DELEGAÇÃO OLÍMPICA

Turma 2 grupo 1

André Daniel Gomes - up201806224

Daniel Garcia Silva - up201806524

Diana Freitas - up201806230

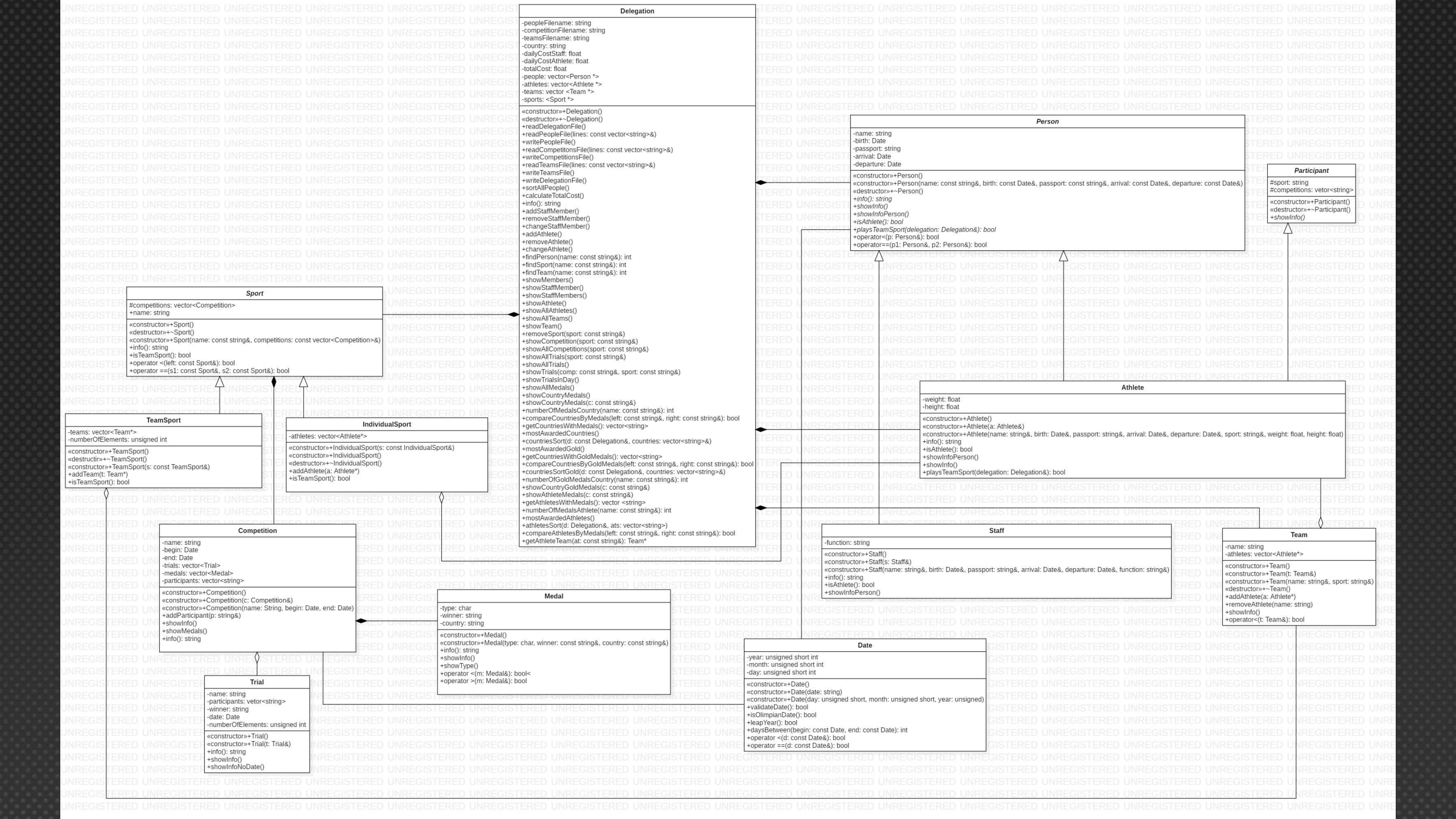
Descrição

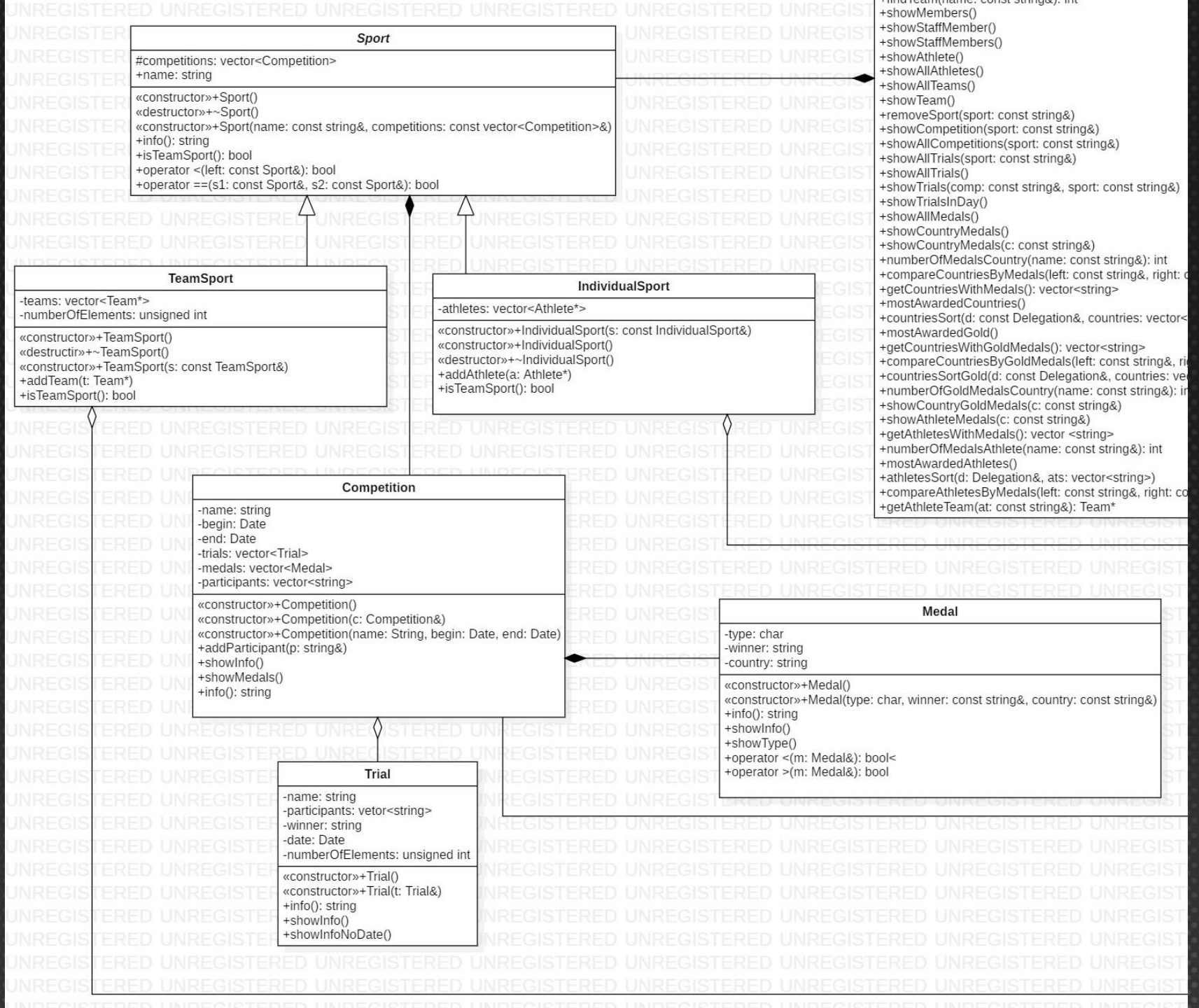
- Gestão da participação da Delegação Olímpica Portuguesa nos Jogos Olímpicos de Tóquio 2020
- Adicionar, remover ou alterar membros da delegação (atletas ou funcionários)
- Visualizar dados sobre os membros da Delegação e as equipas(individualmente ou de todos)
- Visualizar as competições, os jogos e as medalhas
    - Opções avançadas de pesquisa no caso dos jogos (num determinado dia) e inúmeras opções em termos estatísticos no que diz respeito à atribuição de medalhas - rankings
- Cálculo do custo total da participação da Delegação Portuguesa nos Jogos Olímpicos

Solução

- Utilização de menus com funcionalidades diversas de gestão e visualização de dados
- Uso de classes e ficheiros
- Uso de Hierarquia e polimorfismo – inclusive heranças múltiplas e classes abstratas
- Algoritmos de pesquisa e de ordenação e definição de operadores
- Uso de exceções e de mensagens de erro
- Múltiplas funções de verificação da coerência dos dados

Desde já destaca-se a utilização da consola do windows, pois existem funcionalidades no C-Lion que não funcionam como deviam devido ao próprio IDE, tais como o comando "cls", o ctrl-z, entre outras

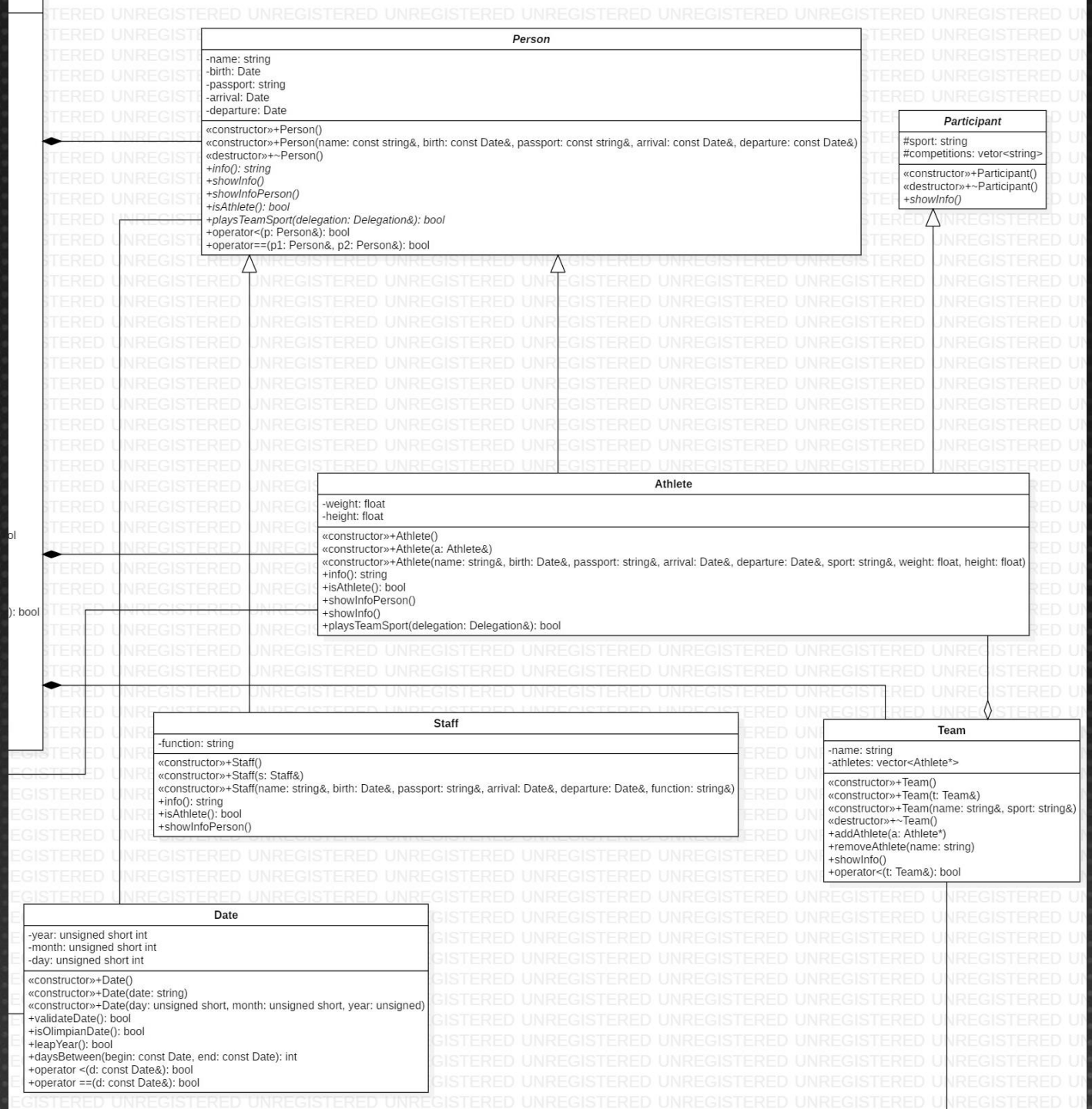

Delegation
-peopleFilename: string
-competitionFilename: string
-teamsFilename: string
-country: string
-dailyCostStaff: float
-dailyCostAthlete: float
-totalCost: float
-people: vector<Person *>
-athletes: vector<Athlete *>
-teams: vector <Team *>
-sports: <Sport *>
«constructor»+Delegation()
«destructor»+~Delegation()
+readDelegationFile()
+readPeopleFile(lines: const vector<string>&)
+writePeopleFile()
+readCompetitonsFile(lines: const vector<string>&)
+writeCompetitionsFile()
+readTeamsFile(lines: const vector<string>&)
+writeTeamsFile()
+writeDelegationFile()
+sortAllPeople()
+calculateTotalCost()
+info(): string
+addStaffMember()
+removeStaffMember()
+changeStaffMember()
+addAthlete()
+removeAthlete()
+changeAthlete()
+findPerson(name: const string&): int
+findSport(name: const string&): int
+findTeam(name: const string&): int
+showMembers()
+showStaffMember()
+showStaffMembers()
+showAthlete()
+showAllAthletes()
+showAllTeams()
+showTeam()
+removeSport(sport: const string&)
+showCompetition(sport: const string&)
+showAllCompetitions(sport: const string&)
+showAllTrials(sport: const string&)
+showAllTrials()
+showTrials(comp: const string&, sport: const string&)
+showTrialsInDay()
+showAllMedals()
+showCountryMedals()
+showCountryMedals(c: const string&)
+numberOfMedalsCountry(name: const string&): int
+compareCountriesByMedals(left: const string&, right: const string&): bool
+getCountriesWithMedals(): vector<string>
+mostAwardedCountries()
+countriesSort(d: const Delegation&, countries: vector<string>&)
+mostAwardedGold()
+getCountriesWithGoldMedals(): vector<string>
+compareCountriesByGoldMedals(left: const string&, right: const string&): bool
+countriesSortGold(d: const Delegation&, countries: vector<string>&)
+numberOfGoldMedalsCountry(name: const string&): int
+showCountryGoldMedals(c: const string&)
+showAthleteMedals(c: const string&)
+getAthletesWithMedals(): vector <string>
+numberOfMedalsAthlete(name: const string&): int
+mostAwardedAthletes()
+athletesSort(d: Delegation&, ats: vector<string>)
+compareAthletesByMedals(left: const string&, right: const string&): bool
+getAthleteTeam(at: const string&): Team*
Person
-name: string
-birth: Date
-passport: string
-arrival: Date
-departure: Date
«constructor»+Person()
«constructor»+Person(name: const string&, birth: const Date&, passport: const string&, arrival: const Date&, departure: const Date&)
«destructor»+~Person()
+info(): string
+showInfo()
+showInfoPerson()
+isAthlete(): bool
+playsTeamSport(delegation: Delegation&): bool
+operator<(p: Person&): bool
+operator==(p1: Person&, p2: Person&): bool
Participant
#sport: string
#competitions: vetor<string>
«constructor»+Participant()
«destructor»+~Participant()
+showInfo()
Sport
#competitions: vector<Competition>
+name: string
«constructor»+Sport()
«destructor»+~Sport()
«constructor»+Sport(name: const string&, competitions: const vector<Competition>&)
+info(): string
+isTeamSport(): bool
+operator <(left: const Sport&): bool
+operator ==(s1: const Sport&, s2: const Sport&): bool
Athlete
-weight: float
-height: float
«constructor»+Athlete()
«constructor»+Athlete(a: Athlete&)
«constructor»+Athlete(name: string&, birth: Date&, passport: string&, arrival: Date&, departure: Date&, sport: string&, weight: float, height: float)
+info(): string
+isAthlete(): bool
+showInfoPerson()
+showInfo()
+playsTeamSport(delegation: Delegation&): bool
TeamSport
-teams: vector<Team*>
-numberOfElements: unsigned int
«constructor»+TeamSport()
«destructir»+~TeamSport()
«constructor»+TeamSport(s: const TeamSport&)
+addTeam(t: Team*)
+isTeamSport(): bool
IndividualSport
-athletes: vector<Athlete*>
«constructor»+IndividualSport(s: const IndividualSport&)
«constructor»+IndividualSport()
«destructor»+~IndividualSport()
+addAthlete(a: Athlete*)
+isTeamSport(): bool
Staff
-function: string
«constructor»+Staff()
«constructor»+Staff(s: Staff&)
«constructor»+Staff(name: string&, birth: Date&, passport: string&, arrival: Date&, departure: Date&, function: string&)
+info(): string
+isAthlete(): bool
+showInfoPerson()
Team
-name: string
-athletes: vector<Athlete*>
«constructor»+Team()
«constructor»+Team(t: Team&)
«constructor»+Team(name: string&, sport: string&)
«destructor»+~Team()
+addAthlete(a: Athlete*)
+removeAthlete(name: string)
+showInfo()
+operator<(t: Team&): bool
Competition
-name: string
-begin: Date
-end: Date
-trials: vector<Trial>
-medals: vector<Medal>
-participants: vector<string>
«constructor»+Competition()
«constructor»+Competition(c: Competition&)
«constructor»+Competition(name: String, begin: Date, end: Date)
+addParticipant(p: string&)
+showInfo()
+showMedals()
+info(): string
Medal
-type: char
-winner: string
-country: string
«constructor»+Medal()
«constructor»+Medal(type: char, winner: const string&, country: const string&)
+info(): string
+showInfo()
+showType()
+operator <(m: Medal&): bool<
+operator >(m: Medal&): bool
Date
-year: unsigned short int
-month: unsigned short int
-day: unsigned short int
«constructor»+Date()
«constructor»+Date(date: string)
«constructor»+Date(day: unsigned short, month: unsigned short, year: unsigned)
+validateDate(): bool
+isOlimpianDate(): bool
+leapYear(): bool
+daysBetween(begin: const Date, end: const Date): int
+operator <(d: const Date&): bool
+operator ==(d: const Date&): bool
Trial
-name: string
-participants: vetor<string>
-winner: string
-date: Date
-numberOfElements: unsigned int
«constructor»+Trial()
«constructor»+Trial(t: Trial&)
+info(): string
+showInfo()
+showInfoNoDate()



Classes

Modalidades e derivadas

- Sport
  - IndividualSport
  - TeamSport
- Competition
- Trial
- Medal



Classes

Pessoas e participantes

- Participant
  - Athlete
  - Team
- Person
  - Athlete
  - Staff
- Team

Herança Múltipla

Estrutura dos ficheiros .txt

Exemplo: estrutura do ficheiro people.txt

Rute Xavier	Nome
01/10/1995	Nascimento
CB023167	Passaport
24/07/2020	Data de chegada
08/08/2020	Data de partida
Vela	Desporto
SailClassification1	Competições
53.6	Peso
182	Altura
Carlos Dias	Nome
09/09/1993	Nascimento
CB009404	Passaport
26/07/2020	Data de chegada
07/08/2020	Data de partida
Limpeza	Função

```
for (size_t i = 0; i < lines.size(); i++) {
        numline++;
        line = lines[i];

        if (line.empty()) { // Se a linha está vazia voltamos a colocar o numLines a 1 para ler a próxima pessoa
            numline = 1;
            i++;
            line = lines[i];
        }

        if (numline == 1) { // se for a primeira linha de uma pessoa vamos ver se é funcionário ou atleta
            readFunc = lines[i + 6].empty();
            competitions.resize(0);
            a = new Athlete();
            s = new Staff();
            competitionsStream.clear();
        }

        //ler atleta ou funcionario
        if (!readFunc) { // se tiver mais de 6 linhas estamos perante um atleta
            //ler atleta
            switch (numline) {
                case 1:
                    if (checkStringInput(line) != 0) {
                        throw FileStructureError(peopleFilename);
                    }
                    a->setName(line);
                    break;
                case 2:
                    if (checkDateInput(line, d) != 0) {
                        throw FileStructureError(peopleFilename);
                    }
                    a->setBirth(d);
                    break;
```

Nota 1: Para facilitar a leitura da informação, transformou-se a informação do ficheiro num vetor de linhas

Nota 2: A distinção entre atletas e Funcionários pode ser identificada pelo número de atributos até ser encontrada uma linha vazia

Nota 3: É necessário criar um novo Atleta/membro do Staff cada vez que se começa a ler um novo membro da delegação, sendo este adicionado ao vetor de pessoas usando o construtor de cópia: `people.push_back(new Athlete(*a));`

Nota 4: Utiliza-se o switch case para definir cada atributo – defenido em cada linha

```
if (system("CLS")) system("clear");
        cout << "_____" << endl << endl;
        cout << "\t\t   Staff's Options " << endl;
        cout << "_____" << endl << endl;

        cout << "1 - Add Staff member" << endl;
        cout << "2 - Remove Staff member" << endl;
        cout << "3 - Change Staff member" << endl;
        cout << "4 - Show Staff member" << endl;
        cout << "5 - Show All Staff members" << endl;
        cout << "0 - BACK" << endl;

        do {
            testinput = checkinputchoice(input, 0, 5);
            if (testinput != 0 && testinput != 2)
                cerr << "Invalid option! Please try again." << endl;
        } while (testinput != 0 && testinput != 2);
        if (testinput == 2)
        { input = "0"; }

        switch (stoi(input)) {
            case 1:
                try{
                    delegation.addStaffMember();
                }
                catch(PersonAlreadyExists & e){
                    cout << e;
                    exceptionHandler();
                }
                break;
```

Nos menus, quando chamada qualquer função que possa lançar uma exceção, utiliza-se o" try catch", imprimindo a mensagem de erro associada à exceção. Utiliza-se também um "handler" que permite ao utilizador regressar ao menu anterior

Tratamento de Exceções

Listagem de exceções utilizadas

- FileError
- FileStructureError
- NonExistentSport
- NoSports
- NonExistentCompetition
- NonExistentTrial
- NonExistentPerson
- NonExistentStaff
- NonExistentAthlete
- NonExistentTeam
- PersonAlreadyExists
- TeamAlreadyExists
- NoMembers
- NoCompetitions
- NoTrials
- NoMedals
- NotATeamSport

```
void Delegation::removeStaffMember() {
    int test = 0;
    int index;
    string input = "", tmp;

    cout << "Name: ";
    getline( &: cin,  &: tmp);
    if (cin.eof()) {
        cin.clear();
        return; //go back on ctrl+d
    }
    cin.clear();
    while (checkStringInput( &: tmp)) {
        cout << "Invalid Name. Try again!" << endl;
        cout << "Name: ";
        getline( &: cin,  &: tmp);
        if (cin.eof()) {
            cin.clear();
            return; //go back on ctrl+d
        }
        cin.clear();
    }
    index = findPerson(tmp);
    if (index == -1 || people.at(index)->isAthlete()) {
        throw NonExistentStaff(tmp);
    } else {
        vector<Person *>::iterator it = people.begin() + index;
        delete *it;
        people.erase(it);
        cout << endl << "Staff Member removed with success!" << endl;
```

CRUD

- **Delegação**: atualização do custo total da delegação ao escrever no ficheiro
- **Pessoas**: Adiconar, remover, mudar atributos ou ver informação de uma pessoa (atleta ou membro do staff)
- **Equipas**: Visualizar informação de uma equipa
- **Competições**: Visualizar informação de comeptições, jogos e medalhas atribuídas (considerando que os jogos já ocorreram)

Outras Funcionalidades

- Utilização de Vetores para guardar apontadores para objetos de classes e classes Derivadas, de string para guardar nomes e de objetos
- Utilização de Pesquisa Sequencial
- Ordenação de Vetores com Sort, funções de comparação de objetos e de apontadores para objetos, operadores, Insertion Sort
- Leitura e escrita em ficheiros
- Utilização do CTRL-Z para voltar atrás
- Validação de Inputs

Solução Peculiar

- Para alterar dados específicos de atletas ou funcionários (que não pertencem à classe mãe Person) no vetor de apontadores para objetos do tipo Person, usámos dynamic cast

Dificuldades/esforço

- Gerir divisão de tarefas/comunicação
- Organização dos dados nos ficheiros