Fitnesscenter.py & main.py – 2 files (Main is importing fitnesscenter.py)

**Flow Diagram of Add\_member function in FitnessCenter class**

class FitnessCenter:  
 def \_\_init\_\_(self):  
 self.members = []  
 self.clubs = [Club("Club A", "Address A"), Club("Club B", "Address B"), Club("Club C", "Address C"),  
 Club("Club D", "Address D")]  
 self.promotion\_start\_date = datetime.date(2023, 4, 1)  
 self.promotion\_end\_date = datetime.date(2023, 4, 30)  
  
 def add\_member(self):  
 while True:  
 name = input("Enter name of member: ")  
 if not name:  
 print("Name cannot be empty. Please try again.")  
 continue  
 break  
  
 while True:  
 membership\_type = input("Enter membership type (Single/Multi): ").lower()  
 if membership\_type not in ("single", "multi"):  
 print("Invalid membership type. Please enter 'single' or 'multi'.")  
 continue  
 break  
  
 member\_id = len(self.members) + 1  
 if membership\_type == "single":  
 club = self.choose\_club()  
 member = SingleClubMember(member\_id, name, club)  
 else:  
 member = MultiClubMember(member\_id, name)  
 if self.is\_promotion\_period():  
 print("You have signed up during the promotion period!")  
 if isinstance(member, SingleClubMember):  
 cost = 40  
 else:  
 cost = 80  
 print(f"You get discount of: ${cost}")  
 else:  
 print(f"{name} has been added as a {membership\_type}-club member with ID {member\_id}.")  
 if isinstance(member, SingleClubMember):  
 cost = 50  
 else:  
 cost = 100  
 print(f"You get discount of: ${cost}")  
 self.members.append(member)

| Start |

|

v

| Enter name |

| of member |

|

v

| Check if name |

| is not empty |

|

+--------------|--------------+

| Yes | | No

+---------------+ | +----------------+

| Continue loop | | | Print error msg |

+---------------+ | +----------------+

|

v

+------------------+

| Enter membership |

| type |

+------------------+

|

v

+-----------------+

| Check if |

| valid type |

+-----------------+

|

+--------------|--------------+

| Yes | | No

+---------------+ | +------------------------+

| Continue loop | | | Print invalid type error |

+---------------+ | +------------------------+

|

v

+-----------------+

| Generate member |

| ID |

+-----------------+

|

v

+-----------------+

| Choose club (if |

| single member) |

+-----------------+

|

v

+-----------------------+

| Create member instance |

+-----------------------+

|

v

+-----------------------+

| Check if promotion |

| period is active |

+-----------------------+

|

+--------------|--------------+

| Yes | | No

+---------------+ | +-------------------------+

| Calculate cost | | | Print regular cost message |

+---------------+ | +-------------------------+

|

v

+-------------------------+

| Add member to members |

| list and print discount |

| message |

+-------------------------+

|

v

+---------+

| End |

+---------+

Flow of promotion\_period function

def is\_promotion\_period(self):  
 today = datetime.date.today()  
 return self.promotion\_start\_date <= today <= self.promotion\_end\_date

+---------+

| Start |

+---------+

|

v

+------------------------+

| Set today = today's date|

+------------------------+

|

v

+------------------------------------------------------+

| return promotion\_start\_date <= today <= promotion\_end\_date|

+------------------------------------------------------+

|

v

+------------------------------+ +------------------------------+

| If True, go to "Output: True"| | If False, go to "Output: False"|

+------------------------------+ +------------------------------+

| |

v v

+--------------+ +--------------+

| Output: True | | Output: False|

+--------------+ +--------------+

| |

v v

+---------+ +---------+

| End | | End |

+---------+ +---------+

Flow Chart of remove\_member function

def remove\_member(self):  
 while True:  
 try:  
 member\_id = int(input("Enter ID of member to remove: "))  
 if member\_id <= 0:  
 raise ValueError  
 break  
 except ValueError:  
 print("Invalid member ID. Please enter a positive integer.")  
 for member in self.members:  
 if member.member\_id == member\_id:  
 self.members.remove(member)  
 print(f"Member with ID {member\_id} has been removed.")  
 return  
 print(f"No member found with ID {member\_id}.")

+---------+

| Start |

+---------+

|

v

+----------------------------------+

| Loop: while True |

| (until valid member ID is entered)|

+----------------------------------+

|

v

+-----------------------------+

| Try to get member ID input |

| as an integer |

+-----------------------------+

|

v

+----------------------------------+

| If successful, check if ID is |

| a positive integer |

+----------------------------------+

|

v

+-----------------------------+ +------------------------------+

| If ID is valid, break loop | | If ID is invalid, raise ValueError|

+-----------------------------+ +------------------------------+

| |

v v

+------------------------------+ +----------------------------------+

| For each member in self.members | | Except block: handle ValueError |

+------------------------------+ +----------------------------------+

|

v

+---------------------------------------+

| If member.member\_id == member\_id, |

| remove the member from self.members |

| and return "Member removed" message |

+---------------------------------------+

|

v

+--------------------------------------------------+

| If no member found with the given ID, return an |

| error message stating "No member found with ID X" |

+--------------------------------------------------+

|

v

+------------+

| End |

|  |
| --- |
|  |

Flow char of display\_memeber\_info method

def display\_member\_info(self):  
  
 while True:  
 try:  
 member\_id = int(input("Enter ID of member to display: "))  
 if member\_id <= 0:  
 raise ValueError  
 break  
 except ValueError:  
 print("Invalid member ID. Please enter a positive integer.")  
 for member in self.members:  
 if member.member\_id == member\_id:  
 print(f"Member ID: {member.member\_id}")  
 print(f"Name: {member.name}")  
 if isinstance(member, SingleClubMember):  
 print(f"Membership Type: Single-Club")  
 print(f"Club: {member.club.name}")  
 else:  
 print(f"Membership Type: Multi-Club")  
 print(f"Points: {member.membership\_points}")  
 return  
 print(f"No member found with ID {member\_id}.")

+---------+

| Start |

+---------+

|

v

+----------------------------------+

| Loop: while True |

| (until valid member ID is entered)|

+----------------------------------+

|

v

+-----------------------------+

| Try to get member ID input |

| as an integer |

+-----------------------------+

|

v

+----------------------------------+

| If successful, check if ID is |

| a positive integer |

+----------------------------------+

|

v

+-----------------------------+ +------------------------------+

| If ID is valid, break loop | | If ID is invalid, raise ValueError|

+-----------------------------+ +------------------------------+

| |

v v

+------------------------------+ +----------------------------------+

| For each member in self.members | | Except block: handle ValueError |

+------------------------------+ +----------------------------------+

|

v

+---------------------------------------+

| If member.member\_id == member\_id, |

| print member information and return |

| "Member information displayed" message |

+---------------------------------------+

|

v

+--------------------------------------------------+

| If no member found with the given ID, return an |

| error message stating "No member found with ID X" |

+--------------------------------------------------+

|

v

+------------+

| End |

+------------+

Flow chat of choose\_club method

def choose\_club(self):  
 while True:  
 print("Choose a club:")  
 for i, club in enumerate(self.clubs):  
 print(f"{i + 1}. {club.name} - {club.address}")  
 choice = input()  
 try:  
 index = int(choice) - 1  
 if index >= 0 and index < len(self.clubs):  
 return self.clubs[index]  
 else:  
 print("Invalid choice. Please choose again.")  
 except ValueError:  
 print("Invalid choice. Please choose again.")

+---------+

| Start |

+---------+

|

v

+------------------------+

| Loop: while True |

| (until valid choice is made) |

+------------------------+

|

v

+-------------------------------+

| Display club options with index|

+-------------------------------+

|

v

+----------------------------------+

| Try to get user's club choice as |

| an integer |

+----------------------------------+

|

v

+-----------------------------+

| If successful, check if the |

| index is valid |

+-----------------------------+

|

v

+--------------------------------+

| If index is valid, return the |

| club corresponding to the index|

+--------------------------------+

|

v

+------------------------------------+

| If index is invalid, print error |

| message and continue looping |

+------------------------------------+

|

v

+-----------------------------+

| If input is not a valid |

| integer, print error message |

+-----------------------------+

|

v

+------------+

| End |

|  |
| --- |
|  |

Flow chart of check\_in\_memeber method

def check\_in\_member(self):  
 while True:  
 try:  
 member\_id = int(input("Enter ID of member to check in: "))  
 if member\_id <= 0:  
 raise ValueError  
 break  
 except ValueError:  
 print("Invalid member ID. Please enter a positive integer.")  
 for member in self.members:  
 if member.member\_id == member\_id:  
 if isinstance(member, SingleClubMember):  
 club = self.choose\_club()  
 member.check\_in(club)  
 else:  
 club = self.choose\_club()  
 member.check\_in(club)  
 return  
 print(f"No member found with ID {member\_id}.")

+-----------------------------------+

| Start function |

+-----------------------------------+

|

v

+-----------------------------------+

| Enter ID of member to check in |

+-----------------------------------+

|

v

+-----------------------------------+

| Check if ID is valid integer |

+---------+-------+----------------+

|

v

+---------+-------+----------------+

| Invalid ID |

+---------+-------+----------------+

|

v

+-----------------------+

| Prompt user to enter ID |

+-----------------------+

|

v

Loop back to

start of block

|

v

+-----------------------------------+

| Valid ID entered |

+---------+-------+----------------+

| |

v |

+---------+-------+----------------+

| Check if member exists |

+---------+-------+----------------+

|

v

+-----------------------+

| Member exists? |

+-----------------------+

|

v

+-----------------------+

| Member is SingleClubMember? |

+-----------------------+

|

v

+---------+-------+----------------+

| Select club to check in |

+---------+-------+----------------+

|

v

+-----------------------+

| Check in SingleClubMember |

+-----------------------+

|

v

+-----------------------------------+

| End function |

+-----------------------------------+

|

v

+---------+-------+----------------+

| Member is MultiClubMember? |

+-----------------------+

|

v

+---------+-------+----------------+

| Select club to check in |

+---------+-------+----------------+

|

v

+-----------------------+

| Check in MultiClubMember |

+-----------------------+

|

v

+-----------------------------------+

| End function |

+-----------------------------------+

|

v

+-----------------------+

| Member does not exist |

+-----------------------+

|

v

+-----------------------------------+

| End function |

+-----------------------------------+

Flow chart of generate\_bill method

def generate\_bill(self):  
  
 while True:  
 try:  
 member\_id = int(input("Enter ID of member to generate bill: "))  
 if member\_id <= 0:  
 raise ValueError  
 break  
 except ValueError:  
 print("Invalid member ID. Please enter a positive integer.")  
 for member in self.members:  
 if member.member\_id == member\_id:  
 #  
 if self.is\_promotion\_period():  
 if isinstance(member, SingleClubMember):  
 Dicost = 40  
 else:  
 Dicost = 80  
 print(f"You get discount of: ${Dicost}")  
 else:  
 if isinstance(member, SingleClubMember):  
 Dicost = 50  
 else:  
 Dicost = 100  
 print(f"You get discount of: ${Dicost}")  
 #  
 print(f"Member ID: {member.member\_id}")  
 print(f"Name: {member.name}")  
 if isinstance(member, SingleClubMember):  
 print(f"Membership Type: Single-Club")  
 print(f"Club: {member.club.name}")  
 cost = 500 # cost for single-club membership  
 else:  
 print(f"Membership Type: Multi-Club")  
 print(f"Points: {member.membership\_points}")  
 cost = 800 # cost for multi-club membership  
 if member.membership\_points > 10:  
 discount = 0.1 # 10% discount for more than 10 points  
 cost \*= (1 - discount)  
 print(f"You have received a 10% discount for having more than 10 points.")  
 cost =cost -Dicost  
 print(f"Total cost: ${cost}")  
 return  
 print(f"No member found with ID {member\_id}.")

Start

|

v

+---------------+

| Enter member |

| ID to generate|

| bill |

+---------------+

|

v

+-------------------------+

| Validate member ID input |

+-------------------------+

|

v

+-------------------------------+

| Check if promotion period |

| and calculate discount amount|

+-------------------------------+

|

v

+-------------------------+

| Display member's details |

| and membership cost |

+-------------------------+

|

v

+--------+

| End |

|  |
| --- |
|  |

Flow chat of run() method

def run(self):  
 while True:  
 print("What would you like to do?")  
 print("1. Add member")  
 print("2. Remove member")  
 print("3. Display member information")  
 print("4. Check in member")  
 print("5. Generate bill")  
 print("6. Quit")  
 choice = input()  
 if choice == "1":  
 self.add\_member()  
 elif choice == "2":  
 self.remove\_member()  
 elif choice == "3":  
 self.display\_member\_info()  
 elif choice == "4":  
 self.check\_in\_member()  
 elif choice == "5":  
 self.generate\_bill()  
 elif choice == "6":  
 print("Goodbye!")  
 break  
 else:  
 print("Invalid choice. Please choose again.")

+----------------+

| Start Program |

+----------------+

|

v

+----------------+

| Print Options |

+----------------+

|

v

+------------------+

| Get User Input |

+------------------+

|

v

+------------------+

| Choice = 1 |

+------------------+

|

v

+------------------+

| Call add\_member |

+------------------+

|

v

+------------------+

| Go to Start |

+------------------+

|

v

+------------------+

| Choice = 2 |

+------------------+

|

v

+------------------+

| Call remove\_member|

+------------------+

|

v

+------------------+

| Go to Start |

+------------------+

|

v

+------------------+

| Choice = 3 |

+------------------+

|

v

+-----------------------+

| Call display\_member\_info|

+-----------------------+

|

v

+------------------+

| Go to Start |

+------------------+

|

v

+------------------+

| Choice = 4 |

+------------------+

|

v

+------------------+

| Call check\_in\_member |

+------------------+

|

v

+------------------+

| Go to Start |

+------------------+

|

v

+------------------+

| Choice = 5 |

+------------------+

|

v

+------------------+

| Call generate\_bill|

+------------------+

|

v

+------------------+

| Go to Start |

+------------------+

|

v

+------------------+

| Choice = 6 |

+------------------+

|

v

+----------------+

| End |

+----------------+

|

v