FITNESS TRACKER SYSTEM

1. Aim of the Project

The aim of this project is to create a simple fitness tracker system using Python that leverages object-oriented programming (OOP) principles. The system allows users to record their workouts, set fitness goals, and track progress towards these goals. The project demonstrates basic OOP concepts such as classes, methods, and encapsulation.

2. Business Problem or Problem Statement

In today's health-conscious world, individuals often struggle to keep track of their fitness activities and measure progress towards their fitness goals. Existing fitness tracking apps can be complex or too feature-heavy for users who only need basic functionality. This project aims to address this problem by providing a straightforward and user-friendly fitness tracker that helps users log workouts, set specific goals, and monitor their achievements.

3. Project Description

The fitness tracker system consists of two primary classes: `User` and `Workout`.

- **User**: Represents an individual who tracks their fitness activities. It allows users to set fitness goals (e.g., calories to burn) and log workouts.

- **Workout**: Represents an individual workout session, including its type, duration, and calories burned.

Users can interact with the system via a command-line interface to input their name, set fitness goals, log workout details, and view their progress towards the goals. The program handles user input to create and manage `User` and `Workout` objects, calculate progress, and display relevant information.

4. Functionalities

- **Create a User**: Allows users to input their name and initialize a `User` object.

- **Set Fitness Goals**: Users can set goals for metrics like calories burned.
- **Log Workouts**: Users can log workout sessions with details such as type, duration, and calories burned.
- **Track Progress**: Users can view their progress towards their set goals based on their logged workouts.
- **Display Information**: Provides a summary of user details, goals, and workout information.

5. Input Versatility with Error Handling and Exception Handling

To make the system robust, input validation and error handling should be implemented. For example, handle cases where non-integer values are entered for goals or workout details, and ensure valid inputs are processed correctly.

Here is an updated code implementation that includes error handling and improved input versatility:

```
class User:
    def __init__(self, name):
    self.name = name
        self.workouts = []
        self.goals = {}

    def add_workout(self, workout):
        self.workouts.append(workout)

    def set_goal(self, goal_type, target_value):
        self.goals[goal_type] = target_value

    def get_progress(self):
        progress = {}
        for goal_type, target_value in self.goals.items():
            total = sum(workout.get(goal_type, 0) for workout in
self.workouts)
            progress[goal_type] = total
        return progress

    def __str__(self):
        return f"User: {self.name}, Goals: {self.goals}, Workouts:
```

```
{self.workouts}"


class Workout:
    def __init__(self, type, duration, calories_burned):
        self.type = type
        self.duration = duration  # in minutes
        self.calories_burned = calories_burned

    def to_dict(self):
        return {
            'type': self.type,
            'duration': self.duration,
            'calories_burned': self.calories_burned
        }

    def __str__(self):
        return f"Workout(type={self.type},
duration={self.duration}min, calories_burned={self.calories_burned})"


def get_int_input(prompt):
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            print("Invalid input. Please enter an integer value.")

def main():
    # Create a new user
    user_name = input("Enter your name: ")
    user = User(user_name)

    # Set fitness goals
    calories_goal = get_int_input("Set your calorie burn goal: ")
    user.set_goal('calories_burned', calories_goal)

    while True:
        # Add workouts
```

```
        workout_type = input("Enter workout type (or 'done' to
finish): ")
        if workout_type.lower() == 'done':
            break

        duration = get_int_input("Enter workout duration in minutes:
")
        calories_burned = get_int_input("Enter calories burned: ")

        workout = Workout(workout_type, duration, calories_burned)
        user.add_workout(workout.to_dict())

    # Display user information and progress
    print(user)
    progress = user.get_progress()
    print("Progress towards goals:")
    for goal_type, total in progress.items():
        print(f"{goal_type}: {total} / {user.goals.get(goal_type, 'Not
Set')}")

if __name__ == "__main__":
    main()
```

6. Code Implementation

The code provided above is a complete implementation of the fitness tracker system with improved input handling. It includes:

- **Classes and Methods**: `User` and `Workout` classes are defined with relevant methods.
- **Error Handling**: The `get_int_input` function ensures that user inputs are integers, handling errors gracefully.

7. Conclusion

This fitness tracker system provides a basic yet functional approach to tracking workouts and fitness goals using OOP principles in Python. It allows users to log their workouts, set goals, and monitor their progress in a user-friendly manner. With basic error handling, the system ensures robust user interactions. This project can be extended with additional features such as different types of goals, more detailed workout metrics, and integration with data persistence solutions for a more comprehensive fitness tracking experience.