C206 - Programação Orientada a Objetos
Fabio de Moura Camargo Neto
github.com/bio353

Relatório Prático 03 - II

Orientações

- Esta avaliação deverá ser realizada de forma individual e possui tempo de duração igual a 1h40.
    - Quaisquer tentativas de comunicação entre colegas durante a realização desta prova resultará no zeramento da nota da avaliação de todos os envolvidos.
- É permitida a consulta de nosso material de sala, incluindo **o GitHub da disciplina**, além da **documentação do Java**, mas não será tolerada, ***de forma alguma***, a consulta com outros colegas.
- Deverá ser entregue, até às 23h10 do dia 18/11/2022, pelo Teams, um arquivo compactado (`.zip`, `.rar` ou `.7z`) com todo o código da avaliação ou o *link* para um repositório no GitHub.
    - Entregas fora do prazo ou em outros formatos serão penalizadas.
    - Caso esteja enviando um arquivo, peço para que coloque seu nome e matrícula no nome dele de alguma maneira. Por exemplo, colocando o nome `Fabio_22.zip`.
- Durante a avaliação, estarei disponível para orientá-los em caso de mau funcionamento dos equipamentos, mas não irei tirar dúvidas quanto a como resolver as questões.
- Esta prova vale 100 pontos, onde 30 dizem respeito à implementação do diagrama, 30 ao uso correto de exceções, 30 à gravação e leitura de arquivo e 10 para a instanciação de objetos e uso de métodos.

Exercício “Scooby-Doo”

Introdução

Hoje, a prova é sobre **Scooby-Doo**! A Gangue do Mistério precisa da sua ajuda…

Você, como bom programador, precisa ajudar a turma do Scooby-Doo a resolver mais um mistério: como criar um bom programa em Java para auxiliá-los a resolver o maior número de mistérios possível.

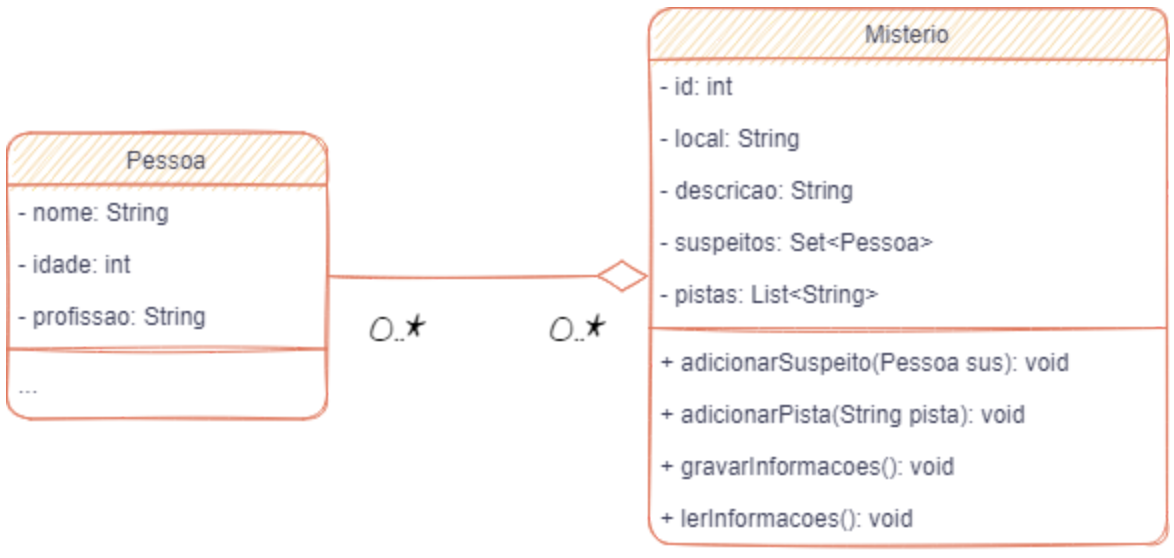
Em nosso sistema, temos salvas as informações de várias **Pessoas**. Algumas delas, podem ser atribuídas como suspeitas em um **Mistério**.

Todo **Mistério** possui um número único de identificação, um lugar onde ocorreu, uma descrição do ocorrido e um conjuto de suspeitos, além de uma lista de pistas que podem ser anexadas a qualquer momento.

No fim do programa, deverá ser possível ler e escrever *logs* sobre **Mistérios**.

Diagrama de Classes

Implemente o diagrama de classes abaixo. Atenção às especificações logo em seguida!



Especificações do Diagrama

Atentem-se às especificações abaixo:

a. Para o registro de cada suspeito, implemente um `HashSet`. Para cada pista, implemente um `ArrayList`.

b. O método `adicionarSuspeito` deverá adicionar um suspeito ao conjunto, mas deverá lançar uma exceção customizada caso o suspeito já tenha sido adicionado.

c. O método `gravarInformacoes` deverá gravar todas informações do **Mistério** um arquivo de texto, ou seja, no formato `.txt`. O local dos arquivos deverá ser dentro de uma pasta chamada “mysteries”, em que o nome do arquivo poderá ser o próprio ID do **Mistério**.

d. Neste mesmo arquivo de texto, deverão estar listadas todas as informações do **Mistério**, como o local onde ocorreu e a descrição do ocorrido, além de listar cada um dos suspeitos com suas respectivas informações e mostrar uma lista com a pistas.

e. O método `lerInformacoes` deverá ler o arquivo de texto daquele daquele **Mistério** específico e mostrar suas informações na tela.

f. Para gravar e ler essas informações, caso prefira, poderá estar utilizando a classe `FileManager`, anexada junto à prova. Mas cuidado! Ela está incompleta…

g. Por fim, crie uma classe principal, nomeada da forma que preferir, como `Main` ou `Principal`, por exemplo, onde deverá estar o método `main`. Dentro deste método `main`, instancie alguns objetos: crie algumas **Pessoas** e crie um **Mistério**, para então utilizar todas as funções implementadas no programa.

Observações Gerais

Algumas dicas para vocês:

a. Façam bom uso de *getters* e *setters*! Não é necessário termos um de cada para todos os atributos. Usem métodos construtores se não planejam alterar aquele atributo com frequência ou caso seja um atributo essencial para todas as instâncias daquela classe.

b. Atentem-se às exceções do tipo *checked*, pois são elas que impedem até mesmo que o código do programa seja compilado!

c. Não é necessário entrar com dados manualmente, ou seja, não é necessário o uso da classe `Scanner` para entrada e saída de dados.

d. Qualquer exceção lançada não deverá interromper o funcionamento do código! Por isso, trate-as devidamente!

e. Boa prova! 🙂