Relatório - Exercício Programa - COO

Thais De Souza Rodrigues - Número USP 11796941
Melissa Akie Inui - Número USP 11908865

1 Compilação e execução

Para realizarmos a compilação e execução, é necessário o uso do terminal. Para ocorrer a compilação, é necessário que seja digitado:

javac Main.java

E, para a execução:

java Main

Após isso, o programa será executado e seu andamento leva em consideração o que o usuário deseja fazer e as informações fornecidas por ele.

2 Implementação realizada

O presente programa implementa um gerenciador de salas de um único local. Desse modo, é importante ressaltar que uma vez inicializado o programa e o usuário colocar o nome de um local para o gerenciador, todas as reuniões e salas criadas posteriormente terão como referência o local informado anteriormente.
No menu principal do programa, localizado na classe Main, tem-se diversos comandos que o usuário faz uso e que, por conseguinte, controlará o andamento da execução do programa.
Sendo eles:

Comando S - cria uma sala de reunião. Nesse caso, é solicitado ao usuário um nome para a sala, a lotação máxima e uma breve descrição sobre a mesma.

Comando R - remove uma sala de reunião, criada anteriormente. Neste caso, solicitamos ao usuário o nome da sala a ser excluída. Caso haja reuniões marcadas para a sala que será excluída, o usuário será notificado e estas reuniões serão desmarcadas.

Comando E - reserva uma sala de reunião para um determinado horário. Neste comando, pede-se do usuário o nome da sala a ser reservada para a reunião e qual horário que ela será utilizada. Para o caso de já haver uma reunião marcada no horário informado pelo usuário, a reserva da sala de reunião não poderá ser realizada.

Comando C - cancela uma reserva, anteriormente marcada. O usuário deve informar o nome da sala que foi marcada a reserva e para qual horário esta reserva estava marcada. O cancelamento da reserva ocorrerá com sucesso caso exista uma reserva marcada com as mesmas informações obtidas do usuário.

Comando I - imprime todas as reservas de salas de reunião e suas respectivas informações (nome, capacidade máxima, observações e o horário que será utilizada). Caso ainda não tenha reservas de salas de reunião, o usuário é notificado.

Comando M - O usuário escolhe um horário que será realizado a reunião. É importante ressaltar que não é necessário que todos os participantes tenham disponibilidade no horário estabelecido. Para o caso de algum participante não ter disponibilidade no horário escolhido para a reunião, o usuário opta por escolher se quer dar prosseguimento com a marcação da reunião. Caso contrário, a reunião é marcada com sucesso.

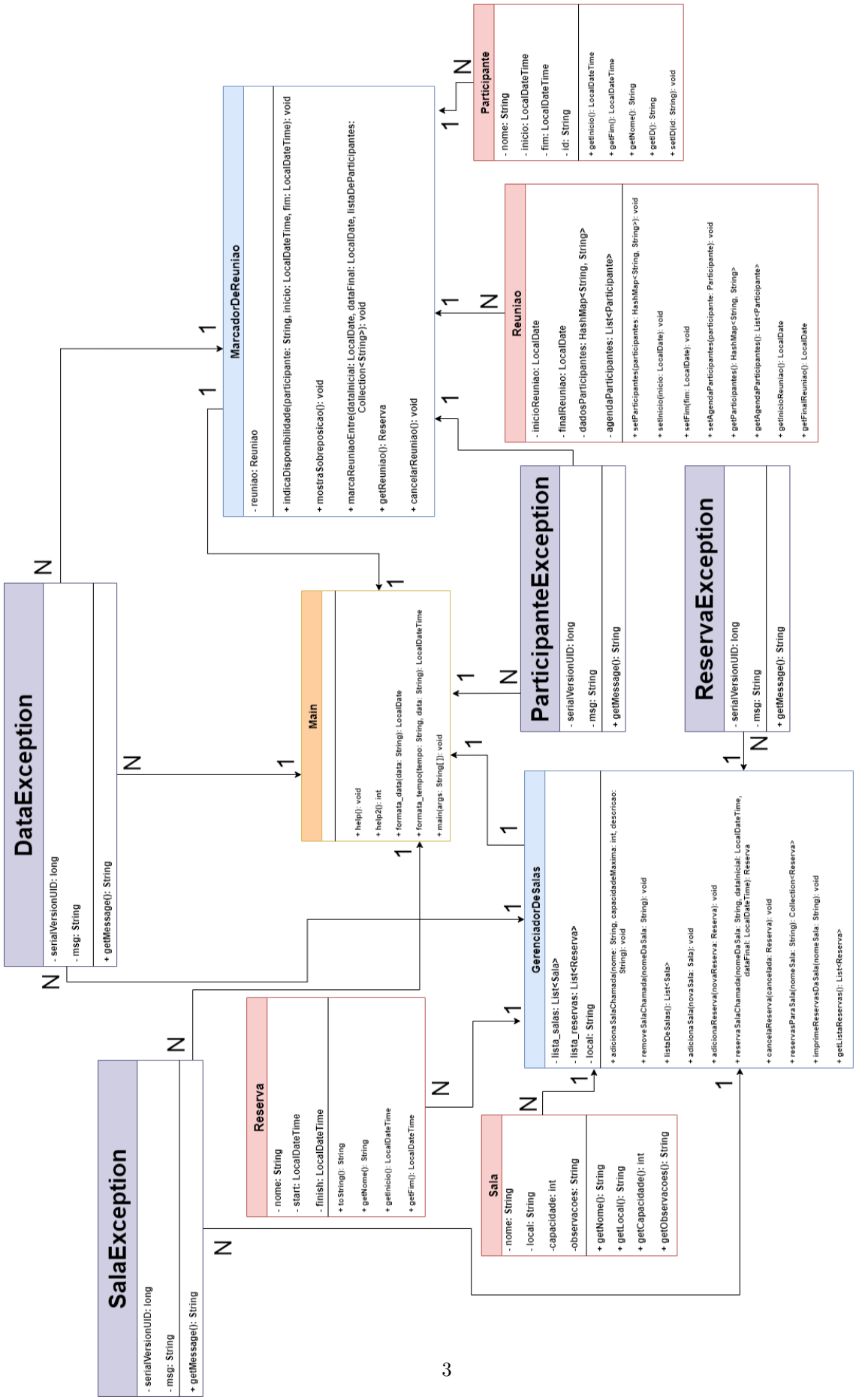
Comando O - mostra a sobreposição de horário em que todos os participantes estão disponiveis. O usuário será informado caso algum participante não tenha disponibilidade no mesmo horário que os outros participantes.

Comando P - adiciona um participante para a reunião. Para isso, o usuário deve informar o nome do participante, e o horário em que o participante está disponível.

Comando F - finaliza o programa

É importante destacar também que para cada comando, foram utilizadas as classes SalaException, ParticipanteException, DataException e ReservaException para poder lidar com eventuais respostas incorretas, fornecidas pelo usuário, em relação ao formato e padrão pedido pelo comando, e evitar que o programa pare de funcionar por este motivo. Assim, é possível que o comando em execução seja interrompido e o usuário seja redirecionado ao menu principal ou o programa pode pedir ao usuário uma nova resposta até que esta condiza com os padrões pedidos.

3 Diagrama UML



3.1 RELACIONAMENTO ENTRE AS CLASSES

O presente programa abrage 11 classes, das quais 4 são destinadas para o tratamento de exceções que o programa por ventura pode vir a lançar. Assim, cada classe GerenciadorDeSalas, MarcadorDeReuniao e Main podem se relacionar com algumas dessas 4 classes Exceptions da seguinte maneira:

- SalaException: possuí uma relação de muitos para um em relação as classes GerenciadorDeSalas e Main;
- DataException: possuí uma relação de muitos para um em relação as classes GerenciadorDeSalas, MarcadorDeReuniao e Main;
- ReservaException: possuí uma relação de muitos para um em relação a classe GerenciadorDeSalas.

Ademais, 4 classes são utilizadas como instânciadoras de objetos. São elas: Sala, Reserva, Reuniao e Participante.

- Reserva: devolve instâncias dessa classe para um GerenciadorDeSalas;
- Sala: devolve instâncias dessa classe para um GerenciadorDeSalas;
- Reuniao: devolve instâncias dessa classe para um MarcadorDeReuniao;
- Participante: devolve instâncias dessa classe para um MarcadorDeReuniao.

Já as classes GerenciadorDeSalas e MarcadorDeReuniao possuem relação de um para um em relação a classe Main. Por fim, uma descrição mais detalhada sobre o funcionamento de cada classe é dada a seguir.

3.2 Classe Main

A nossa classe Main é a classe principal do nosso programa. Nela, focamos no que o usuário deseja realizar, obter ou utilizar, bastando apenas fazer uso dos comandos solicitados pelo programa. Para os casos em que não seja inserido corretamente o que o programa solicitar, ele irá tomará atitudes. Como, por exemplo, pedir uma nova resposta do usuário até obter uma correta com os padrões solicitados, ou irá finalizar o comando que estava em processo de execução e retornar ao menu principal do programa.

main: Este método implementa todos os comandos necessários para a interação do programa com o usuário.

3.3 Classe Horário

Irá realizar a formatação das datas e horários recebidos, para que dessa forma a utilize no programa.

formata_data: Recebe uma String com o formato "xx/xx/xxxx", que representa uma data. O método a transforma em uma variavél de tipo *LocalDate* e a retorna.

formata_tempo: Recebe uma String com o formato "xx/xx/xxxx"e outra "mm:mm:mm", que representa um horário. O método utiliza ambos para gerar uma variavél de tipo *LocalDateTime* e a retorna.

3.4 Classe GerenciadorDeSalas

Na classe GerenciadorDeSalas, temos diversos métodos implementados que são responsáveis pelo gerenciamento correto das salas de reuniões e reservas destas, sendo eles:

adicionaSalaChamada: Recebe o nome de uma sala, sua capacidade máxima e uma breve descrição da mesma. Caso já exista uma sala com o mesmo nome informado a sala não é criada. Caso contrário, criamos uma instância de Sala que por conseguinte cria uma sala e, por fim, utilizamos o método *adicionaSala* para adicionar esta sala a lista de salas existentes deste gerenciador. Caso já exista uma sala, será lançada uma exceção *SalaException*, que será explicada posteriormente.

removeSalaChamada: Recebe o nome de uma sala a ser removida. Caso a sala não esteja na lista de salas existentes neste gerenciador, não é feito sua remoção e lança-se uma exceção *SalaException*, que será explicada posteriormente. Caso contrário, removemos a sala e qualquer reunião que eventualmente esteja marcada para esta sala.

listaDeSalas: Retorna uma lista de salas existentes nesse gerenciador de salas.

adicionaSala: Adiciona uma instância de Sala na lista de salas deste gerenciador.

adicionaReserva: Adiciona uma instância de Reserva de sala na lista de reservas existentes do gerenciador.

reservaSalaChamada: Cria uma reserva com as informações passadas no parâmetro do método: nome de sala e os horários de início e fim da reserva. Verifica-se, inicialmente, se a sala que deseja-se reservar existe, caso não, é lançado uma exceção SalaExeception. Após isto, é verificado se há uma reserva marcada para esta sala no horário desejado. Caso afirmativo, lança-se uma exceção *DataException*. O método retorna uma instância de Reserva, caso a reserva tenha sido criada com sucesso.

cancelaReserva: Recebe uma instância Reserva e verifica-se se esta reserva existe neste gerenciador, caso sim, remove-se a instância Reserva da lista de reservas existentes do gerenciador.

reservasParaSala: Recebe o nome de uma sala e cria uma Collection de instâncias Reserva contendo todas as reservas existentes da mesma. Por fim, o método retorna esta Collection.

imprimeReservasDaSala: Recebe uma string com o nome da sala e imprime todas as reservas existentes para esta sala, caso exista.

getListaReservas: Devolve uma lista com as reservas existentes no gerenciador.

setLocal: Recebe uma String que representa um local e aloca a variável.

3.5 Classe Reserva

Esta é uma classe que objetiva armazenar e fornecer os dados de uma reunião. Desse modo, são criados 3 atributos: um que armazena o nome da sala que a reserva será feita, e os outros dois são destinados a guardar o horário de ínicio e fim da reserva. Em relação aos métodos, temos:

toString: Retorna uma String que informa as datas e os horários que se inicia e finaliza a reunião.

getNome: Retorna o valor contido no atríbuto *nome* desta classe. Ou seja, o nome da sala que esta reserva utiliazará.

getInicio: Retorna o valor contido no atríbuto *start* desta classe. Ou seja, a data e o horário que se iniciará a reserva da sala.

getFim: Retorna o valor contido no atríbuto *finish* desta classe. Ou seja, a data e o horário que se finalizará a reserva da sala.

3.6 Classe Sala

Esta é uma classe que objetiva armazenar e fornecer os dados de uma sala de reunião. Desse modo, são criados 4 atributos: um que armazena o nome da sala, nome do local onde a sala se localiza, a capacidade máxima de pessoas que cabem na sala e uma breve observação a respeito da mesma. Em relação aos métodos, temos:

getNome: Retorna o valor do atributo *nome* desta classe. Ou seja, o nome da sala de reunião.

getLocal: Retorna o valor do atributo *local* desta classe. Ou seja, o nome do local onde a sala de reunião se localiza.

getCapacidade: Retorna o valor do atributo *capacidade* desta classe. Ou seja, número máximo de pessoas que a sala de reunião comporta.

getObservacoes: Retorna o valor do atributo *observacoes* desta classe. Ou seja, as observacoes sobre a sala de reunião.

3.7 Classe Participante

Esta é uma classe que objetiva armazenar e fornecer os dados de um participante. Desse modo, são criados 3 atributos: um que armazena o nome do participante e os horários de início e fim da disponibilidade do participante. Em relação aos métodos, temos:

getInicio: Retorna o valor do atríbuto *inicio* desta classe. Ou seja, o horário de início da disponibilidade do participante.

getFim: Retorna o valor do atríbuto *fim* desta classe. Ou seja, o horário final da disponibilidade do participante.

getNome: Retorna o valor do atríbuto *nome* desta classe. Ou seja, o nome do participante.

3.8 Classe Reuniao

Esta é uma classe que objetiva armazenar e fornecer os dados de uma reunião. Desse modo, são criados 3 atributos: uma lista ligada que armazena os nomes dos participantes da reunião, e os horários de início e fim que será realizada a reunião. Em relação aos métodos, temos:

setInicio: Atualiza o atríbuto *inicioReuniao* desta classe com o valor passado no parâmetro do método. Ou seja, o horário que se iniciará a reunião.

setFim: Atualiza o atríbuto *finalReuniao* desta classe com o valor passado no parâmetro do método. Ou seja, o horário que se finalizará a reunião.

setAgendaParticipantes: Adiciona a instância Participante, passado no parâmetro do método, no atríbuto *agendaParticipantes* desta classe. Porém, antes se verifica se o atributo já foi inicializado. Caso não, incializa-se e, logo após, é realizada a adição da instância Participante na lista *agendaParticipantes*.

getAgendaParticipantes: Primeiramente, verifica-se se a lista *agendaParticipantes* contendo instancias de Participante desta reunião já foi inicializado. Caso não, inicializamos. Logo após a veri-

ficação retorna-se essa Lista.

getInicioReuniao: Retorna o valor do atríbuto *inicioReuniao*. Ou seja, o horário que se iniciará a reunião.

getFinalReuniao: Retorna o valor do atríbuto *finalReuniao*. Ou seja, o horário que se finalizará a reunião.

3.9 Classe MarcadorDeReuniao

Esta é uma classe que objetiva marcar uma reunião e realizar outras funcionalidades a partir dela. Desse modo, é criado um unico atributo, uma instância de Reunião. Em relação aos métodos, temos:

marcarReuniaoEntre: Este método objetiva marcar uma reunião no horário e com os participantes passados no parâmetro do método. Inicialmente, verifica-se a existência de participantes. É checado em seguida, se os horários de início e fim que será realizado a reunião estão em ordem cronológica. Caso não esteja, é lançado a execeção *DataException*.
A última verificação checa se todos os participantes estão disponíveis no horário proposto para a reunião. Caso não, lança-se uma exceção *DataException*.
Por fim, atualizamos o valor do horário do atríbuto *reuniao* com os horários de inicialização e finalização da reunião que está sendo marcada.

indicaDisponibilidadeDe: Este método objetiva criar uma instância de Participante e adicionalo na lista de participantes da reunião que será marcada, levando em consideração a disponibilidade de cada participante.
Inicialmente, é verificado se o atributo *reuniao* desta classe já foi inicializado. Caso não, o inicializamos.
Caso o nome do participante passado por parâmetro já exista, será incluído. Dado que pode haver pessoas com o mesmo nome Além disso, também é checado se a ordem cronológica do horário de inicio e fim da disponibilidade do participante está correta. Caso contrário, é lançado uma exceção *DataException*, detalha posteriormente. Se não ocorrer nenhum erro, uma instância de Participante é criada e atríbuido a ela os dados do participante (nome e horário de disponibilidade).

mostraSobreposicao: Mostra entre o intervalo de horários que todos os participantes, qual horário podem estar presentes em uma futura reunião, ou seja, seus horários em comum. Caso algum participante não tenha disponibilidade comum com os outros participantes, é impresso um aviso.

cancelarReuniao: Cancela uma reunião anteriormente marcada.

getReuniao: retorna uma instância de Reuniao que é atríbuto da classe.

3.10 Classes Data Exception, ReservaException e SalaException

Estas classes objetivam o lançamento de exceções quando erros inesperados ocorrem ao longo da execução do programa.

A primeira é utilizada para erros relacionados a datas e horários como nos casos onde a data e/ou o horário final de um intervalo cronológicamente precede a data e/ou o horário do início do intervalo, ou ainda, quando uma reunião se encontra ocupada no horário que se deseja reserva-la.

A segunda tem relação com instâncias de Participante, quando ocorre de acessar uma lista de participantes vazia ou não inicializada.
ReservaException é lançada quando acessamos reservas inexistentes ou salas de reunião que não estão disponíveis para o uso no horário especificado, e, por fim, SalaException é utilizada quando procura-se por uma instância de Sala inexistente.

As trÊs classes possuem um atríbuto *msg* que guarda uma mensagem que é passada no construtor da classe e que avisa o usuário sobre uma ocorrência inesperada, de modo que o programa não finalize por conta disso. Em relação aos métodos, as quatro possuem:

getMessage: Retorna o valor do atríbuto *msg* da classe.

4 Exemplos de utilização e saídas

Estes são alguns exemplos de utilizações do programa

4.1 Caso 1 - Marcar reunião quando alguém não pode

Entrada
Terra media
P
Legolas
123456
30/07/2021 - 18:15:00 | 30/07/2021 - 23:00:00
2
P
Smeagol
145236
10/02/2021 - 18:00:00 | 18/03/2021 - 17:00:00
2
M
18/02/2021
18/03/2021
1
2
F

Saida

```
*=*=*=*=*=*=*=*=*=*=*=  INICIANDO O MARCADOR DE REUNIAO =*=*=**=*=*=*=*=*=*=*=*=*

>>>> Em qual local serao realizadas as reunioes?
======================== ESCOLHA UM COMANDO ========================
|  1. Cria uma sala de reuniao -------------------- <COMANDO>: S  |
|  2. Remover uma sala de reuniao ----------------- <COMANDO>: R  |
|  3. Reservar uma sala de reuniao ---------------- <COMANDO>: E  |
|  4. Cancelar uma reserva ------------------------ <COMANDO>: C  |
|  5. Imprimir as reservas das salas -------------- <COMANDO>: I  |
|  6. Adicionar um participante na reuniao -------- <COMANDO>: P  |
|  7. Marcar uma reuniao -------------------------- <COMANDO>: M  |
|  8. Mostrar sobreposicao de horarios ------------ <COMANDO>: O  |
|  9. Finalizar o programa ------------------------ <COMANDO>: F  |
===================================================================
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=- COLETANDO DADOS DO PARTICIPANTE -=-=-=-=-=-=-=-=
  1. Nome do participante:
  2. Crie um ID de 6 numeros:
  3. Horario de disponibilidade
Por favor, utilize o formato < dd/mm/yyyy - hh:mm:ss | dd/mm/yyyy - hh:mm:ss > :


=-=-=-=-=-=-=-=-= DADOS COLETADOS COM SUCESSO =-=-=-=-=-=-=-=-=-=

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= O PARTICIPANTE FOI ADICIONADO COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=- COLETANDO DADOS DO PARTICIPANTE -=-=-=-=-=-=-=-=
  1. Nome do participante:
  2. Crie um ID de 6 numeros:
  3. Horario de disponibilidade
Por favor, utilize o formato < dd/mm/yyyy - hh:mm:ss | dd/mm/yyyy - hh:mm:ss > :


=-=-=-=-=-=-=-=-= DADOS COLETADOS COM SUCESSO =-=-=-=-=-=-=-=-=-=

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= O PARTICIPANTE FOI ADICIONADO COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= MARCANDO UMA REUNIAO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  1. Data de inicio da reuniao < dd/mm/yyyy >:
  2. Data de termino da reuniao < dd/mm/yyyy >:
  OPS! Nao sao todos os participantes que estao disponiveis no horario da reuniao.
  >>> Gostaria de marcar a reuniao mesmo assim? Digite um dos comandos a baixo:
   SIM : 1    NAO : 2
   >>> =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= REUNIAO MARCADA COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
```

4.2 Caso 2 - Marcar reunião quando todos podem

Entrada
The Promissed Nerverland
P
Emma
123456
30/07/2021 - 18:15:00 | 30/07/2021 - 23:00:00
2
P
Norman
145236
30/07/2021 - 18:15:00 | 30/07/2021 - 23:00:00
2
M

30/07/2021
31/07/2021
2
F

Saida

```
*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=  INICIANDO O MARCADOR DE REUNIAO =*=*=**=*=*=*=*=*=*=*=*=*=*

>>>> Em qual local serao realizadas as reunioes?
======================== ESCOLHA UM COMANDO ========================
|  1. Cria uma sala de reuniao ---------------------- <COMANDO>: S  |
|  2. Remover uma sala de reuniao ------------------- <COMANDO>: R  |
|  3. Reservar uma sala de reuniao ------------------ <COMANDO>: E  |
|  4. Cancelar uma reserva -------------------------- <COMANDO>: C  |
|  5. Imprimir as reservas das salas ---------------- <COMANDO>: I  |
|  6. Adicionar um participante na reuniao ---------- <COMANDO>: P  |
|  7. Marcar uma reuniao ---------------------------- <COMANDO>: M  |
|  8. Mostrar sobreposicao de horarios -------------- <COMANDO>: O  |
|  9. Finalizar o programa -------------------------- <COMANDO>: F  |
=================================================================
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=- COLETANDO DADOS DO PARTICIPANTE -=-=-=-=-=-=-=-=
  1. Nome do participante:
  2. Crie um ID de 6 numeros:
  3. Horario de disponibilidade
Por favor, utilize o formato < dd/mm/yyyy - hh:mm:ss | dd/mm/yyyy - hh:mm:ss > :


=-=-=-=-=-=-=-=-=-= DADOS COLETADOS COM SUCESSO =-=-=-=-=-=-=-=-=-=

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= O PARTICIPANTE FOI ADICIONADO COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=- COLETANDO DADOS DO PARTICIPANTE -=-=-=-=-=-=-=-=
  1. Nome do participante:
  2. Crie um ID de 6 numeros:
  3. Horario de disponibilidade
Por favor, utilize o formato < dd/mm/yyyy - hh:mm:ss | dd/mm/yyyy - hh:mm:ss > :


=-=-=-=-=-=-=-=-=-= DADOS COLETADOS COM SUCESSO =-=-=-=-=-=-=-=-=-=

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= O PARTICIPANTE FOI ADICIONADO COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= MARCANDO UMA REUNIAO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  1. Data de inicio da reuniao < dd/mm/yyyy >:
  2. Data de termino da reuniao < dd/mm/yyyy >:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= REUNIAO MARCADA COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
```

4.3 Caso 3 - Criar e reservar sala

Entrada

Kunugigaoka
S
E-3
3
melhor sala e nao se discute
2
E
E-3
10/02/2021 - 18:00:00 | 18/03/2021 - 17:00:00
2

I
2
R
E-3
2
F

Saida

```
*=*=*=*=*=*=*=*=*=*=*=  INICIANDO O MARCADOR DE REUNIAO =*=*=**=*=*=*=*=*=*=*=*=*

>>>> Em qual local serao realizadas as reunioes?
======================= ESCOLHA UM COMANDO =======================
|  1. Cria uma sala de reuniao --------------------- <COMANDO>: S  |
|  2. Remover uma sala de reuniao ------------------ <COMANDO>: R  |
|  3. Reservar uma sala de reuniao ----------------- <COMANDO>: E  |
|  4. Cancelar uma reserva ------------------------- <COMANDO>: C  |
|  5. Imprimir as reservas das salas --------------- <COMANDO>: I  |
|  6. Adicionar um participante na reuniao --------- <COMANDO>: P  |
|  7. Marcar uma reuniao --------------------------- <COMANDO>: M  |
|  8. Mostrar sobreposicao de horarios ------------- <COMANDO>: O  |
|  9. Finalizar o programa ------------------------- <COMANDO>: F  |
====================================================================
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-=-=-=-=-= CRIANDO UMA SALA DE REUNIAO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  1. Qual sera o nome da sala?
  2. Qual eh a capacidade maxima de pessoas nessa sala?
 >>>
  3. Escreva uma breve descricao da sala em apenas uma linha:
=-=-=-=-=-=-=-=-=-=-=-=-= SALA DE REUNIAO CRIADA COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-= RESERVANDO UMA SALA DE REUNIAO =-=-=-=-=-=-=-=-=-=

1. Qual eh o nome da sala a ser reservada?
2. Qual sera o horario que a sala sera utilizada?
Por favor, utilize o seguinte formato < dd/mm/yyyy - hh:mm:ss | dd/mm/yyyy - hh:mm:ss > :

=-=-=-=-=-=-=-=-=-= RESERVA DE SALA REALIZADA COM SUCESSO =-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-=-=-=-= RESERVAS DE CADA SALA =-=-=-=-=-=-=-=-=-=-=-=
>>>>>>>>>>>>>>>>>>>>>>. SALA E-3 .<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>> Inicio: 10/2/202 as 18h0min  Fim: 18/3/202 as 17h0min
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
=-=-=-=-=-=-=-=-=-=-=-=-=-=-= REMOVENDO UMA SALA DE REUNIAO =-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  1. Digite o nome da sala que sera removida:   A sala E-3 no local  foi removida.
  A reserva da data 10/2/202 ate 18/3/202 foi cancelada.
=-=-=-=-=-=-=-=-=-=-=-=-=-= SALA DE REUNIAO REMOVIDA COM SUCESSO =-=-=-=-=-=-=-=-=-=-=-=-=-=

<<< Gostaria de rever a tabela de COMANDOS? Digite 1 ou 2 >>>
   SIM : 1    NAO : 2
   >>>
<INSIRA UM COMANDO> :
```