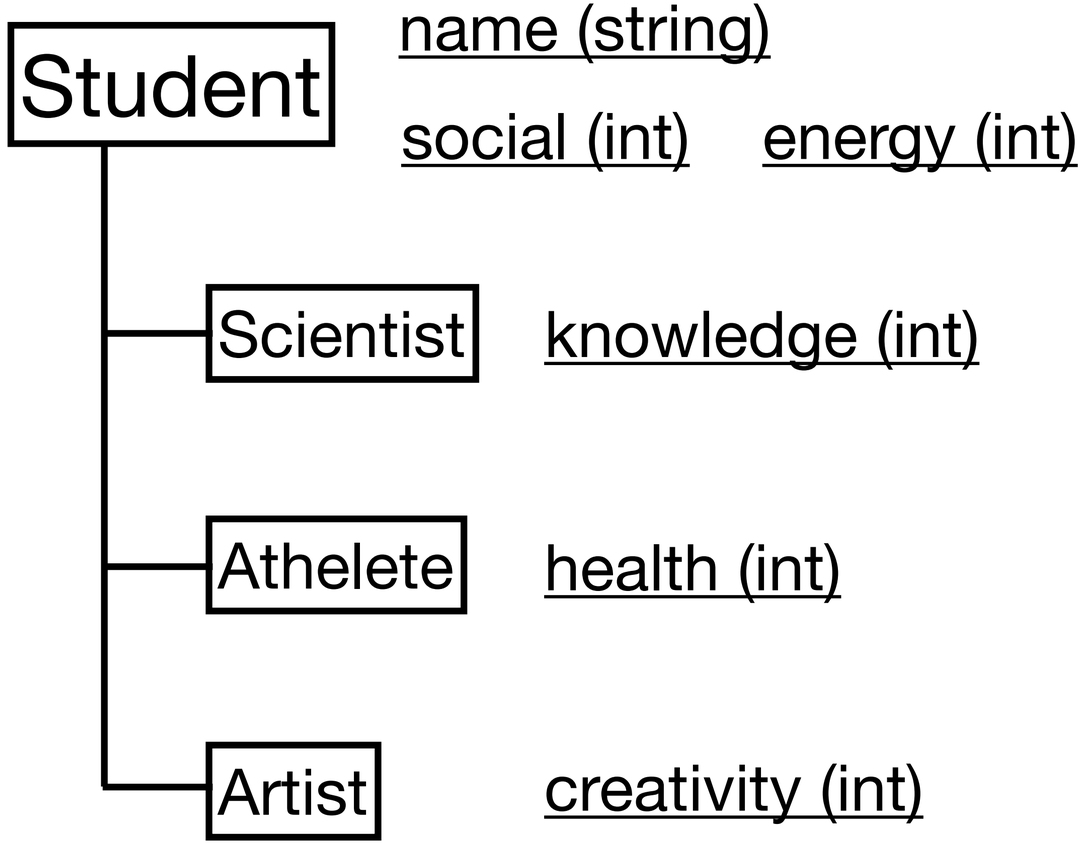
Assignment 1

Description

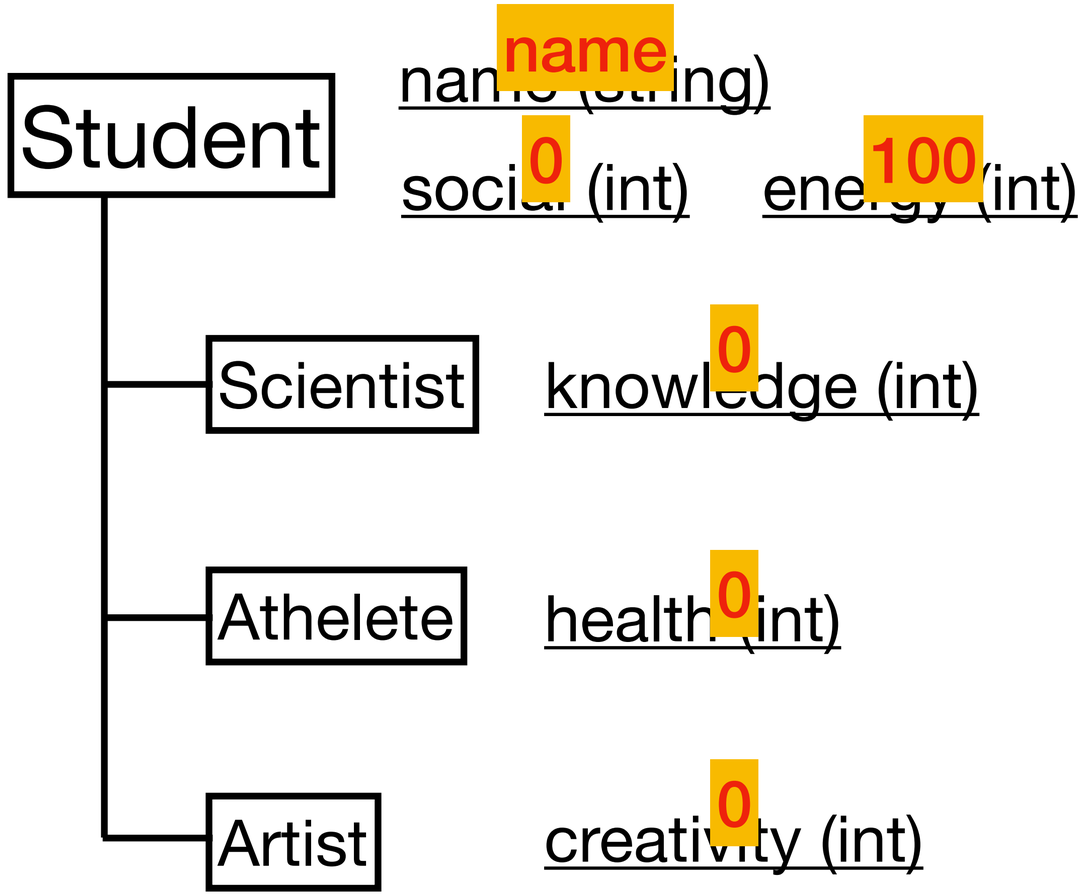
```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



Only accessible by the derived class and itself

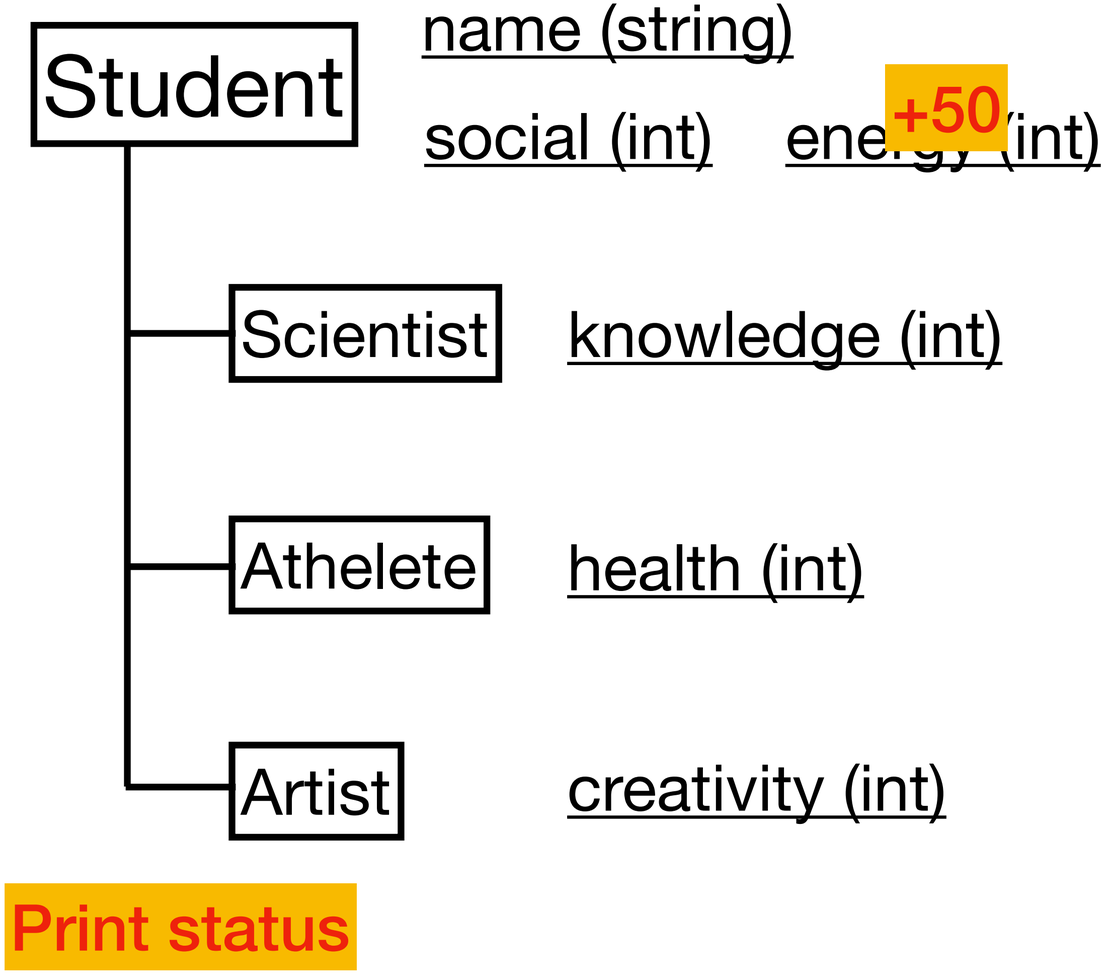
Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



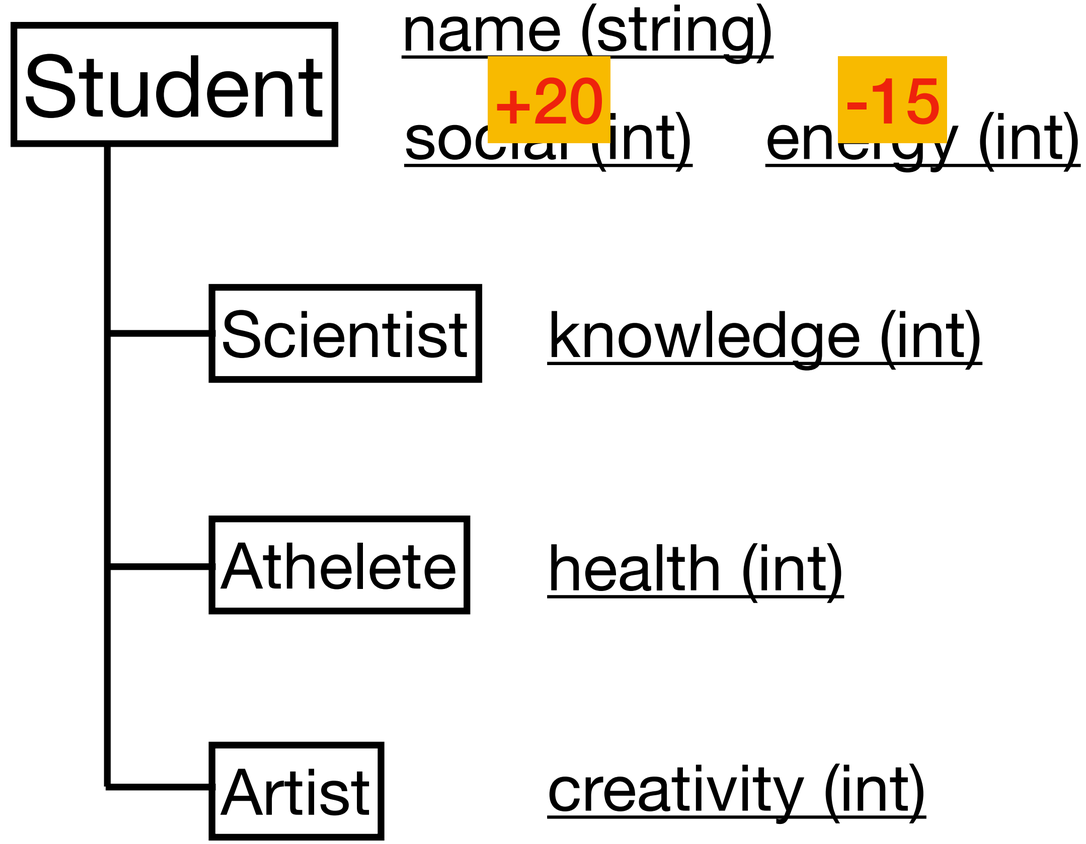
Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



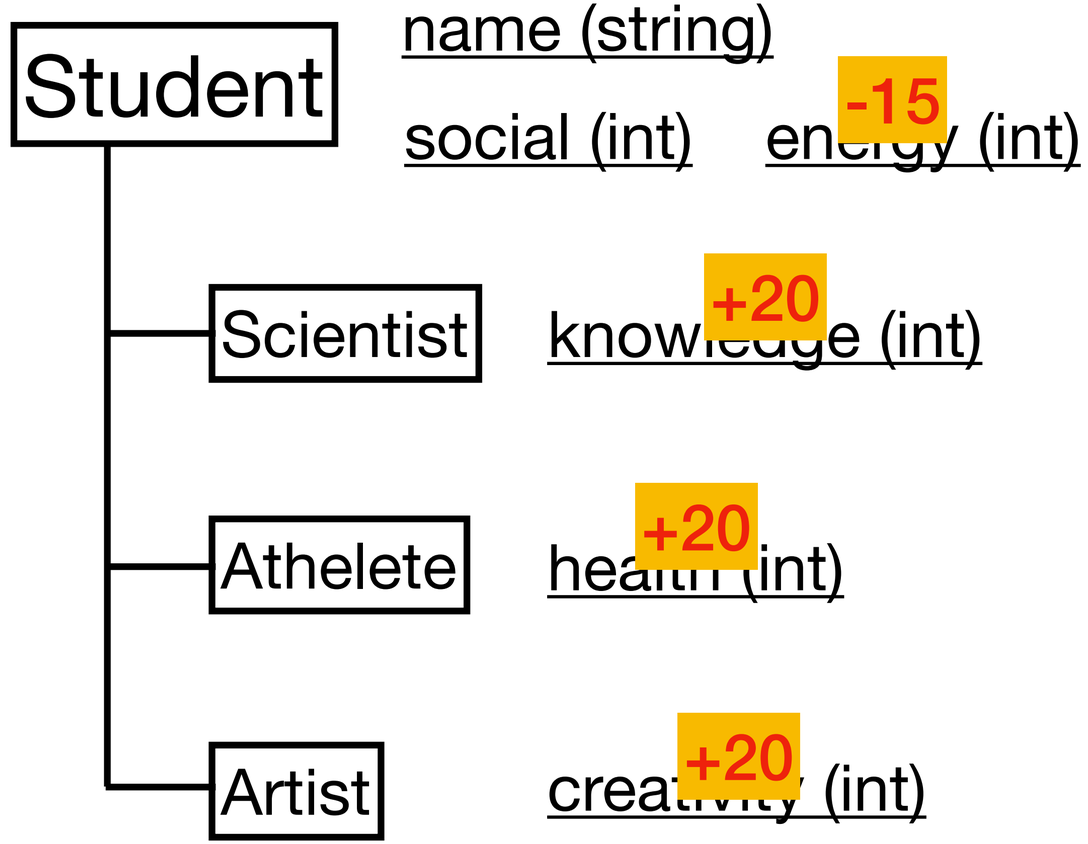
If energy is greater than or equal to 15

Print status

Else, Print name is too tired to meet friends.

Description

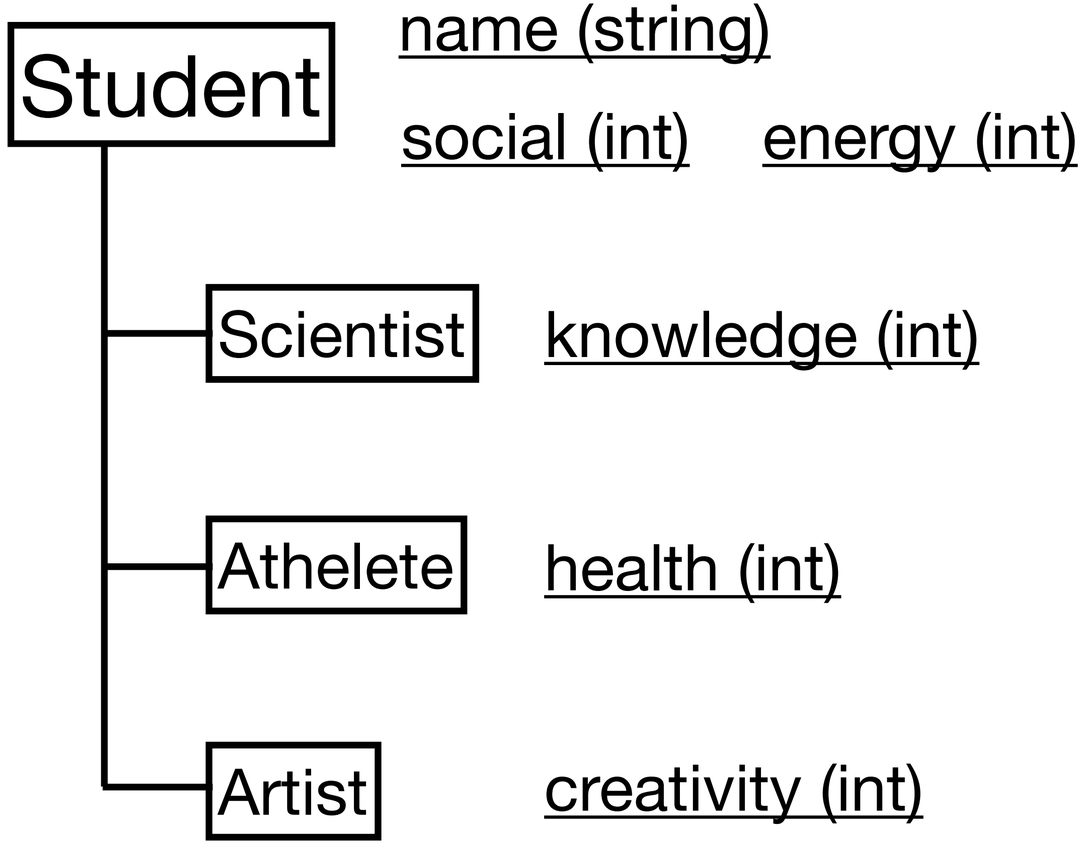
```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



Else, Print “name is too tired to [study / exercise / work on art]”

Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```

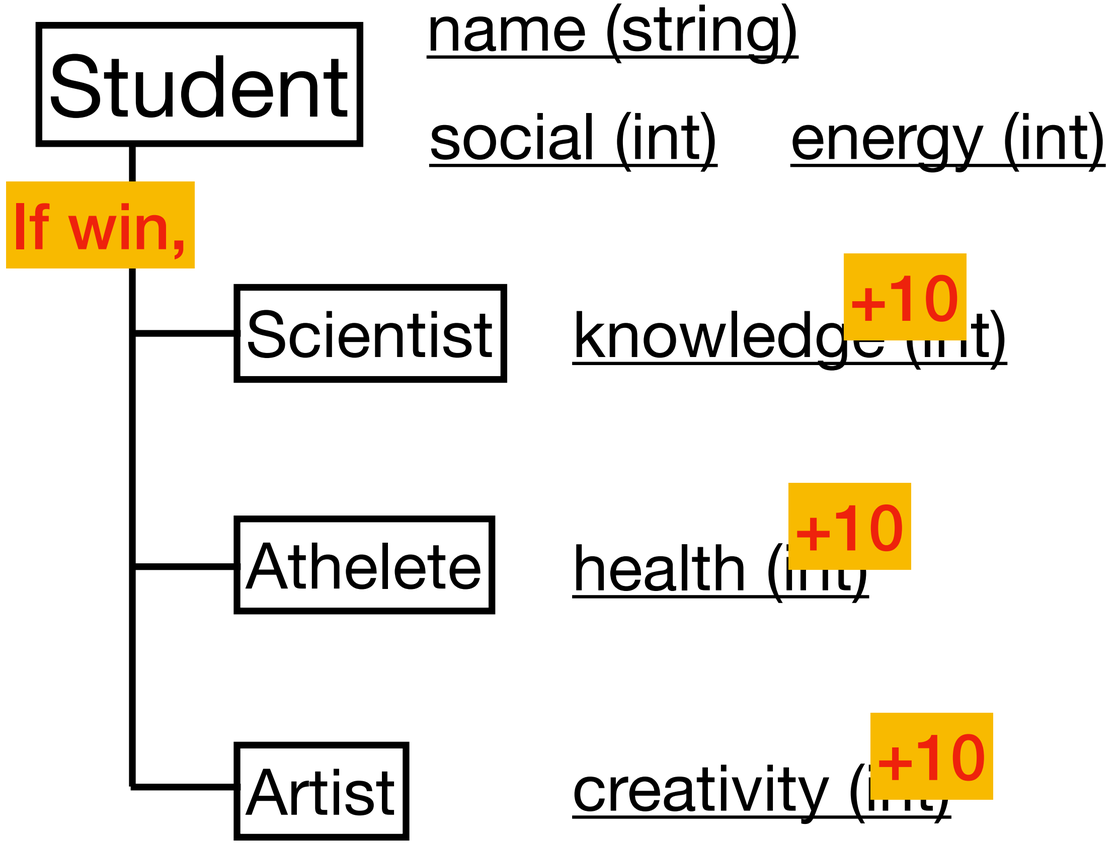


Return social + energy + [knowledge / health / creativity]

Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```

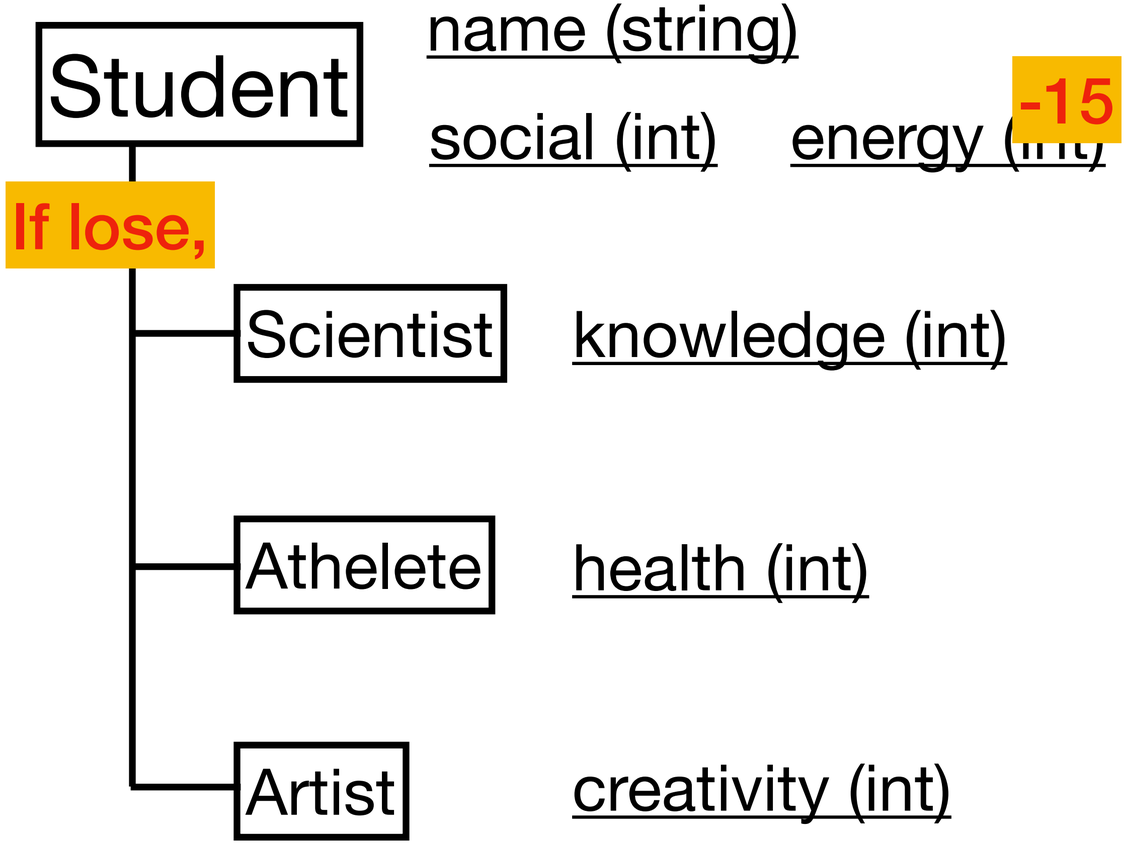
```
enum ContestResult
{
    WIN,
    LOSE,
    TIE
};
```



Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```

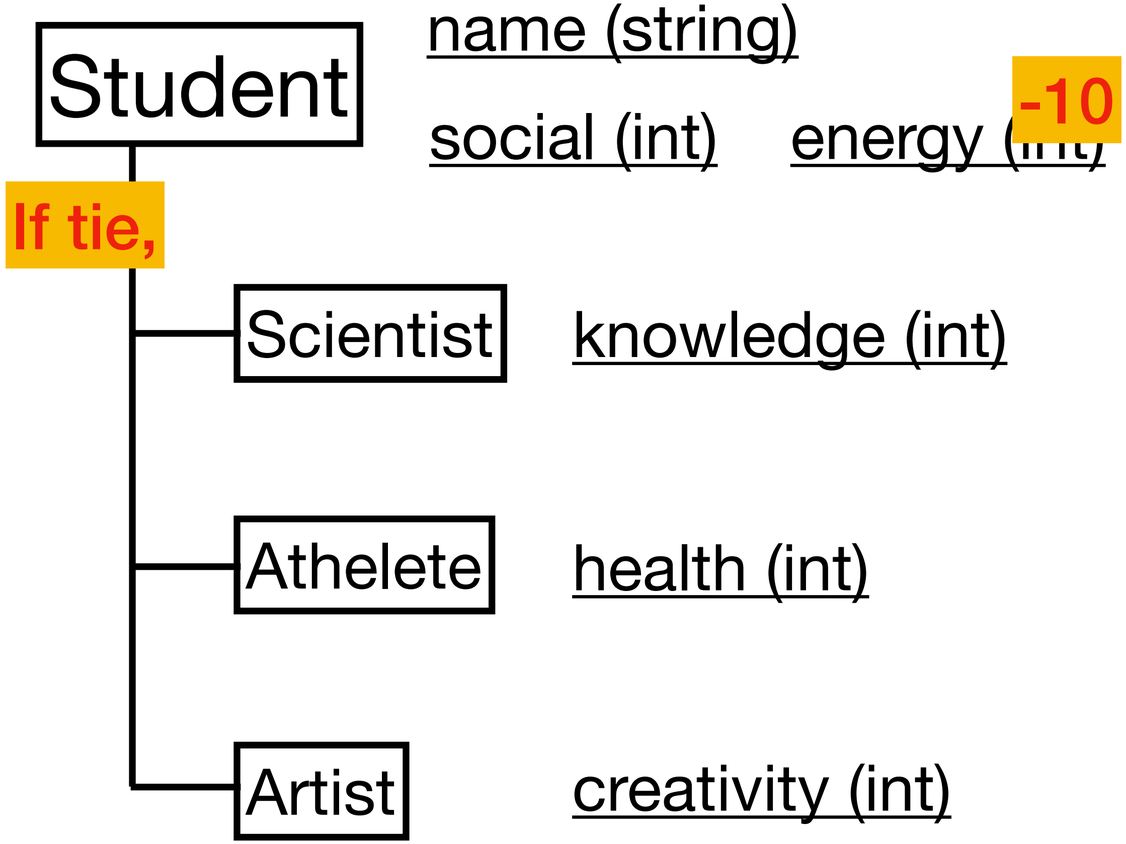
```
enum ContestResult
{
    WIN,
    LOSE,
    TIE
};
```



Description

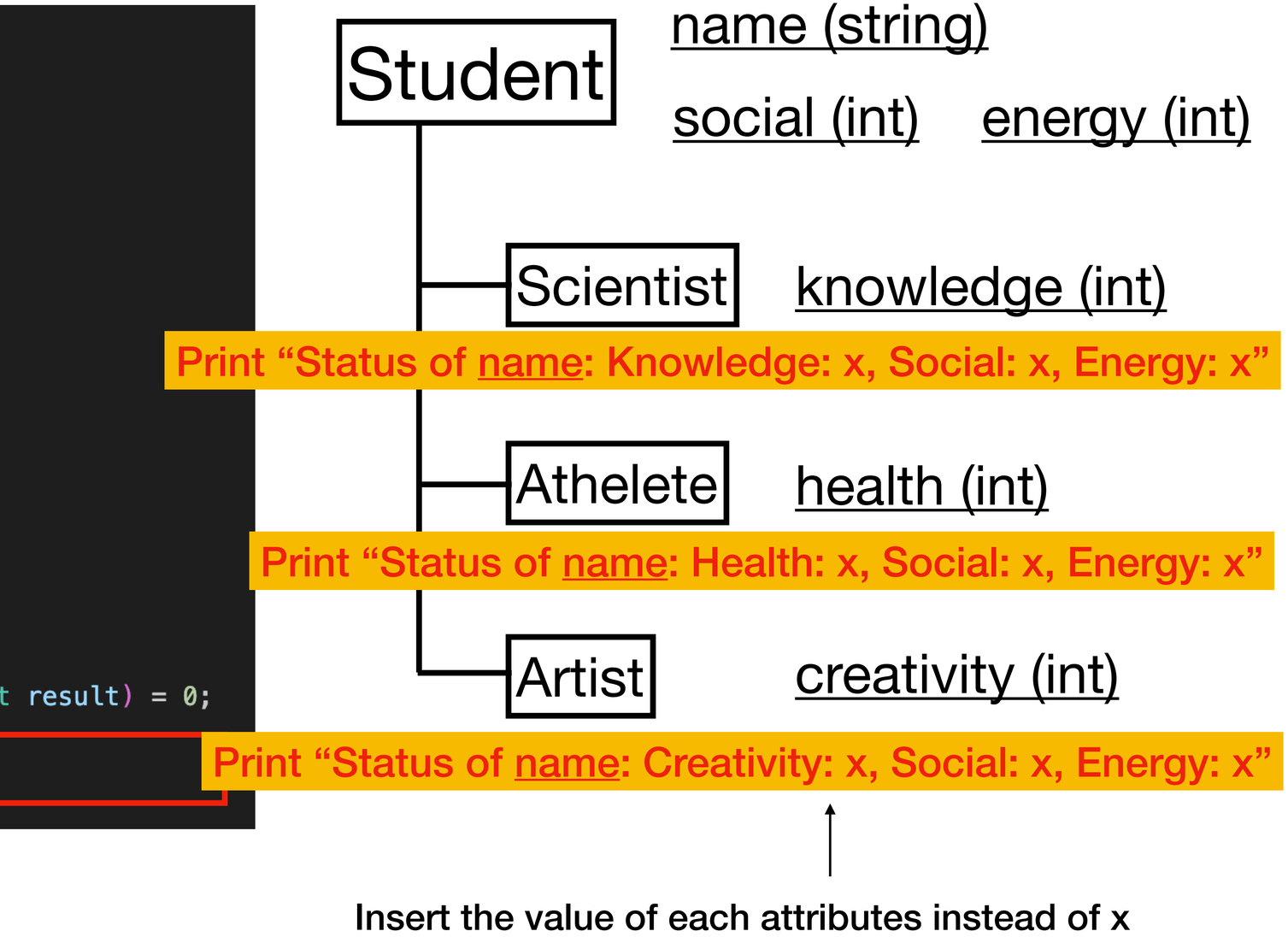
```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```

```
enum ContestResult
{
    WIN,
    LOSE,
    TIE
};
```



Description

```
class Student
{
public:
    Student(std::string name) {}

    void rest()
    {
    }

    void meetFriends()
    {
    }

    virtual void doActivity() = 0;

    virtual int getStats() = 0;

    virtual void updateAfterContest(ContestResult result) = 0;

    virtual void showStatus() = 0;
};
```



Description

```
class Game
{
private:
    Student *students[MAX_STUDENTS];
    int studentCount;

public:
    Game()
    {
        studentCount = 0;
    }

    Student *getStudent(std::string name) {}

    void trainStudent() {}

    void addStudent() {}

    void contestStudents() {}

    void showStatus() {}

    void run() {}
};
```

Return the student whose name is {name}

*There is no case of duplicated student name

*There is no case where the student

with the given name does not exist

Description

1. Ask the user to enter the student name
2. Display the training menu and ask user to choose an activity
3. Process the chosen activity
4. Repeat until the user chooses to exit(5)

*There is no case where the student with the given name does not exist

*There is no case where the user enters an invalid choice

Description

```
void trainStudent() {}
```

```
--- Training Menu ---
1. Do Activity
2. Meet Friends
3. Rest
4. Show Status
5. Exit
Enter your choice:
----------------------
```

1. Ask the user to enter the student name
2. Display the training menu and ask user to choose an activity
3. Process the chosen activity
4. Repeat until the user chooses to exit(5)

*There is no case where the student with the given name does not exist

*There is no case where the user enters an invalid choice

Description

```
class Game
{
private:
    Student *students[MAX_STUDENTS];
    int studentCount;

public:
    Game()
    {
        studentCount = 0;
    }

    Student *getStudent(std::string name) {}

    void trainStudent() {}

    void addStudent() {}

    void contestStudents() {}

    void showStatus() {}

    void run() {}
};
```

1. Ask the user to enter the student type and name

2. Create a new student object according to the given type and add it to the students array

*There is no case where the user enters an invalid student type or the name of an existing student

*There is no case where the student count exceeds MAX_STUDENTS

Description

```
void contestStudents() {}
```

1. Ask the user to enter the names of two students

2. If the energy of any student is less than 15, display a message "${StudentName} is too tired to contest."

Description

```
void contestStudents() {}
```

3. Contest the two students

- The student with higher stats wins

- If the stats are equal, it's a tie

- Display the result

  - If it is not a tie, display "${StudentName} wins!”

  - If it is a tie, display "It's a tie!"

Description

```
void contestStudents() {}
```

4. Update the stats of the students according to the result

*There is no case where the user enters the name of a non-existing student or the same name for both student

Description

```
class Game
{
private:
    Student *students[MAX_STUDENTS];
    int studentCount;

public:
    Game()
    {
        studentCount = 0;
    }

    Student *getStudent(std::string name) {}

    void trainStudent() {}

    void addStudent() {}

    void contestStudents() {}

    void showStatus() {}

    void run() {}
};
```

Display the status of all students

Description

```
--- Main Menu ---
1. Add Student
2. Train Student
3. Contest Students
4. Students Status
5. Exit
Enter your choice:
---------------------
```

```
class Game
{
private:
    Student *students[MAX_STUDENTS];
    int studentCount;

public:
    Game()
    {
        studentCount = 0;
    }

    Student *getStudent(std::string name) {}

    void trainStudent() {}

    void addStudent() {}

    void contestStudents() {}

    void showStatus() {}

    void run() {}
};
```

1. Display the main menu and ask user to choose an activity

2. Process the chosen activity

3. Repeat until the user chooses to exit(5)

과제 수행 방법

1. ETL -> Assignment1.zip 다운로드
2. 압축 해제 후 폴더 이름을 {20XX-XXXXX_Assignment1}로 수정
    * Assignment1 -> 20XX-XXXXX_Assignment1
3. Windows의 경우 task.json 파일 주석 해제
4. Skeleton 코드를 이용하여 과제 수행
5. {20XX-XXXXX_Assignment1.zip} 형태로 압축하여 과제함에 제출
    * ex) 2024-12345_Assignment1.zip

채점 관련 (필독)

채점은 입출력 기반으로 이루어질 예정

이전 슬라이드에 명시된 사항 지키지 않을 시 (파일명 오류 등) **0점처리**

기한 4/9 11:59pm 까지 / 총 8점

Late submission

- 04/10 12:00am ~ 4/16 11:59pm) 일당 1점 감점
- 4/17 12:00am ~) 0점