”Student Management System"

Program Overview:

The program is designed to manage a list of students with unique IDs. It separates the students into two groups: those with even IDs and those with odd IDs. The program demonstrates Object-Oriented Programming (OOP) concepts, including Inheritance, Polymorphism, and Encapsulation.

Program Structure:

1. Person class: The base class, representing a person with a name.
2. Student class: The derived class, inheriting from Person, representing a student with a student ID.
3. students_data list: A list of tuples containing student data (ID, name).
4. students_list: A list of Student objects created from students_data.
5. Separation logic: Students are separated into even and odd ID groups.

Important Points:

1. The code demonstrates Inheritance, Polymorphism, and Encapsulation.
2. It creates a list of 10 students with unique IDs and separates them into even and odd ID groups.
3. The Student class inherits from the Person class, overriding the get_role() method.
4. The code uses Encapsulation to hide data (e.g., name, student_id) and provide methods to access it.

Inheritance:

Student class inherits from Person class using class Student(Person):
super().__init__(name) calls the Person class constructor
Student class has access to Person class methods and attributes

Polymorphism:

get_role() method is overridden in Student class
Different behavior for get_role() depending on the object type (Person or Student)

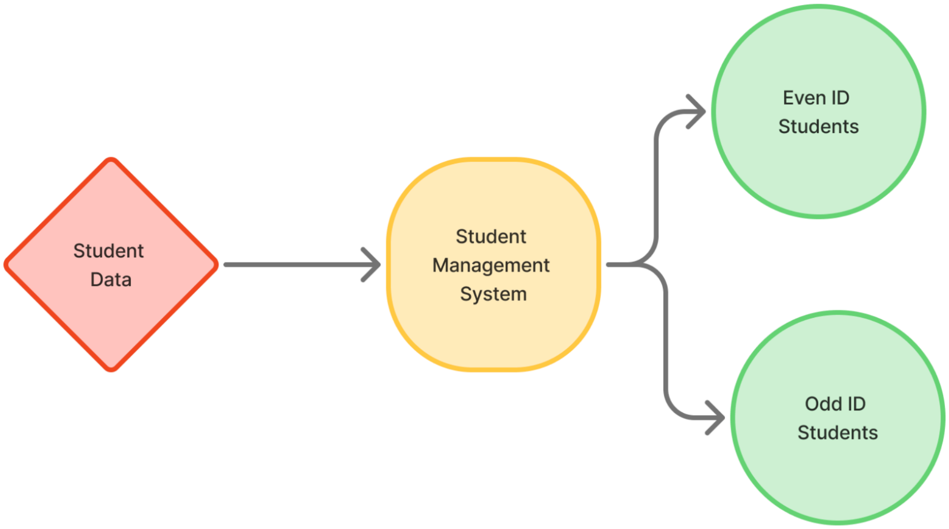
Encapsulation:

Data (name, student_id) is hidden inside the class
Methods (extract_id_number(), display_details()) provide controlled access to data

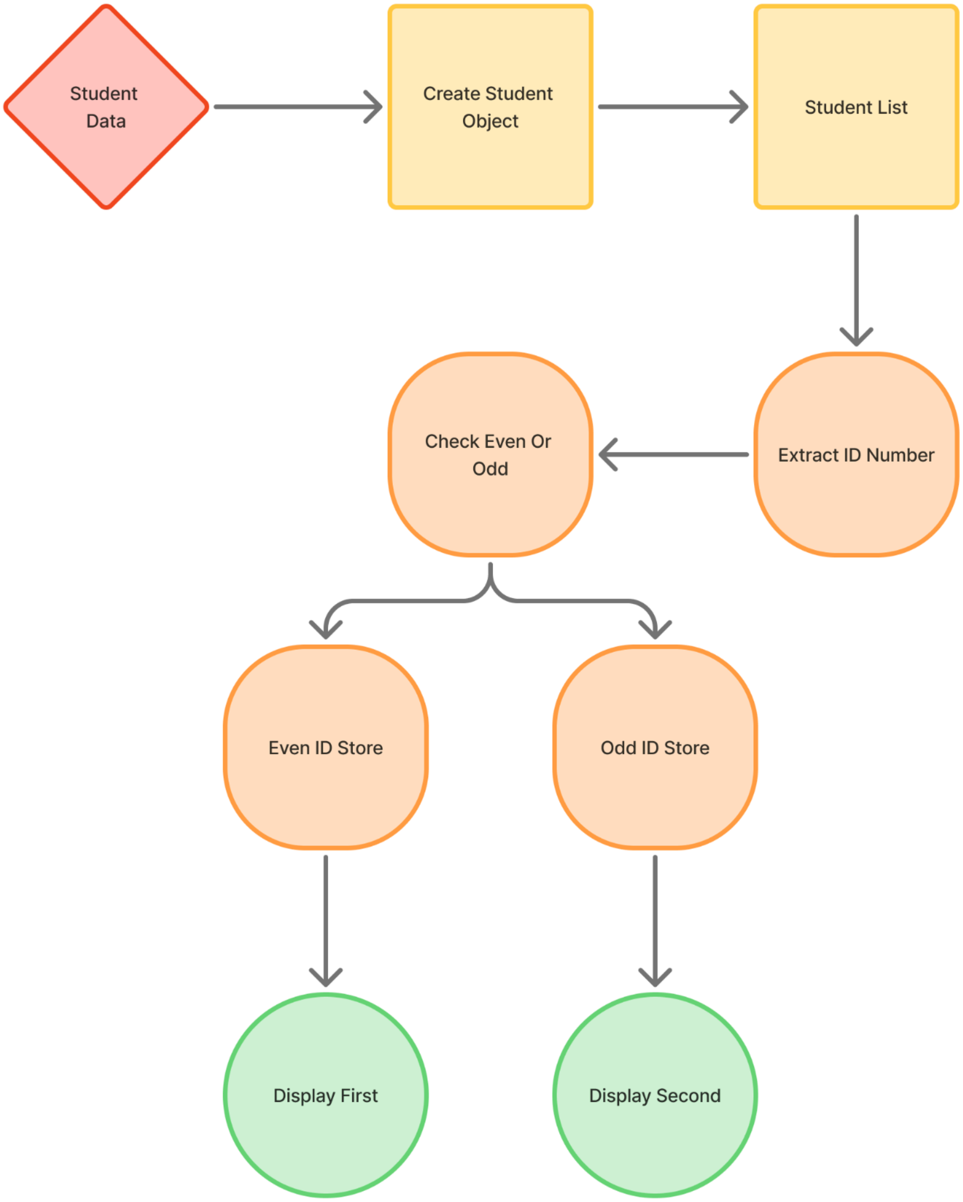
Key Code Snippets:

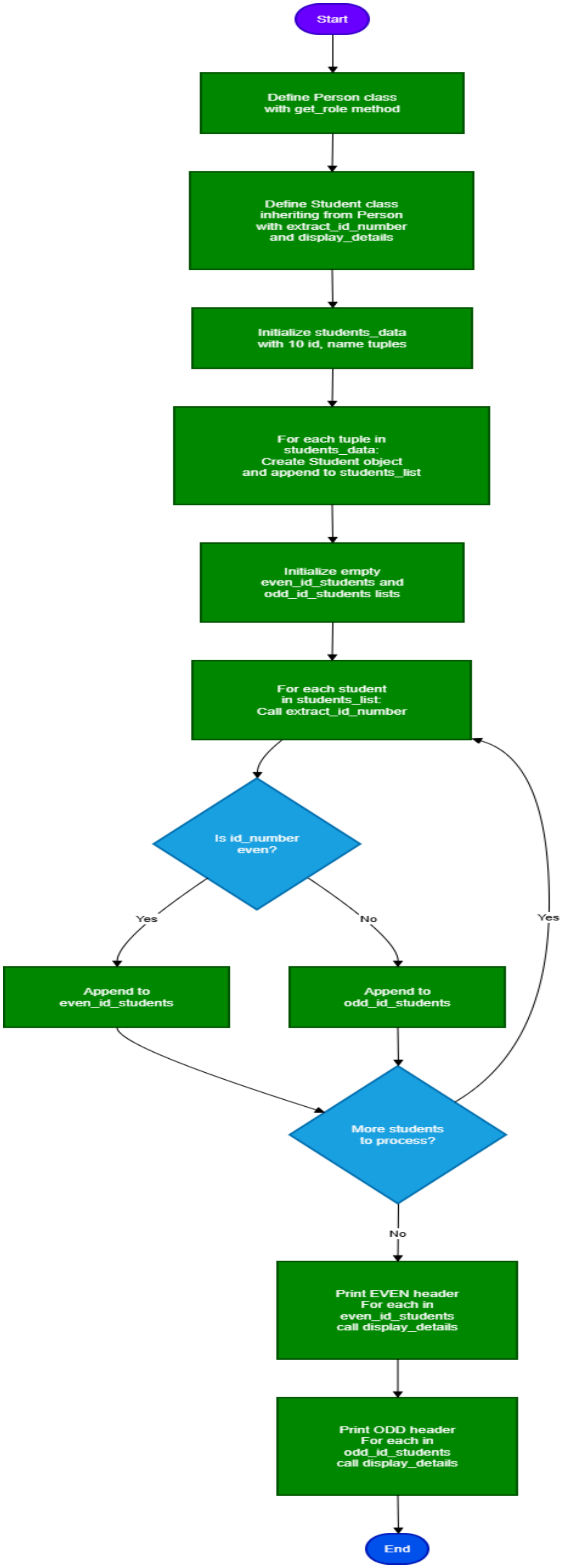
class Student(Person): - Inheritance
super().__init__(name) - Inheritance
def get_role(self): return "I am a Student" - Polymorphism
self.name = name - Encapsulation
id_number = student.extract_id_number() - Encapsulation

Level 0


Student
Data
Student
Management
System
Even ID
Students
Odd ID
Students

Level 1


Student Data
Create Student Object
Student List
Extract ID Number
Check Even Or Odd
Even ID Store
Odd ID Store
Display First
Display Second


Start
Define Person class with get_role method
Define Student class inheriting from Person with extract_id_number and display_details
Initialize students_data with 10 id, name tuples
For each tuple in students_data: Create Student object and append to students_list
Initialize empty even_id_students and odd_id_students lists
For each student in students_list: Call extract_id_number
Is id_number even?
Yes
No
Append to even_id_students
Append to odd_id_students
Yes
More students to process?
No
Print EVEN header For each in even_id_students call display_details
Print ODD header For each in odd_id_students call display_details
End

--FLOWCHART