Project 6 Deliverables

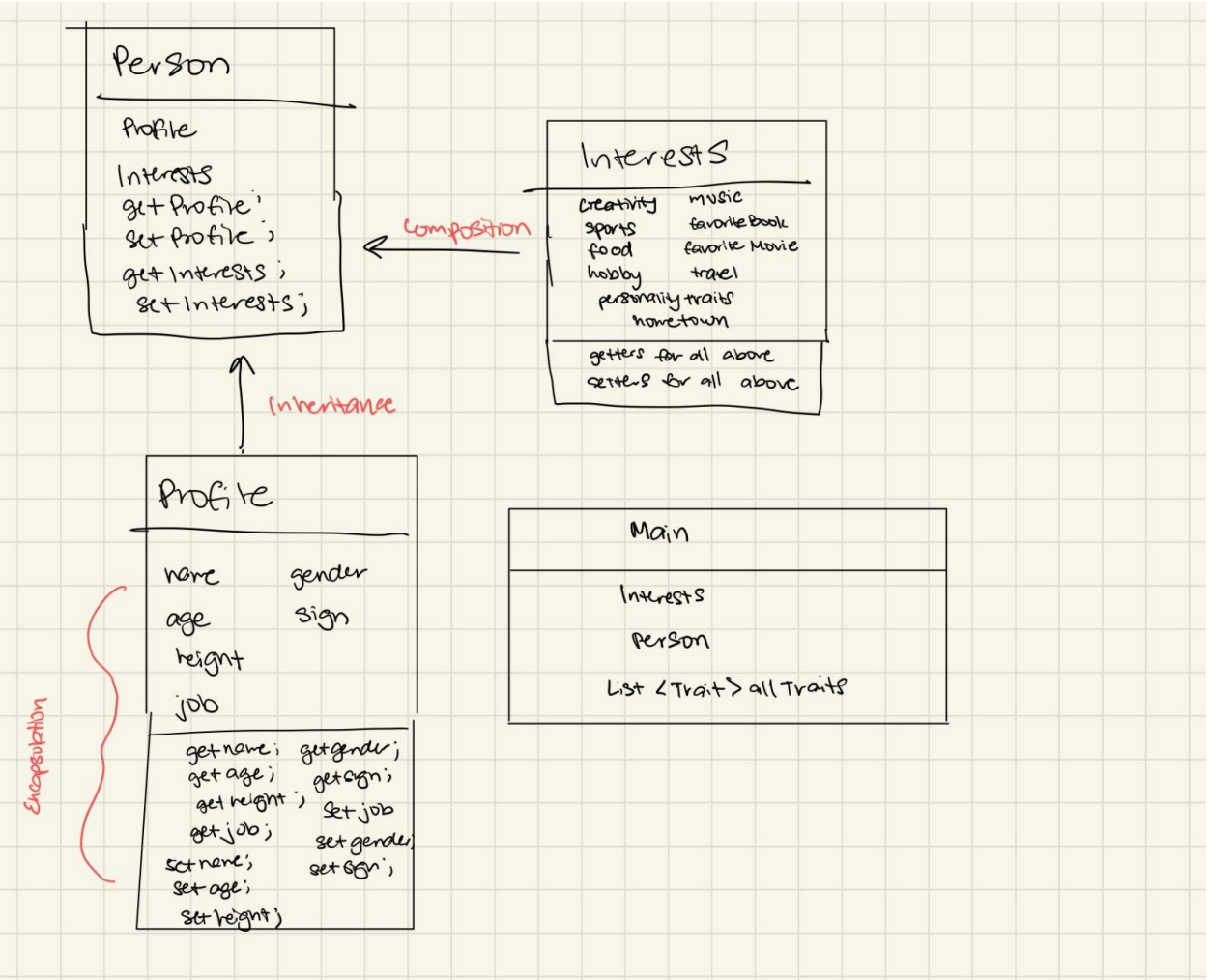
Project's Title: "LoveQuest"
Team Members: Birwa Balar and Abby Haines

Status Summary (15 points):

- **Work Done**
    - For the first week of the project we focused mainly on the person class and implementing the proper code for creating the users profile for the game. Their profile includes basic information and their interests in order to calculate compatibility with other users. Thus we created a Person class which included an instance of the Interests class and the Person class extends the Profile class. Thus Person inherits from Profile. We split this up by having one person focus on the Profile class and another person focus on the Interests class.
- **Changes or Issues Encountered**
    - Some challenges we faced was redundancy with the Profile and Interests classes. Some categories were able to fit in both classes and thus to ensure there is nothing redundant we had to go through each category in both classes to make sure nothing was added twice. Other than that, we didnt have much deviations from the initial Project 5 design, we think most of the changes will come when the compatibility portion of the game is implemented as the algorithm may face challenges.
- **Patterns**
    - 1. Inheritance: The Person class extends the Profile class, meaning it inherits all of its properties and behaviors. This is a classic example of inheritance, where a subclass inherits characteristics from a superclass.
    - 2. Composition: The Person class also uses composition by having an instance of the Interests class as one of its properties. This represents a "has-a" relationship, where a Person has Interests.
    - 3. Encapsulation: By using private fields with public getters and setters in both Profile and Interests classes, our design adheres to the encapsulation principle. This ensures that the internal representation of an object is hidden from the outside, only allowing access through defined interfaces.
    - 4. Single Responsibility Principle (SRP): Each class in our design has a clear and distinct responsibility. Profile handles personal attributes like age and job, while Interests deals with a person's interests. This separation of concerns makes your classes more modular and easier to maintain.
    - 5. Delegation: In the Person class's toString method, we delegate the responsibility of creating a string representation to the toString methods of Profile

and Interests. This is an example of delegation, where an object relies on another to provide a specified set of functionalities.

Class Diagram (10 points):



Plan for Next Iteration (10 points):

The rest of the game would be the testing for compatibility with mock users we create, creating these mock users, and implementing the main game functionality that leads the user through the game. So far we did more of the back side data part of storing the data the users input, thus now we would focus on the actual user interface portion of the game and putting the pieces together. We will also be working on allowing users to save profiles of people who interest them.

REPO LINK: https://github.com/balarbirwa/CSCI4448-Final-Project/tree/main