINT("NO DISCOUNT APPLIED.")
DEF MAIN():
  PRINT("=" * 50)
  PRINT("WELCOME TO THE TIU COFFEE EXPRESS BILLING SYSTEM")
  PRINT("=" * 50)
  CUSTOMER_NAME = INPUT("ENTER CUSTOMER NAME/ID: ").STRIP()
  CURRENT_ORDER = COFFEEORDER(CUSTOMER_NAME)
  DISPLAY_MENU(MENU)
  TAKE_ORDER(CURRENT_ORDER)

  IF NOT CURRENT_ORDER.ITEMS:
    PRINT("\N👋 ORDER WAS EMPTY. EXITING SYSTEM.")
    RETURN
  APPLY_DISCOUNT_PROMPT(CURRENT_ORDER)
  FINAL_TOTAL = CURRENT_ORDER.CALCULATE_TOTAL()
  RECEIPT_OUTPUT = CURRENT_ORDER.GENERATE_RECEIPT()
  PRINT("\N" + "=" * 50)
  PRINT("     ")
  PRINT(RECEIPT_OUTPUT)
  CURRENT_ORDER.SAVE_RECEIPT(RECEIPT_OUTPUT)
IF __NAME__ == "__MAIN__":
  MAIN()
```

IMPLEMENTATION & DOCUMENTATION

CODE FLOW

THE PROGRAM STARTS IN THE MAIN() FUNCTION, WHICH INITIALIZES THE MENU AND CALLS THE PRIMARY INTERACTION FUNCTIONS

INPUT VALIDATION

ENSURES USER INPUT FOR ITEM ID AND QUANTITY IS VALID (E.G., HANDLING NON-NUMERIC INPUT).

FILE I/O

FILE HANDLING IS DONE SECURELY USING THE WITH OPEN(...) CONSTRUCT, ENSURING THE FILE IS ALWAYS CLOSED, AND USING THE APPEND MODE ('A') TO SAVE EVERY NEW TRANSACTION

DOCUMENTATION

MY CODE IS WELL-COMMENTED THROUGHOUT, EXPLAINING COMPLEX CALCULATIONS AND THE PURPOSE OF EACH FUNCTION, ENHANCING READABILITY.

FINAL PRODUCT

WITH DISCOUNT

```
==========================================


==========================================
              TIU COFFEE EXPRESS
==========================================
RECEIPT ID: 5962 | DATE: 2025-12-04 02:36:38
CUSTOMER: YESH
------------------------------------------
ITEM              QTY   PRICE
------------------------------------------
ESPRESSO           3    7.50
------------------------------------------
SUBTOTAL:                $   7.50
DISCOUNT APPLIED:        -$  1.12
==========================================
TOTAL DUE:               $   6.38
==========================================
THANK YOU FOR SUPPORTING TIU EVENTS!
==========================================
```

WITH OUT DISCOUNT

```
==========================================


==========================================
              TIU COFFEE EXPRESS
==========================================
RECEIPT ID: 2278 | DATE: 2025-12-04 02:41:53
CUSTOMER: YESH
------------------------------------------
ITEM              QTY   PRICE
------------------------------------------
CROISSANT          2    4.00
LATTE              1    4.00
------------------------------------------
SUBTOTAL:                $   8.00
TOTAL DUE:               $   8.00
==========================================
THANK YOU FOR SUPPORTING TIU EVENTS!
==========================================
```

THANK YOU

Make sure to graB some coffee