# Trabalho – Parte 1

| MC322 – Programação Orientada a Objetos<br>Instituto de Computação<br>Universidade Estadual de Campinas | | Diagnosticando Doenças<br>André Santanchè<br>2019.1 |
|---|---|---|

## Instruções

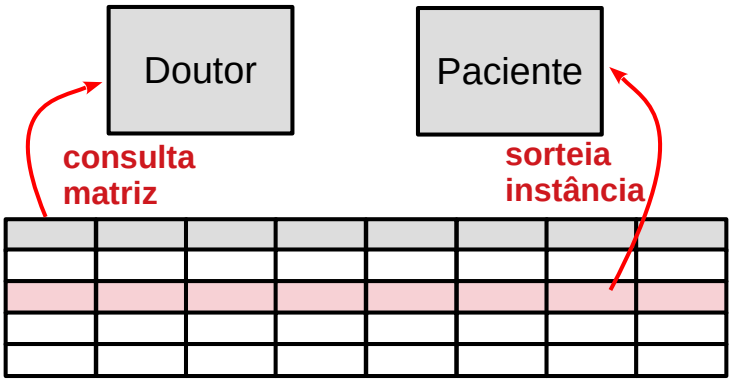
Este trabalho consiste na implementação de programa em Java capaz de simular o comportamento de um médico diagnosticando a doença de um paciente no contexto do Zombie Health, que é uma extensão do problema apresentado no Roteiro 2 em: https://github.com/santanche/java2learn/tree/development/notebooks/pt/c03oo-zombie.

## Especificações

O `Roteiro 2` solicita que sejam escritas duas classes que representem dois zumbis. O primeiro zumbi está doente (`Paciente`) e o segundo é um médico (`Doutor`). Ambos têm acesso a uma matriz sintomas/diagnóstico recuperada a partir do arquivo CSV: `/notebooks/db/zombie/zombie-health-spreadsheet-ml-training.csv`

O `Paciente` sorteia uma linha da matriz de sintomas/diagnóstico e esta será a sua doença (vide figura). O `Doutor` não sabe que doença foi sorteada pelo `Paciente`.

O `Doutor` deve fazer perguntas ao `Paciente` para diagnosticar sua doença apenas a partir dos sintomas. As perguntas podem ser simples, por exemplo, a pergunta pode ser apenas o nome do atributo do sintoma (por exemplo, `chest_pain` ou `trembling_finger`). A resposta pode se limitar a verdadeiro ou falso. Para o diagnóstico, o `Doutor` tem acesso a toda a matriz sintomas/diagnóstico (vide figura). Há casos em que o mesmo conjunto de sintomas leva a mais de uma doença; nesse caso, o `Doutor` deve diagnosticar todas as doenças possíveis.




### Abordagem de Componentes

Leia o Roteiro 06 que está no diretório:
https://github.com/santanche/java2learn/blob/master/notebooks/pt/c03oo-zombie/

Nele você encontrará orientações sobre como criar e usar componentes no Zombie Health. Você deve usar essa abordagem para o que será tratado a seguir.

### Entrevistado / Entrevistador

Para que o sistema cumpra com seu objetivo, é necessário criar dois componentes: um entrevistado (que será o `Paciente`) e um entrevistador (que será o `Doutor`). Para estes dois elementos foram definidas três interfaces. `IResponder` é implementado pelo `Paciente` e especifica dois métodos que podem ser acessados publicamente:

1) `public String ask(String question)` → Este método recebe como parâmetro uma string correspondente a alguma propriedade que pode ou não estar presente no arquivo que descreve o animal escolhido pelo entrevistado, e retorna uma string contendo um de três valores possíveis: "`yes`", "`no`" ou

“unknown”. As duas primeiras respostas podem ser inferidas caso a propriedade se encontre no arquivo do animal em questão. A terceira resposta é utilizada caso contrário.

2) `public boolean finalAnswer(String answer)` → Este método pode ser invocado apenas uma vez, recebendo como parâmetro uma string identificando a doença que o entrevistador julga que o entrevistado tem. O método retorna um valor booleano indicando o acerto ou erro por parte do entrevistador. Este método só poderá ser acionado uma única vez ao final da entrevista.

As duas outras interfaces servem para o `Doutor` se conectar ao entrevistado (`IResponderReceptacle`) e entrevistá-lo (`IEnquirer`):

`IResponderReceptacle`

- `public void connect(IResponder responder)` → Esse método recebe como parâmetro um objeto que implemente a interface `IResponder`, e tem como único objetivo conectar o entrevistador ao entrevistado

`IEnquirer`

- `public void startInterview()` → Esse método só funcionará se houver um entrevistado conectado. Ele inicia a sequência de perguntas que deve levar o entrevistador a identificar qual a doença que entrevistado tem. Ao final, esse método deve se utilizar do método `finalAnswer` disponível em `IResponder` para verificar o acerto ou o erro por parte do entrevistador.

Há uma implementação de referência em: https://github.com/santanche/java2learn/tree/master/notebooks/pt/c03oo-zombie/s05template

Cada equipe deverá criar um repositório GIT para a sua implementação e submeter o endereço em tarefa específica no Classroom até o dia 23/04.

Este trabalho é dividido em duas partes:

# Parte 1

Nessa parte você deve aumentar a capacidade do entrevistador em duas direções:

a) se tornando mais generalista → o entrevistador deve se tornar capaz de diagnosticar doenças com quaisquer sintomas e com outros pacientes além daqueles que você conhece na base;

b) se tornando mais eficiente → há formas mais eficientes de diagnosticar a doença sem que seja necessário se fazer todas as perguntas; nesse caso você deve buscar estratégias para fazer isso.

Para essas duas funcionalidades, recomenda-se que se projetem componentes que se conectem ao entrevistador dando-lhe funcionalidades extra. Por exemplo: componentes que reorganizam o armazenamento dos dados para torná-los mais eficientes; componentes que criam estruturas de dados para diagnóstico mais eficiente etc.

### Duplas

Esta parte do trabalho deve ser feita e submetida em duplas.

### Aula de Treinamento

No dia 23/04 haverá uma aula de treinamento para a sua solução da parte 1. O professor acompanhado do PED e do PAD da disciplina implementarão “secretamente” um objeto com a interface `IResponder` que acessa tabelas que você não conhece, da seguinte maneira:

- tabela com o mesmo layout da que você tem acesso, mas com novos pacientes;
- tabela com sintomas e pacientes diferentes da que você conhece, mas com a mesma lógica (diversas colunas de sintomas com verdadeiro e falso e no final uma coluna `diagnostic` com o diagnóstico).

Espera-se que você tenha uma primeira versão do `IEnquirer` fazendo uso de estruturas de dados em Java para melhor estruturar o projeto com desempenho satisfatório.

Lembre-se que o `IEnquirer` pode (e deve) fazer uso das implementações já existentes para manipular os arquivos `CSV` conforme julgar necessário. Pode ser utilizada qualquer abordagem para a escolha das perguntas a serem feitas ao `IResponder`. A única restrição existente é que não é permitido

fazer uma mesma pergunta duas vezes. Vale lembrar que o `IResponder` não trapaceia: ou seja, ele é sempre sincero ao responder as perguntas recebidas.

A base `CSV` contendo os sintomas/doenças só será revelada no dia 23/04. Para facilitar os testes, sugerimos que os alunos enviem suas bases de exemplo para o repositório git online, assim é possível compartilhar a base e melhorar a qualidade dos testes.

# Parte 2 – ZombieVersum

Tratando o problema de uma forma mais ampla, é possível se expandir o cenário do diagnóstico criando-se componentes conectáveis em diversas direções:

- Visualização – componentes que apresentam graficamente alguma análise sobre as doenças.
- Pacientes – componentes que representam novos modelos de pacientes que respondem a `IResponder`. Por exemplo:
  - componentes que respondem a questões em linguagem natural;
  - componentes que respondem perguntas mais sofisticadas (não somente sim/não);
  - componentes que respondem perguntas realistas acessando outros tipos de bases de sintomas/doenças.
- Doutores – podem ser criados componentes doutores mais espertos, seja com o que foi feito na parte 1, seja integrando novos recursos como:
  - componentes de aprendizagem de máquina;
  - componentes que tratam doenças realistas por acesso a bases externas que ajudem em um diagnóstico.

### Equipes

Para essa parte, as duplas se juntarão em equipes maiores de no máximo seis componentes, entretanto, cada dupla precisa ficar responsável completamente por alguns componentes dentro da equipe.

### Apresentação de Ideias de Componentes

No dia 23/04 cada equipe (seis componentes) pode falar 3 minutos para apresentar as ideias de componentes que teve. Trata-se de uma breve apresentação somente com as ideias.

### Mercado de Componentes

No dia 30/04 até as 10h cada equipe deve escolher alguns de seus componentes para publicar no Mercado de Componentes. A ideia é que esses componentes possam ser reusados por outras equipes. A publicação consiste em uma descrição do componente, acompanhada de suas interfaces providas e requeridas.

No dia 30/04 na aula de laboratório será feito em sala um mercado de componentes em que eles serão negociados.

As interfaces providas e requeridas dos componentes e alguns aspectos da sua funcionalidade podem ser adaptados durante ou depois do mercado para que eles sejam capazes de se casar com componentes produzidos por outras equipes. Antes e depois do mercado, é permitida a propaganda e o debate entre as equipes no Classroom sobre componentes que estão sendo desenvolvidos, para que haja um alinhamento de interesses.

Esta etapa será avaliada conforme:

- o número de componentes que a equipe conseguir comprar e/ou vender no mercado;
- o número de diferentes equipes das quais se conseguir comprar e/ou vender.