POO - JAVA

INDICES

1. PILARES
2. ASSOCIAÇÕES
3. DIAGRAMA UML
4. EXERCÍCIOS

ABSTRAÇÃO

Organiza o código em torno de objetos, que representam entidades do mundo real com atributos e métodos.

- Foco em modelar comportamentos e responsabilidades
- Melhora reutilização e manutenção do código

ENCAPSULAMENTO

O Encapsulamento protege os dados internos do objeto e controla seu acesso.

- Garante segurança e integridade dos dados
- Usa métodos getters e setters para manipulação controlada

HERANÇA

A Herança permite que uma classe herde atributos e métodos de outra.

- Promove reuso de código
- Define uma relação “é um” entre classes

POLIMORFISMO

O Polimorfismo permite que diferentes classes respondam de maneiras distintas ao mesmo método.

- “Muitas formas” de um mesmo comportamento
- Facilita extensibilidade do sistema

ASSOCIAÇÕES

1. RELACIONAMENTOS
2. AGREGAÇÃO
3. COMPOSIÇÃO

AGREGAÇÃO

Organiza o código em torno de objetos, que representam entidades do mundo real com atributos e métodos.

- Foco em modelar comportamentos e responsabilidades
- Melhora reutilização e manutenção do código

COMPOSIÇÃO

O Encapsulamento protege os dados internos do objeto e controla seu acesso.

- Garante segurança e integridade dos dados
- Usa métodos getters e setters para manipulação controlada

COLLECTIONS

1. ARRAYS
2. HASH SET
3. HASH MAP
4. LINKED HASH SET

ARRAY LIST

Sequência ordenada de elementos. Permite elementos duplicados. Acesso rápido a elementos por índice (.get(i)). Listas de itens onde a ordem de inserção é importante (ex: histórico de pedidos, lista de alunos por chamada).

Exemplo: ArrayList<String> nomes = new ArrayList<>();

HASH SET

O HashSet é a implementação mais performática da interface Set. Sua principal característica é não permitir elementos duplicados, garantindo a unicidade de cada item. Ele não mantém a ordem de inserção, pois a organização interna é baseada no hash code dos objetos. É a escolha ideal para coleções onde apenas a presença única do elemento importa.

Exemplo de declaração: HashSet<Integer> ids = new HashSet<>();

HASH MAP

O HashMap armazena dados no formato chave-valor, sendo a implementação mais comum da interface Map. A chave deve ser única, servindo como um índice ultrarrápido para recuperar o valor associado (.get(chave)). O acesso e a manipulação são extremamente rápidos, mas a ordem de inserção não é preservada. É perfeito para mapeamentos e dicionários (ex: CPF: Pessoa).

Exemplo de declaração: HashMap<String, Livro> catalogo = new HashMap<>();

LINKED HASH SET

O LinkedHashSet combina as características de outras Collections: garante a unicidade dos elementos (como o HashSet), mas mantém a ordem exata de inserção (como o ArrayList). Sua organização interna é feita por hashing com uma lista ligada que preserva a sequência. É útil quando você precisa de unicidade sem abrir mão da ordem cronológica de como os itens foram adicionados.

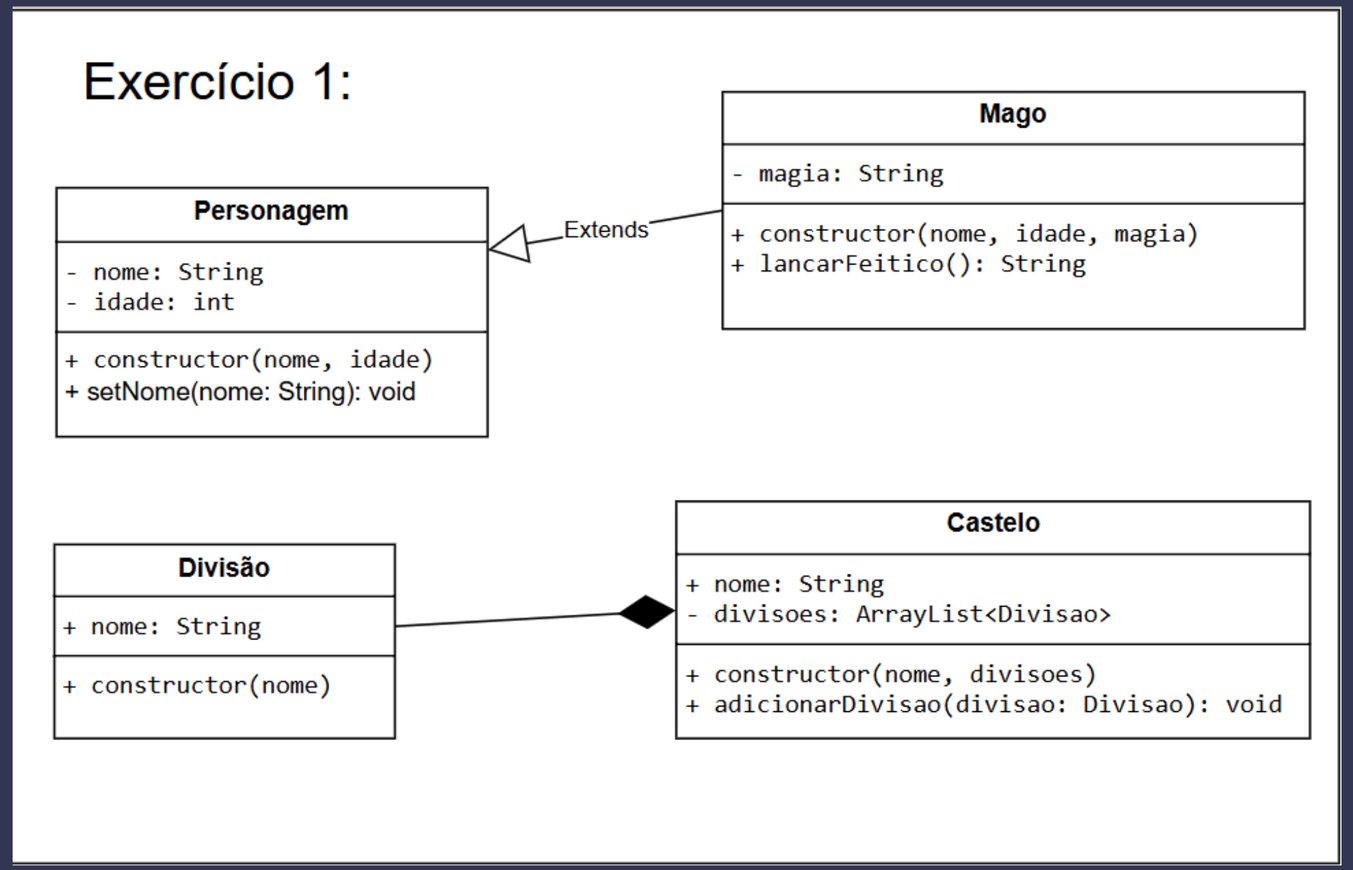
Exemplo de declaração: LinkedHashSet<String> logs = new LinkedHashSet<>();

EXERCÍCIOS

EXERCÍCIO 1

CENÁRIO: O CASTELO ANIMADO É UMA ESTRUTURA QUE AGREGA VÁRIAS DIVISOES. PERSONAGENS COMO O MAGO HOWL, QUE PODEM HABITÁ-LOS, HERDAM CARACTERÍSTICAS BASE DE PERSONAGEM COMUNS.

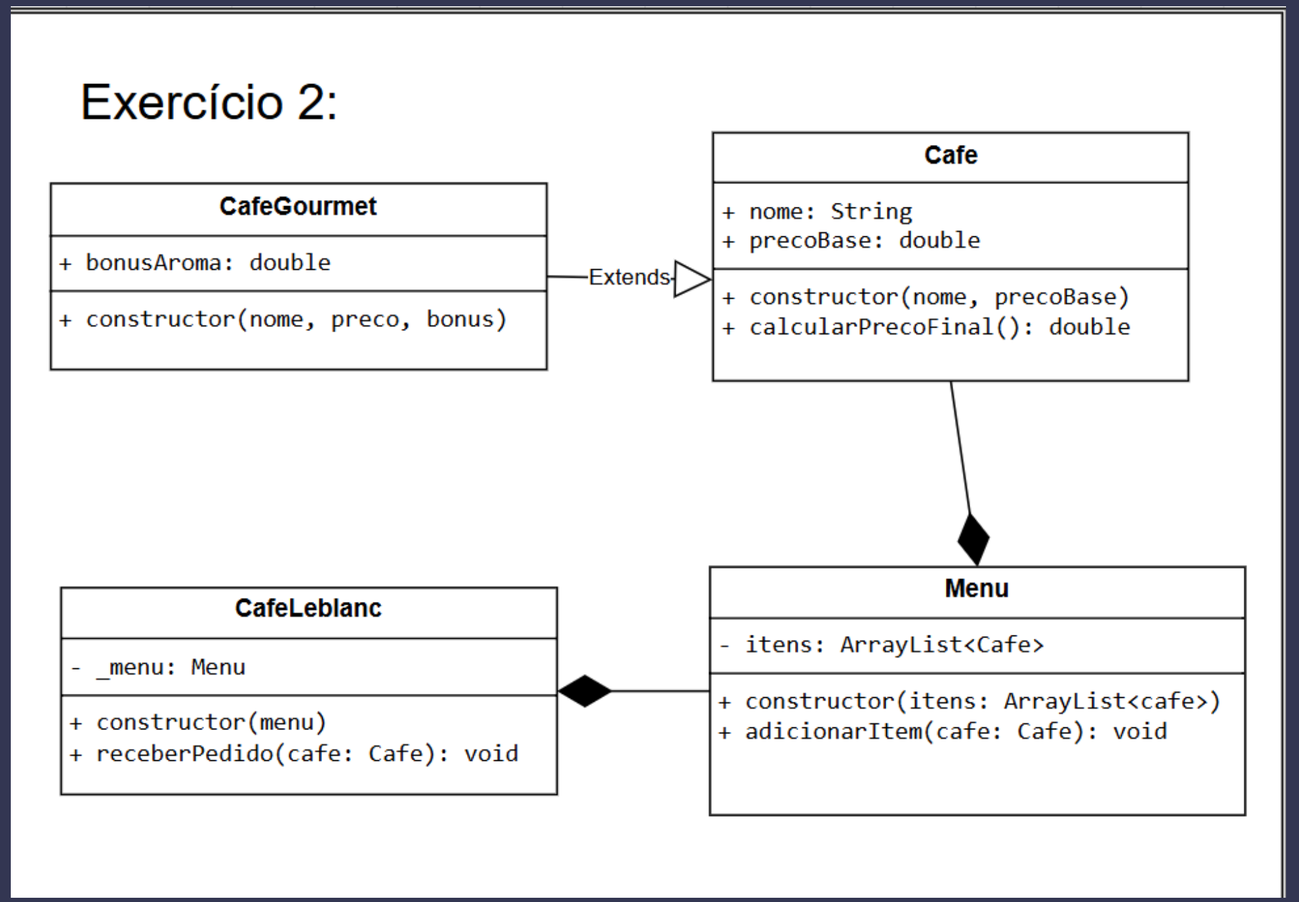
1. IMPLEMENTE AS CLASSES DO DIAGRAMA (MAGO, PERSONAGEM).
2. USE HERANÇA E ENCAPSULAMENTO (PRIVATE + GETTERS/SETTERS)
3. IMPLEMENTE A RELAÇÃO DE AGREGAÇÃO NO CASTELO, QUE DEVE USAR ARRAYLIST<DIVISAO> PARA GERENCIAR AS DIVISÕES.
4. IMPLEMENTE O MÉTODO ADICIONARDIVISAO(DIVISAO: DIVISAO) NO CASTELO.



EXERCÍCIO 2

CENÁRIO: O PROPRIETARIO (SOJIRO) ADMINISTRA O MENU DA CAFETERIA LEBLANC. O SISTEMA DEVE DIFERENCIAR OS TIPOS DE CAFÉ SERVIDOS.

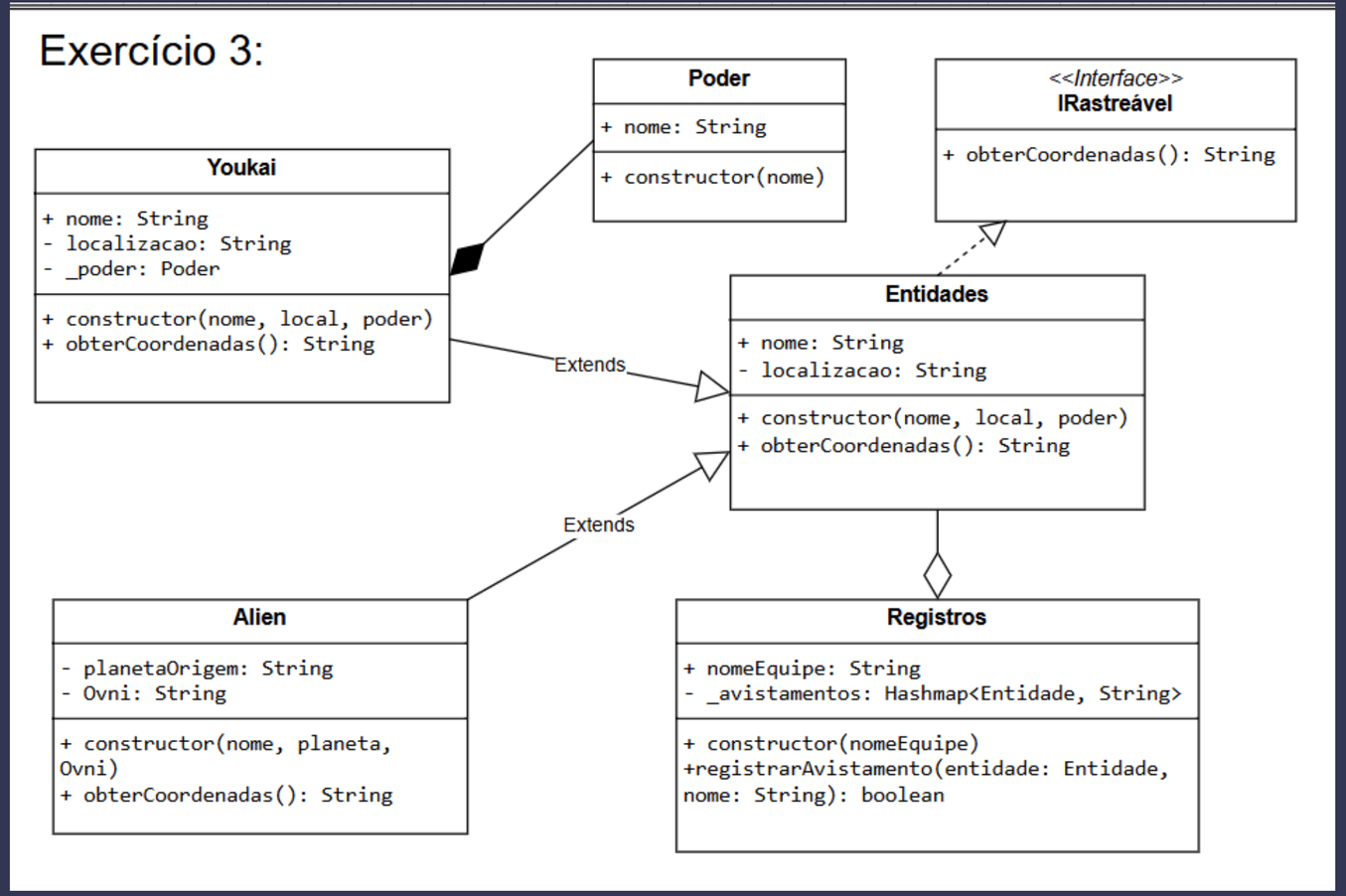
1. IMPLEMENTE AS CLASSES CAFE E CAFEGOURMET. USE HERANÇA PARA ESPECIALIZAR OS TIPOS DE CAFÉ.
2. USE POLIMORFISMO (@OVERRIDE) PARA QUE CAFEGOURMET.CALCULARPRECOFINAL() ADICIONE UM BONUSAROMA.
3. IMPLEMENTE COMPOSIÇÃO FORÇANDO O MENU A POSSUIR CAFÉS E A CAFETERIA A POSSUIR UM MENU
4. O MENU DEVE GERENCIAR OS ITENS DISPONÍVEIS UTILIZANDO ARRAYLIST<CAFE> PARA MANTER A ORDEM DE EXIBIÇÃO.



EXERCÍCIO 3

CENÁRIO: ESTÃO OCORRENDO INVSÕES POR TODO O MUNDO DE ALIENS E YOUKAI. A EQUIPE DE DETETIVES OCULTOS DEVE REGISTRAR TODAS AS ENTIDADES AVISTADAS. CADA ENTIDADE (ALIEN OU YOUKAI) TEM UM TIPO DE PODER (COMPONENTE) E DEVE SER RASTREÁVEL.

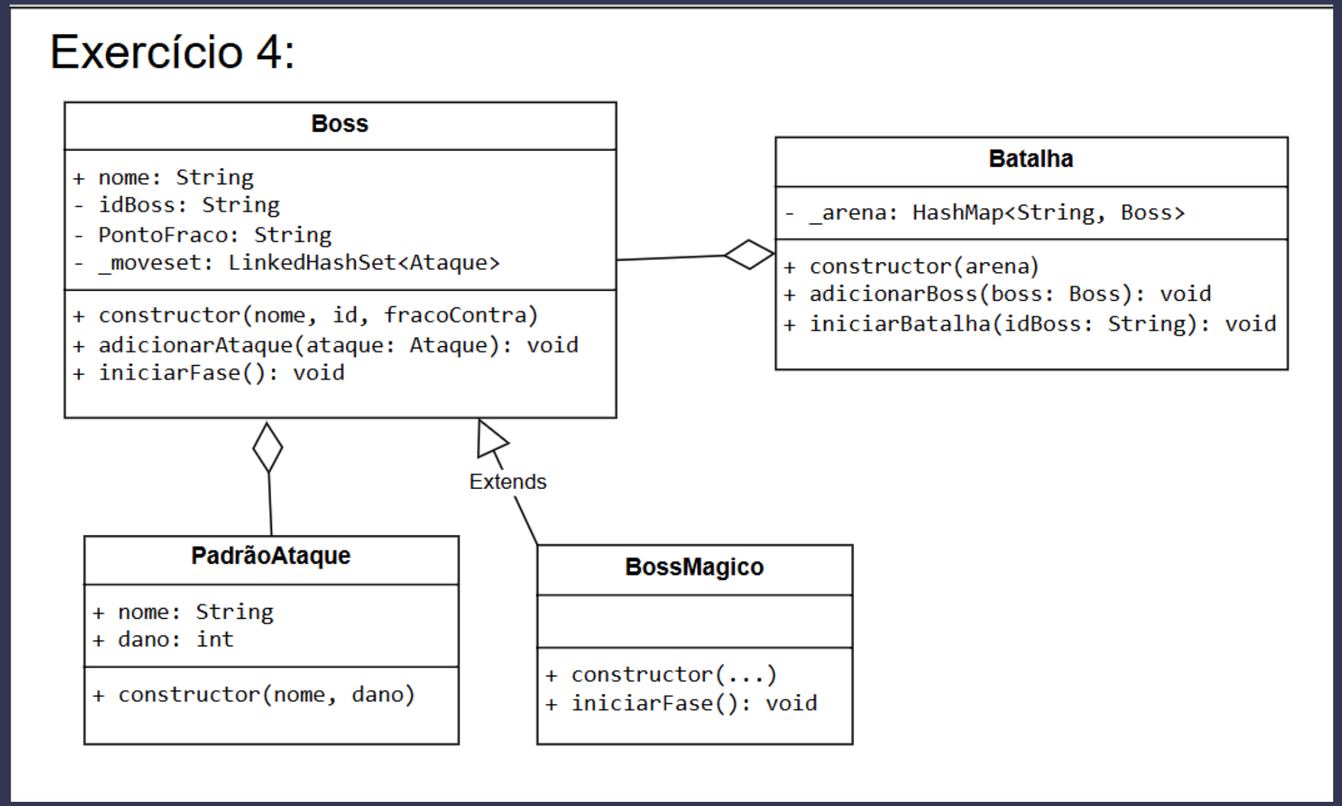
1. CRIE A INTERFACE RASTREAVEL COM O MÉTODO OBTERCOORDENADAS().
2. ALIEN E YOUKAI DEVEM IMPLEMENTAR A INTERFACE RASTREAVEL.
3. O PODER DEVE SER CRIADO POR COMPOSIÇÃO DENTRO DO ALIEN OU YOUKAI.
4. O REGISTOOCULTO DEVE USAR HASHMAP<> PARA GARANTIR QUE CADA ENTIDADE AVISTADA (IDENTIFICADA PELO NOME) SEJA ADICIONADA APENAS UMA VEZ.



EXERCÍCIO 4

CENÁRIO: UM ENCONTRO (ARENA) ORGANIZA VÁRIOS BOSSES (COMO O BOSSMAGICO), QUE POSSUEM UM PONTOFRACO E UM MOVESET DEFINIDOS.

1. USE COMPOSIÇÃO PARA QUE O BOSS POSSUA UM PADRÃO DE ATAQUES ARMAZENADO COMO LINKED HASH SET E HERANÇA PARA CRIAR O BOSSMAGICO.
2. IMPLEMENTE A AGREGAÇÃO NA BATALHA, QUE DEVE USAR HASHMAP<STRING, BOSS> PARA MAPEAR O ID DO BOSS À ARENA EM QUE OCORRERÁ A BATALHA DESTE BOSS).
3. O MÉTODO INICIARBATALHA(IDBOSS) DO ENCONTRO DEVE USAR O HASHMAP PARA LOCALIZAR O BOSS.



https://github.com/zsh3rlock/so1-monitoria