# MC322 – Programação Orientada a Objetos
Laboratório 04 – 2s2021
**TESTE PRÁTICO 1**

Leonardo Montecchi (Professor) Thales Eduardo Nazatto Ângelo Renato Pazin Malaguti
Leonardo de Sousas Rodrigues

Para perguntas ou dúvidas usem o Discord (`https://discord.gg/Xzpz9epNTG`)

## 1 Submissão

**Data de entrega**

- **19/09/2021** até às **23h59**.

**Submissão**

- Ao criar o projeto Java no Eclipse (ou ferramenta equivalente), selecionar **JavaSE-11** como JRE
- IMPORTANTE: Nomear o projeto na forma **RA_Lab04** e o pacote base na forma **com.unicamp.mc322.lab04**. Substitua RA com o seu *Registro Acadêmico* (matrícula).
- Submeta o trabalho no link de entrega na página do Classroom da disciplina, em formato de arquivo compactado (zip) com o nome **RA_Lab04.zip**. Substitua RA com o seu *Registro Acadêmico* (matrícula). Entregas feitas de outras formas não serão consideradas.
- O arquivo compactado deve conter o projeto inteiro ("File / Export" no Eclipse, ou crie o arquivo manualmente).

**Critérios de avaliação**

- **Este laboratório VALE nota.**

  Faz parte da avaliação saber quais classes devem ser criadas e em quais classes cada método deve ser implementado. Em particular, o código será avaliado de acordo com os seguintes aspectos:

  1. Definição de Classes (30%).
  2. Aplicação de princípios de Programação Orientada a Objetos (POO) (30%).
     - Ex: *encapsulamento* de atributos, *responsabilidades* de classes.
  3. Funcionalidades implementadas (40%).

## 2 Especificação do Sistema

Você foi contratado pela secretaria municipal de saúde para desenvolver o aplicativo *VacinaCovid*, uma solução para gerenciar o agendamento e aplicação da vacina para COVID-19. O aplicativo deve implementar as seguintes funcionalidades.

- O sistema permite o cadastro de usuários (pacientes), informando nome completo, CPF, data de nascimento e endereço. Por simplicidade, considere que o endereço é definido por um par de coordenadas $(X, Y)$. O CPF deve ser único.

- O sistema permite o cadastro de postos de saúde. Um posto de saúde possui um nome, um endereço e um máximo de vagas que podem ser atendidas por dia. Como para os usuários, também o endereço é definido por um par de coordenadas $(X, Y)$. O nome do posto deve ser único.
- Cada posto de saúde funciona para vacinação apenas em determinados dias da semana, sendo no máximo três dias da semana. Por exemplo: segunda, terça e sábado. Os dias de funcionamento do posto podem mudar ao longo do tempo.
- O sistema pode ser configurado com a faixa de idade que está sendo atualmente atendida, ex: acima de 60 anos, acima de 30 anos, etc.
- O sistema deve permitir agendar a vacinação em um posto de saúde específico, informando CPF e nome do posto. O agendamento funciona da seguinte forma:
  - Se o usuário não estiver cadastrado, nenhum agendamento é feito.
  - Se o usuário não estiver na faixa de idade atualmente atendida, nenhum agendamento é feito.
  - Se o usuário tiver já agendamento em algum posto, nenhum novo agendamento é feito.
  - Caso contrário, o agendamento é feito para o primeiro dia livre, considerando os dias da semana de funcionamento do posto, e as vagas por dia disponíveis.
  - Se o agendamento tiver sucesso, o sistema retorna um comprovante de agendamento, contendo CPF, data e nome do posto. O comprovante de agendamento não pode ser alterado.
- O sistema deve permitir agendar a vacinação no posto de saúde *mais perto do endereço do usuário* (considere a distância em linha reta), informando apenas o CPF. O agendamento funciona com as mesmas regas acima, com a diferência que o posto de saúde é determinado pelo sistema com base na distância.
- O sistema deve permitir agendar a vacinação no posto de saúde *que tiver a vaga mais cedo possível*, informando apenas o CPF. O agendamento funciona com as mesmas regas acima, com a diferência que o posto de saúde é determinado pelo sistema com base na primeira data livre.
- O sistema deve permitir imprimir os detalhes de todos os postos de saúde, incluíndo: nome, endereço, dias de funcionamento, vagas livres para os próximos 7 dias.
- O sistema deve permitir imprimir os detalhes de cadastro de todos os usuários.

## 3 Exemplo de fluxo de execução

Considere o seguinte cenário como um exemplo de fluxo de execução. Esse código deve ser considerado como uma das possíveis execuções do programa. Porém, ele também define um contrato do que deve ser implementado: deve ser possível rodar esse trecho de código na sua solução. É permitido alterar o nome das classes, por exemplo utilizar uma classe diferente para datas.

```
1 public class Runner {
2
3   public static void main(String[] args) {
4
5         VacinaCovid app = new VacinaCovid();
6         app.setIdadeMinimaAtendida(60);
7
```

```
8         app.cadastrarUsuario("Jose da Silva", "123.456.789-01",
9                              new Date(1960,12,03), new Position(10,30));
10        app.cadastrarUsuario("Maria Assuncao", "321.654.987-10",
11                             new Date(1999,4,11), new Position(-43,101));
12
13        app.cadastrarPosto("SOUSAS", new Position(0,20), 5);
14        app.cadastrarPosto("BARAO GERALDO", new Position(-20,40), 2);
15
16        Reserva r1 = app.agendar("123.456.789-01", "SOUSAS");
17        Reserva r2 = app.agendarPerto("321.654.987-10");
18
19        app.imprimirSituacaoPostos();
20
21        app.imprimirCadastroUsuarios();
22
23    }
24
25 }
```

## 4 Observações

- Existem várias formas de representar datas e dias da semana em Java, por exemplo as classes:
  - `java.time.ZonedDateTime`
  - `java.util.Date`

  Contudo, **não é necessário usar essas classes**, uma solução simples é implementar a própria classe de gerenciamento das datas.
- Tudo o que não está especificado no texto é deixado livre. Caso ache necessário, justifique eventais escolhas com comentários no código.

## 5 Boas Práticas

- Nomes de classes devem começar com letra maiúscula;
- Nomes de variáveis e métodos devem começar com letra minúscula;
- Nomes de classes devem ser substantivos;
- Nomes de métodos devem ser verbos ou começar com verbo;
- Nomes compostos por mais de uma palavra devem ser escritos na forma *CamelCase*. Isto é, com a primeira letra de cada palavra maiúscula. Por exemplo: "get something" deve ser escrito como `getSomething` (método), e uma classe que representa *something* poderia ser declarada como `Something`.