객체지향프로그래밍 - 과제 #2

- 202184021 소프트웨어전공 박지민

1. 문제 정의

1. 플레이어는 각각 이름을 가지고 있으며, `Player` 클래스로 구현한다.
2. 두 명의 플레이어가 참여하는 `GamblingGame`을 구현한다.
3. 각 플레이어는 자신의 이름을 입력하고, 게임은 순서대로 진행된다.
4. 플레이어는 자신의 차례에 Enter 키를 눌러 게임을 진행한다.
5. 세 개의 랜덤 숫자를 생성하며, 세 숫자가 모두 동일하면 해당 플레이어가 승리한다.
6. 승자가 나올 때까지 게임은 계속 진행된다.

2. 문제 해결 방법

2-1. 클래스 설계

- **Player 클래스**
  - `std::string name` : 플레이어의 이름을 저장한다.
  - `Player(std::string& name)` : 생성자로 이름을 초기화한다.
  - `std::string& getName()` : 플레이어의 이름을 반환한다.
- **GamblingGame 클래스**
  - `Player* players[2]` : 두 명의 플레이어를 저장하는 포인터 배열.
  - `int index` : 현재 차례인 플레이어의 인덱스.
  - `GamblingGame(Player* player1, Player* player2)` : 생성자로 두 플레이어를 초기화한다.
  - `void play()` : 게임의 전체 흐름을 관리한다.

2-2. 동작 과정

1. main.cpp에서 사용자로부터 두 플레이어의 이름을 입력받아 Player 객체를 **생성**한다.
2. GamblingGame 객체를 생성하여 두 플레이어를 전달한다.
3. GamblingGame의 `play()` 함수를 호출하여 게임을 시작한다.
   - 현재 플레이어가 Enter 키를 누르면 세 개의 랜덤 숫자를 생성한다.
   - 세 숫자가 모두 동일한지 검사하여 승리 여부를 판단한다.
   - 승리 시 게임을 종료하고, 그렇지 않으면 다음 플레이어로 넘어간다.

3. 아이디어 평가

- **객체지향 프로그래밍의 적용**
  - Player와 GamblingGame 클래스를 분리하여 **역할과 책임**을 명확히 하였다.
  - 데이터 은닉을 위해 멤버 변수를 `private`로 선언하고, 접근을 위한 멤버 함수를 제공하였다.
- **코드의 확장성 및 유지보수성**
  - 현재는 두 명의 플레이어로 제한되어 있지만, 플레이어 배열을 동적으로 할당하면 다수의 플레이어로 확장할 수 있다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

- **게임 진행 흐름 제어**

    - GamblingGame 클래스의 play() 함수에서 무한 루프를 사용하여 게임을 진행하고, 승리 조건이 만족되면 루프를 탈출하여 게임을 종료한다.
    - 플레이어의 순서를 관리하기 위해 index 변수를 사용하여 0과 1 사이에서 토글한다.
- **랜덤 숫자 생성 및 비교**
    - srand() 함수를 사용하여 시드 값을 초기화하여 매 실행 시 다른 랜덤 값을 생성한다.
    - 세 개의 랜덤 숫자를 배열에 저장하고, 첫 번째 숫자와 나머지 숫자를 비교하여 모두 동일한지 판단한다.