FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Programação Orientada aos Objetos

Alexandre Américo Ferreira – 2016240875

Guilherme Moita Pontes – 2016242153

Relatório Projeto

"Serão Convívio de POO"

Ano letivo 2017/2018

Neste relatório explicaremos a estrutura geral do programa, apresentaremos os diagramas inicial e final e a descrição das principais estruturas e métodos utilizados. Os métodos menos específicos e mais gerais a todas as classes encontram-se comentados no **JavaDoc**.

De um modo geral, o programa é composto na totalidade por 9 interfaces gráficas. No menu principal temos acesso às 3 principais categorias: **Pessoas**, **Locais** e **Receitas**.

- Na interface **Pessoas**, cada pessoa pertencente ao DEI, pode registar-se no convívio, obter um login e posteriormente adicionar/remover até 5 locais que pretende visitar.
- Na interface **Locais** é possível consultar listas de todos os locais do convívio, dos locais mais visitados por ordem decrescente e as guest lists dos bares que o utilizador desejar.
- Na interface **Receitas** é possível consultar a receita mínima de cada local individualmente ou a receita mínima total do convívio.
- Por fim, o botão **Sair** guarda para o ficheiro de objetos "fich_obj.dat" o estado atual do programa e encerra-o.

Diagrama inicial:

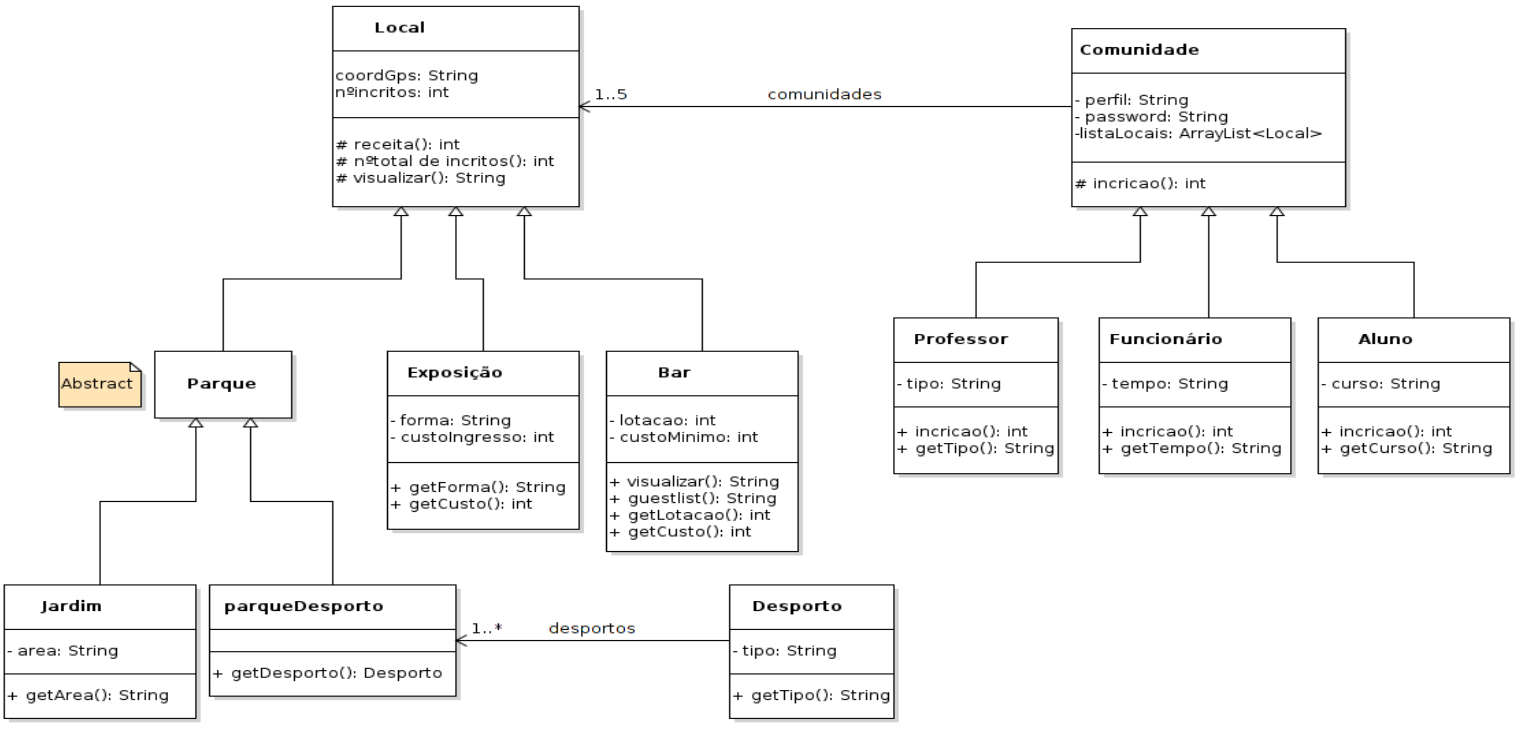
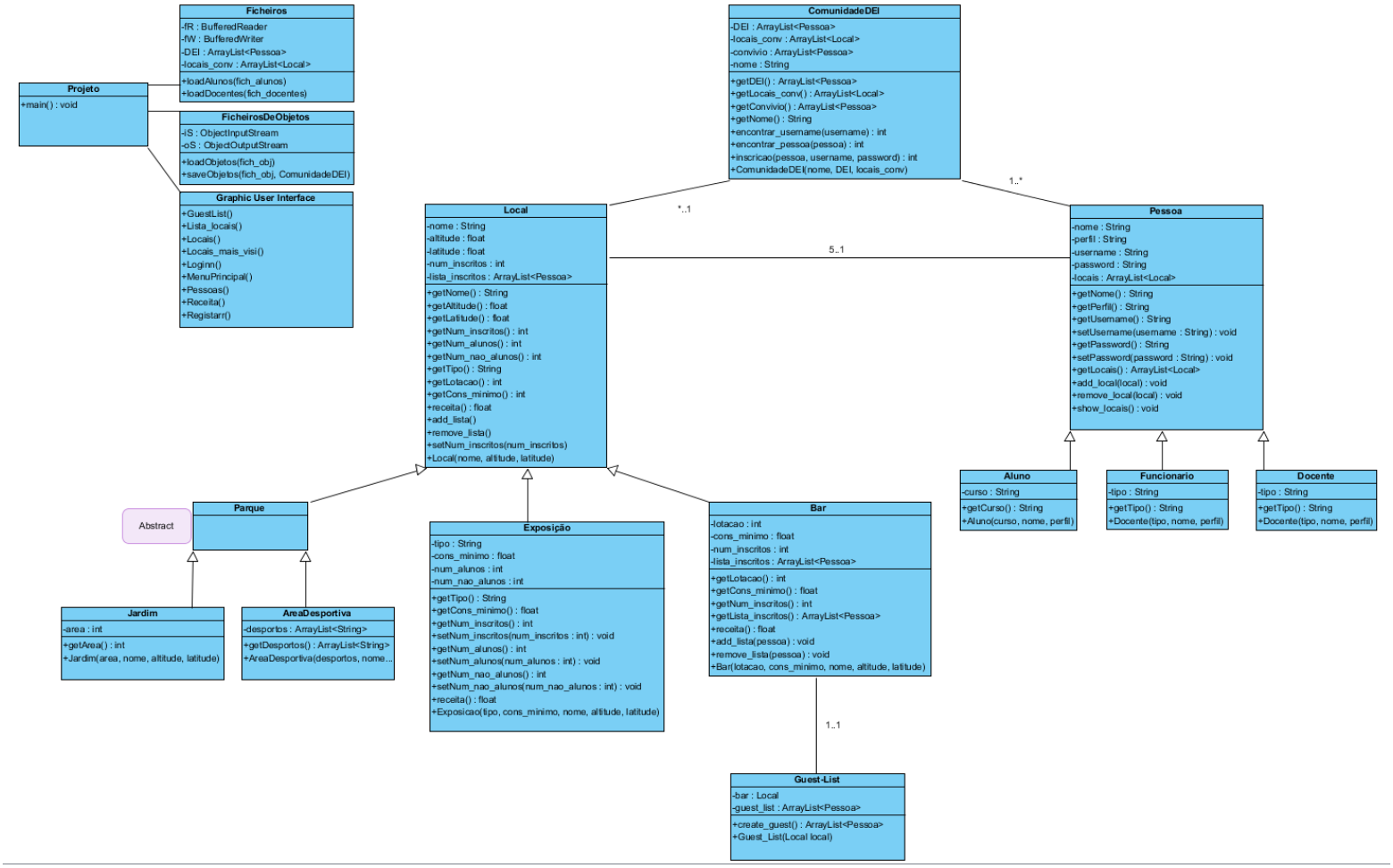
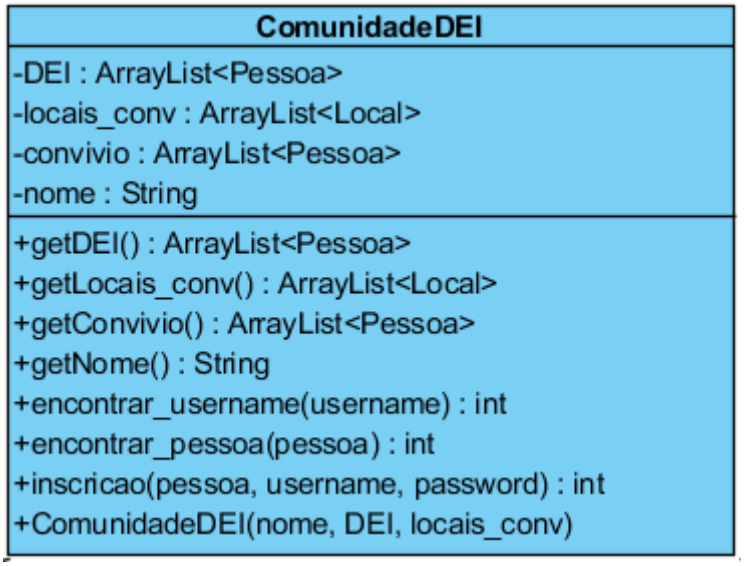


Diagrama final:





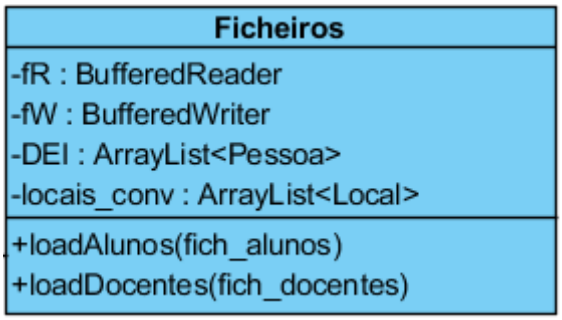
A classe **ComunidadeDEI** vai, como o nome indica, criar a comunidade para este convívio. Para o fazer recorremos a:

- um **ArrayList** de **pessoas** nomeado **DEI** que corresponde a todos os alunos, docentes e funcionários pertencente ao DEI e que é carregado a partir dos ficheiros de texto;
- um **ArrayList** de **pessoas** nomeado **convívio** que corresponde a todos os alunos, docente e funcionários já inscritos no convívio, juntamente com os usernames e passwords de login;
- um **ArrayList** de **locais** nomeado **locais_conv** que corresponde a todos os locais em que as pessoas do convívio se podem inscrever.

O método **inscrição** é o método mais relevante nesta classe. Ele recebe uma pessoa pertencente ao DEI, um **username** e uma **password** escolhidos pelo utilizador. O objetivo é adicionar ao convívio a pessoa em questão verificando primeiro se a pessoa está contida no DEI (retorna -1 caso não esteja), se o username escolhido ainda não se encontra em utilização (retorna -2 caso já esteja) e por último se a pessoa ainda não está escrita no convívio (retorna -3 caso esteja). Concluindo com sucesso, é retornado 0. Estes retornos de controlo são passados para a interface que irá apresentar o sucesso ou o erro da operação através de um "switch".

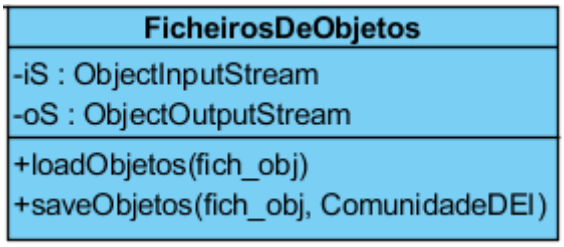
Para além dos comuns "getters", são declaradas duas funções de auxilio à função "**inscrição**" nomeadas "**encontrar_username**" e "**encontrar_pessoas**" para averiguar se os usernames e pessoas passados como argumentos já se encontram utilizados.

Para o save e load de dados, sempre que o programa é iniciado vai ser verificado se já existe ficheiro de objetos. Caso exista, o programa vai ser carregado a partir do mesmo e será restaurado o estado anterior. Caso não exista, o programa irá recorrer aos ficheiros de texto para carregar pessoas e locais neles contidos. Para isso, foram usadas duas classes: **Ficheiros** e **FicheirosDeObjetos**.

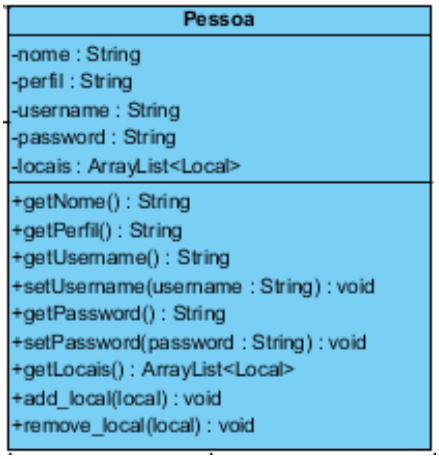


- A classe **Ficheiros** tem como atributos um **BufferedReader** "**fR**" e um **BufferedWriter** "**fW**" de modo a manipular os ficheiros .txt e **dois ArrayLists**, em que um armazenará as pessoas e outro os locais lidos. Como não será necessário alterar os ficheiros .txt apenas são usados métodos de **load**. A informação

relativa às pessoas está guardada linha a linha pelo nome seguido das caraterísticas necessárias a cada classe separadas por "tab" e existem 3 ficheiros para as 3 classes ("**Alunos.txt**", "**Docentes.txt**" e "**Funcionarios.txt**").
No caso dos locais, é tudo agrupado no ficheiro "Locais.txt" e é lido de forma um pouco diferente: cada linha contém o tipo de local, o nome e só depois as caraterísticas necessárias a cada um, tudo separado por "tabs".
Os métodos "**getDei**()" e "**getLocais**()" servem apenas para passar como argumento os ArrayLists criados a partir dos ficheiros ao construtor "**ComunidadeDEI**".



- Por outro lado, a classe **FicheirosDeObjetos** tem como atributos um "**ObjectInputStream**" nomeado "**iS**" e um "**ObjectOutputStream**" nomeado "**oS**". Como os nomes indicam a função "**loadObjetos**(**ficheiros_objetos**)" irá carregar para o programa os dados contidos em "**ficheiro_objetos**" e a função "**saveObjetos**(**ficheiro_objetos**, **ComunidadeDEI**)" irá guardar para o "**ficheiro_objetos**" fornecido o estado atual da "**ComunidadeDEI**" cada vez que se encerra o programa.



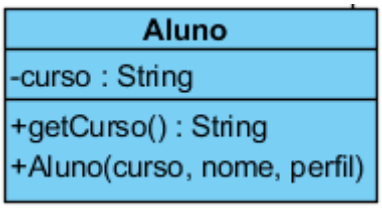
A classe **Pessoa** é a classe "pai" das classes **Aluno, Funcionário e Docente** recorrendo ao polimorfismo. Tem como atributos **nome, perfil, username, password** e um ArrayList de locais em que essa pessoa se encontra inscrita com capacidade até 5.

Os "getters" e "setters" necessários tanto para os atributos da classe como para o polimorfismo são criados nesta classe juntamente com os métodos mais

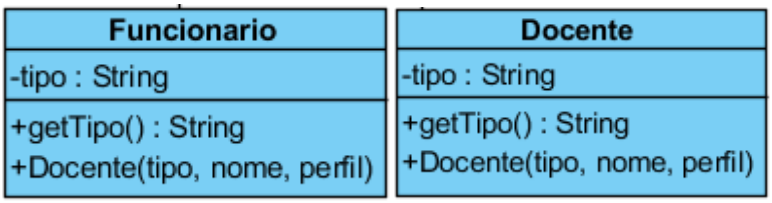
importantes e que serão usados pelas classes que a extendem. Os referidos métodos são:

- **add_local(Local local):** tem como função inscrever uma pessoa no convívio e retorna um inteiro com a informação se a pessoa foi inscrita no local com sucesso ou não. Primeiramente vamos buscar o **num_alunos**, **num_nao_alunos** e **num_inscritos** do local pois se o local for uma exposição os alunos tem desconto por isso e necessário diferencia-los. Depois verifica se a pessoa já está inscrita no local. Percorre o arraylist de locais donde esta está inscrita e se encontrar o local retorna -2. Caso isto não se verifique o próximo passo é ver se a pessoa já está inscrita em 5 locais. Se o tamanho do arraylist "**locais**" for maior ou igual a 5 retorna -3. Se isto também não se verificar o passo seguinte é ver que tipo de local é que estamos a falar. A distinção é feita graças ao polimorfismo. Se o local tiver lotação diferente de **0** estamos a falar de um bar, se tiver tipo diferente de **""** estamos a falar de uma exposição e se ambos estes atributos forem **0** e **""** respetivamente estamos a falar dos outros locais (**AreaDesportiva, Jardim**). Se a lotação for diferente de 0 vamos ver se o bar já está cheio. Se não tiver o local é adicionado a lista "**locais"** da pessoa e a pessoa é adicionada a **lista_inscritos** do bar, retornando 0 no fim. Se o bar tiver lotação esgotada retorna -1. Se o tipo for diferente de "" vamos ver se a pessoa em causa se trata de um aluno ou não aluno. Se for aluno o método **getCurso()** for diferente de "" é por se trata de um aluno. Assim o **num_alunos** vai ser incrementado uma vez e redefinido no local através do método **setNum_alunos()**. Se não for um aluno o **num_nao_alunos** vai ser incrementado uma vez e redefinido no local através do método **setNum_nao_alunos().** Se o local não for nem um bar nem uma exposição o **num_inscritos** vai ser incrementado uma vez e redefinido no local através do método **setNum_inscritos()**.

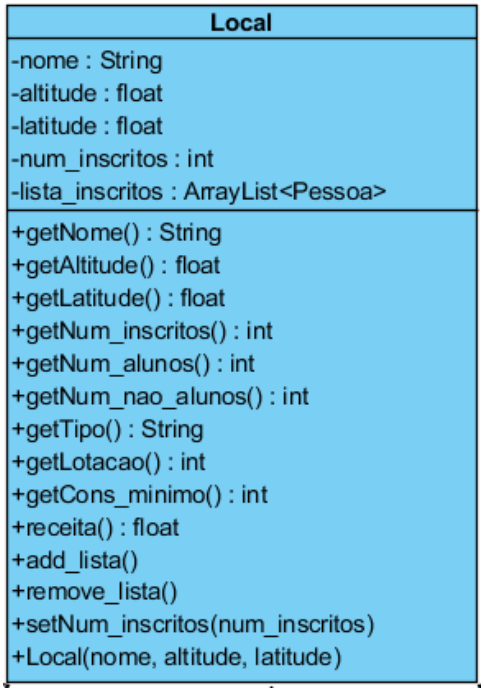
- **remove_local(Local local):** é um método **void** que tem como objetivo remover uma pessoa do local passado como argumento. Funciona de forma semelhante ao método **add_local**. Carrega o **num_alunos, num_nao_alunos e num_inscritos** do local e depois percorre o array **locais.** Se encontra o local no array irá proceder às verificações como no método **add_local()** só que em vez de incrementar os valores decrementa-os e no caso se o local for bar em vez de adicionar a pessoa a lista de inscritos do bar remove-a dessa lista.



A classe **Aluno** descende da classe **Pessoa**, herdando assim os seus atributos. Para além da herança também possui um outro atributo que é o curso que, através do polimorfismo, irá permitir distinguir se uma pessoa é um aluno ou não. Esta classe apenas possui um método **getCurso()** que retorna o curso do aluno. Aqui é que acontece o polimorfismo.

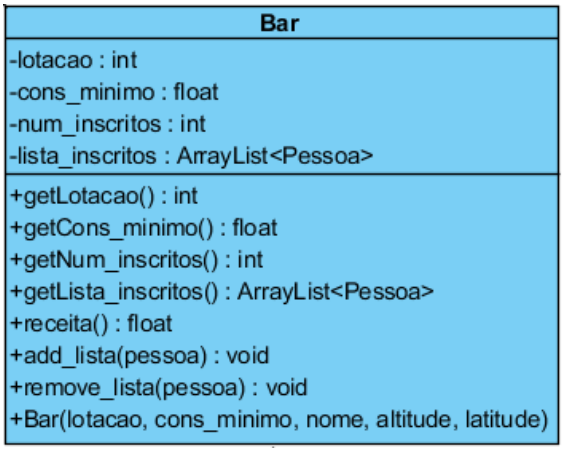


A classe **Docente** e **Funcionário,** tal como a classe Aluno, também descendem da classe pessoa. Apenas possuem um método **getTipo()** que torna que tipo de docente/funcionário a pessoa é (polimorfismo).

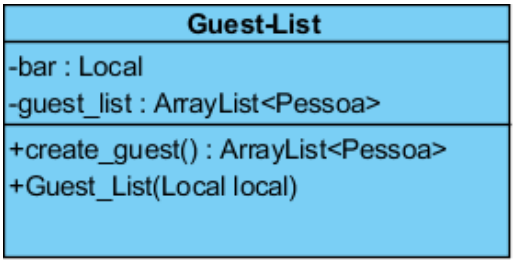


A classe **Local** é a classe "pai" das classes **Parque, Exposição e Bar** recorrendo ao polimorfismo. O seu construtor tem como atributos **nome, altitude, latitude, num_inscritos** e um ArrayList de pessoas inscritas nesse local.

Os “getters” e “setters” necessários tanto para os atributos da classe como para o polimorfismo são criados nesta classe juntamente com os métodos mais importantes e que serão usados pelas classes que a extendem.



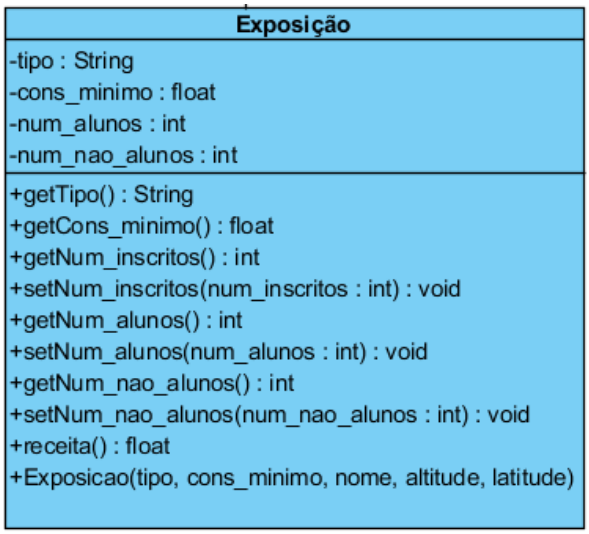
A classe **Bar** tem uma particularidade diferente dos outros locais: esta classe não contem apenas o numero de inscritos, ao invés contém uma lista de inscritos para facilitar a criação da guest list. Tem como métodos o **construtor**, os “**getters**” e “**setters**”, uma função para adicionar pessoas ao atributo “lista de inscritos”, outra para as remover e por fim a função do cálculo de receitas. Como não existem casos especiais de preços para os bares a receita é calculada simplesmente multiplicando o numero de inscritos pelo consumo mínimo.



A classe **Guest-List** está sempre associada a um bar. O seu **construtor** recebe-o como argumento e cria um **ArrayList** a ser preenchido com **pessoas**, sendo o tamanho 20% da lotação do bar. O único método é “**create_guest**” que como o nome indica vai criar a guest list de um determinado bar. Para isso a lista de inscritos do bar é percorrida duas vezes:

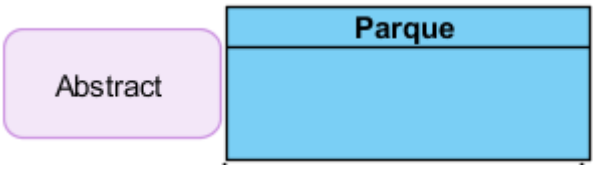
- a primeira para preencher a guest list com pessoas de perfil Boémio (pois têm prioridade),
- a segunda para, caso a guest list ainda não se encontre cheia, preencher o resto das pessoas “não Boémias” de acordo com a ordem de chegada.

Caso não existam pessoas inscritas no bar e, consequentemente na guest list, irá ser retornado **null**. Caso seja criada uma guest list com sucesso, esta será retornada.

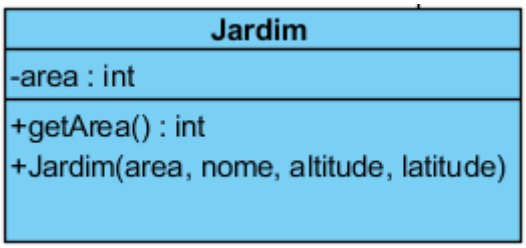


A classe **Exposição** é constituída pelo seu **construtor**, "**getters**" e "**setters**" e pelo método **receita**. Nesta classe temos duas particularidades:

- o parâmetro "**num_inscritos**" existente noutros locais está dividido em "**num_alunos**" e "**num_nao_alunos**"
- o método **receita** tem de aplicar um desconto de 10% aos alunos, sendo este o motivo da divisão previamente explicada.



A classe **Parque** é uma classe abstrata, isto é, serve apenas para diferenciar as classes **Jardim e AreaDesportiva** que vão descendem dela.



A classe **Jardim** extende a classe abstrata parque e é constituída apenas pelo seu **construtor** e o método **getArea().**

A classe **AreaDesportiva**, tal como a classe **Jardim**, extende a classe abstrata **Parque** e é constituída pelo seu **construtor** e pelo método **getDesportos()**. Os desportos relativos a cada **AreaDesportiva** são armazenados num **ArrayList** de **Strings**.