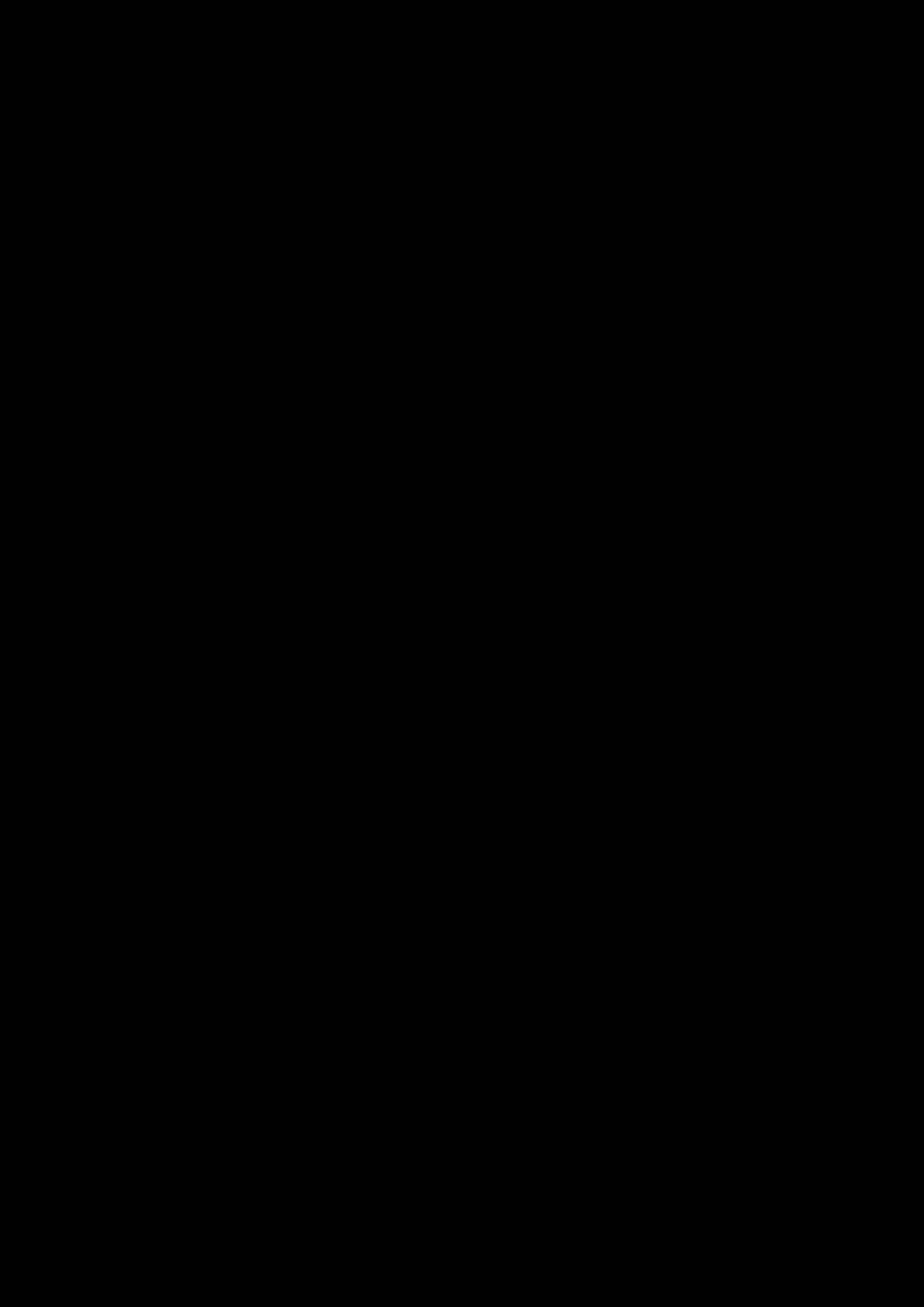
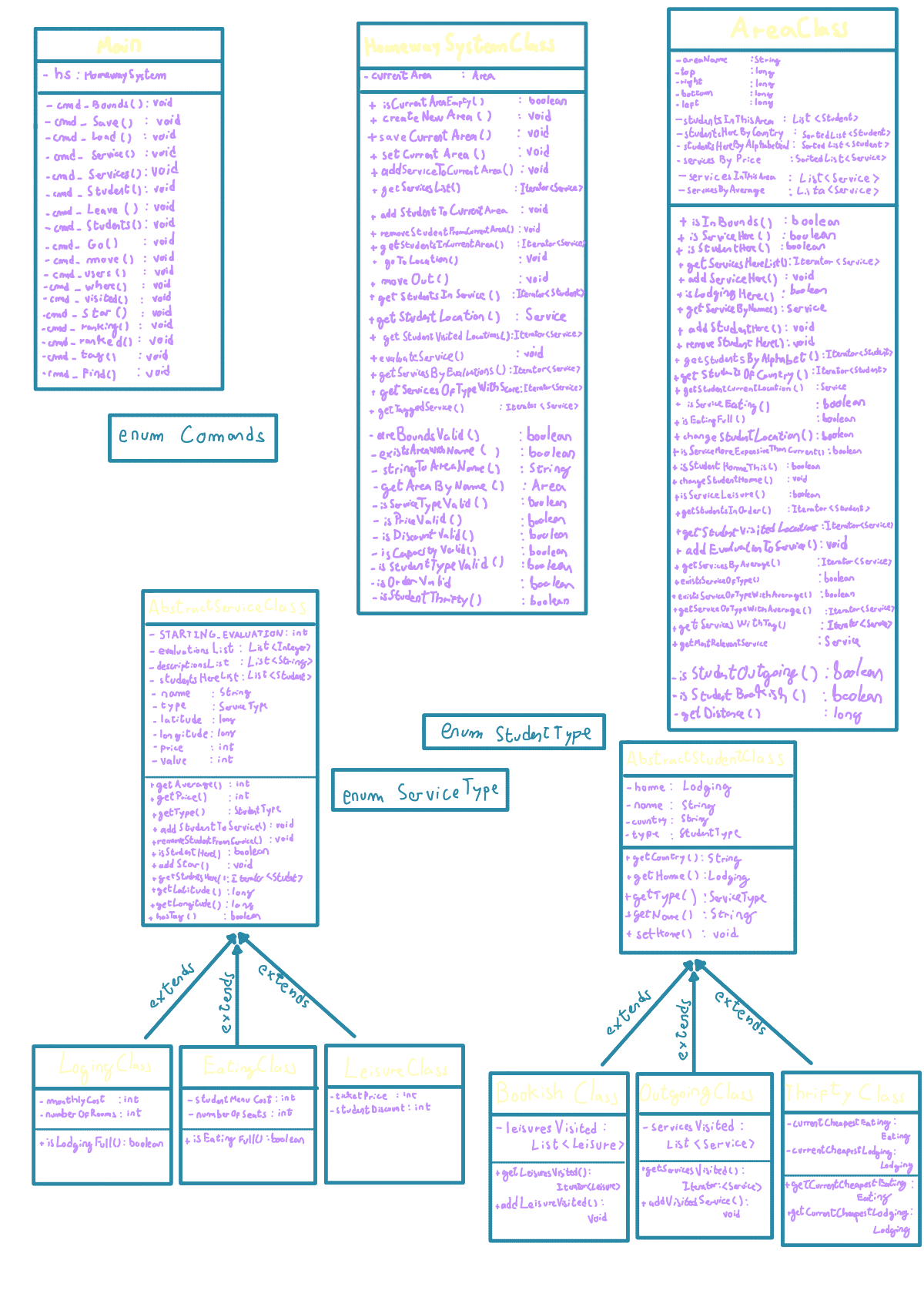
AED Projeto 1 – Diagrama de Classes

João Pereirinha 64382

miguel silva 68510



**cmd\_go:**

- HomeSystemClass - goToLocation()

- verifica se o service e student estão na currentArea com currentArea.isServiceHere e currentArea.isStudentHere respetivamente

- verifica se o estudante está a ir para um leisure ou eating com currentArea.isLodgingHere() (assume que o tipo é válido porque na 1a verificação, verificámos que a location para onde o estudante está a ir é conhecida pela currentArea)

- verifica se o estudante já está na location dada verificando se o nome da localização dada é igual ao nome do serviço onde ele está atualmente com currentArea.getStudentCurrentLocation()

- verifica se é um eating service com currentArea.isServiceEating(), e caso seja, verifica se ele está cheio com currentArea.isEatingFull() e vai buscar a EatingClass a função com o mesmo nome.

- com todas as verificações concluídas, chamamos currentArea.changeStudentLocation(), que vai retirá-lo do serviço anterior e da lista nos respetivo serviço, que é studentsHereList

- se o estudante for outgoing (isStudentOutgoing()) adicionamos à sua lista de services visitados (servicesVisited) o lodging com student.addVisitedService(lodging), assim como se for bookish (isStudentBookish()) adiciona

à sua lista de leisures visitados (leisuresVisited)

- se o estudante for thrifty (isStudentThrifty()) e o tipo da location dada for eating, chamamos currentArea.isServiceMoreExpensiveThanCurrent(), que vai comparar student.getCurrentCheapestEating().getPrice() com o preço do dado eating o sere caso seja verdade, vai imprimir "<student name> is distracted!"

**cmd\_star:**

- HomeSystemClass - evaluateService()

- verifica se n está entre 1 e 5

- verifica se o student está na currentArea com currentArea.isStudentHere()

- usa currentArea.addEvaluationTopService() que por sua vez usa service.addStar(int n, String desciption) para adicionar às listas evaluationList e descriptionList.

- atualiza servicesByAverage, recalculado a média do serviço que foi avaliado e caso mude, atualiza a sua posição na lista.

**cmd\_ranking:**

- HomeSystemClass - getServiceByEvaluation()

- chamamos o método currentArea.getServicesByAverage(), que vai devolver um iterador da lista servicesByAverage

**cmd\_ranked:**

- HomeSystemClass - getServiceOfTypeWithScore()

- verifica se n está entre 1 e 5

- verifica se o student está na currentArea com currentArea.isStudentHere()

- verifica se o tipo do serviço é válido com isTypeValid()

- verifica se existem serviços do tipo dado com currentArea.existsServiceOfType()

- verificamos na lista de servicesByAverage se existem serviços do tipo dado com n average usando currentArea.existsServiceOfTypeWithAverage()

- usa currentArea.getServiceOfTypeWithAverage()

**cmd\_tag:**

- HomeSystemClass - getTaggedServices()

- usa currentArea.getServicesWithTag(), que vai iterar StudentsInThisArea, e em cada iteração vai chamar o método service.hasTag()

**cmd\_find:**

- HomeSystemClass - findRelevatService()

- verifica se o tipo do serviço é válido com isServiceTypeValid()

- verifica se o student está na currentArea com currentArea.isStudentHere()

- verifica se existem serviços do tipo dado com currentArea.existsServiceOfType()

- usa currentArea.getMostRelevantService, onde ele vai verifica o tipo de estudante e dependendo disso, vai devolver o serviço com melhor média para estudantes bookish e outgoing e o mais barato para estudantes thrifty

**cmd\_student:**

- HomeSystemClass - addStudentToCurrent()

- verifica se o tipo é valido com isStudentTypeValid()

- verifica se o lodging dado existe na currentArea usando currentArea.isLodgingHere()

- verifica se o lodging está cheio com currentArea.getServiceByName().isLodgingFull()

- verifica se o nome já existe na currentArea com currentArea.isStudentHere()

- usa currentArea.addStudentHere(Student student, Lodging lodging) para adicionar o estudante para o adicionar às listas StudentInThisArea, StudentsHereByCountry e StudentsHereByAlphabet (usadas no comando students)

- usa lodging.addStudentToService() para colocar o student no lodging

- se o estudante for outgoing (isStudentOutgoing()) adicionamos à sua lista de services visitados (servicesVisited) o lodging com student.addVisitedService(lodging)

**cmd\_leave:**

(assume que um estudante sai de uma área, mas é e não é guardado no sistema)

- HomeSystemClass - removeStudentFromCurrentArea()

- Verifica se o estudante não existe na currentArea com !currentArea.isStudentHere()

- remove o estudante da currentArea com currentArea.removeStudentHere(), retirando-o da lista StudentsInThisArea, StudentsHereByCountry e StudentsHereByAlphabet

**cmd\_students:**

- HomeSystemClass - getStudentsInCurrentArea()

- usa currentArea.getStudentsByAlphabet caso all e currentArea.getStudentsOfCountry(String country) caso country, que vai passar neste caso apenas os estudantes do dado país

- Main - cmd\_students:

- se o iterador não tiver elementos damos print de "No students yet!" ou "No students

from <name>!" caso all ou country respetivamente

**cmd\_move:**

- HomeSystemClass - moveOut()

- verifica se o lodging e student estão na currentArea com currentArea.isLodgingHere e currentArea.isStudentHere respetivamente

- verifica se a home do estudante já é a dada com currentArea.isStudentHomeThis()

- verificamos se o lodging dado está cheio com currentArea.getServiceByName().isLodgingFull()

- se o estudante for thrifty (isStudentThrifty()) e ele tentar ir para uma location mais cara (maior que currentCheapest)

- chama currentArea.changeStudentHome() que faz student.setHome()

**cmd\_users:**

- HomeSystemClass - getStudentsInService()

- verifica se o tipo não é < ou > com isOrderValid()

- verifica se o serviço não existe no sistema com currentArea.isServiceHere()

- verifica se o serviço é eating ou lodging com !currentArea.isServiceLeisure()

- chama currentArea.getStudentsInOrder() e retorna um iterador de estudantes atravéz de service.getStudentsHere()

**cmd\_where:**

- HomeSystemClass - getStudentLocation()

- verifica se o student está na currentArea com currentArea.isStudentHere

- vamos iterar os serviços e em cada um deles chamar o método service.isStudentHere com currentArea.getStudentCurrentLocation()

**cmd\_visited:**

- HomeSystemClass - getStudentVisitedLocation()

- verifica se o student está na currentArea com currentArea.isStudentHere()

- verifica se o estudante é thrifty com isStudentThrifty()

- usa currentArea.getStuedentVisitedLocations() vai buscar o iterador de serviços visitados, caso ele seja bookish, com student.getLeisuresVisited(), e se ele for Outgoing faz student.getServicesVisited()

GERAL:

- isCurrentAreaEmpty() é público porque o main usa o método para verificar se pode correr comandos diferentes de bounds, help e quit, caso contrário, a inserção de outros comandos resulta em "System bounds not defined."

- Verificações de erro são feitas recorrendo a excepções.

- Os métodos do Main chamam cada 1 um método no HomeSystemClass

- Cada classe tem uma interface correspondente. Os nomes são os seguintes:

- Classe - Interface

- HomewaySystemClass - HomewaySystem

- AreaClass - Area

- AbstractServiceClass - Service

- LodgingClass - Lodging

- EatingClass - Eating

- LeisureClass - Leisure

- AbstractStudentClass - Student

- BookishClass - Bookish

- OutgoingClass - Outgoing

- ThriftyClass - Thrifty

LISTAS ESPECÍFICAS USADAS:

**AreaClass:**

- studentsInThisArea : DoublyLinkedList

- studentsHereByCountry : sortedDoublyLinkedList

- studentsHereByAlphabet : sortedDoublyLinkedList

- servicesInThisArea : DoublyLinkedList

- servicesByPrice : sortedDoublyLinkedList

- servicesByAverage : sortedDoublyLinkedList

**AbstractServiceClass:**

- evaluationsList : ListInArray

- descriptionsList : ListInArray

- studentsHereList : DoublyLinkedList

COMANDOS:

**cmd\_bounds:**

- HomeSystemClass - createNewArea():

- guarda a currentArea num ficheiro de texto usando saveCurrentArea()

- verifica se já há uma área com o mesmo nome na pasta "data" onde as áreas são guardadas com existsAreaByName()

- verifica se bounds sao válidos usando areBoundsValid()

- cria um objeto AreaClass, guardando-o na variável currentArea, usando setCurrentArea()

**cmd\_save:**

- HomeSystemClass - saveCurrentArea():

- transforma o nome da currentArea com espaços no formato certo com \_ usando stringToAreaName() do nome da currentArea

- guarda a currentArea num ficheiro de texto

**cmd\_load:**

- HomeSystemClass - setCurrentArea()

- guarda a currentArea num ficheiro de texto usando saveCurrentArea()

- transforma o nome da area dada com espaços no formato certo com \_ usando stringToAreaName() do nome da currentArea

- Verifica se não há uma área com o mesmo nome na pasta "data" com !existsAreaByName()

- carrega a área usando getAreaByName para a currentArea

**cmd\_service:**

- HomeSystemClass - addServiceToCurrentArea()

- verifica se o tipo é valido com isServiceTypeValid()

- verifica se a localização está dentro da currentArea usando currentArea.isInBounds()

- verifica o preço, o discount e a capacity sao erradas com isPriceValid(), isCapacityValid() e isCapacityValid() respetivamente.

- verifica se não existe um servico com o mesmo nome na currentArea com !currentArea.isServiceHere(name)

- adiciona o serviço à currentArea usando currentArea.addServiceHere(), que por sua vez adiciona o serviço às listas ServicesInThisArea e ServicesByPrice (que vai ser relevante por causa dos thrifty students)

**cmd\_services:**

- HomeSystemClass - getServicesList()

- retorna currentArea.getServicesHereList()

- Main - cmd\_services

- se a o iterador nao tiver elementos damos print de "No services yet!"