Uma imagem com texto

Descrição gerada automaticamente

**Faculdade de Ciências Exatas e de Engenharia**

2022/2023

**Programação Orientada por Objetos**



**Trabalho elaborado por:**

Diogo Miguel Paixão - 2079921

José Gonçalves - 2112821

Marcos Fernandes – 2041518

Funchal, 19 de Dezembro de 2022

**Índice**

[***1. Introdução***](#_heading=h.gjdgxs) ***4***

[***2. Ideia principal do projeto***](#_heading=h.30j0zll) ***4***

[***3. Procedimento e implementação do código***](#_heading=h.1fob9te) ***4***

[3.1. Classes](#_heading=h.bnk2e0mv3dum) 4

[3.1.1. Classes Animais](#_heading=h.56x3uv76px5o) 4

[3.1.1.1. Animal](#_heading=h.rvucp0m1hl7d) 4

[3.1.1.2. Subclasses de Animal](#_heading=h.2jcuf8zan9nx) 6

[3.1.2. Classes Genomas](#_heading=h.blcg0b6fekgm) 7

[3.1.3. Classes Mutações](#_heading=h.zf6pzl79b451) 7

[3.1.4. Instalação](#_heading=h.oi06rn28rktg) 7

[3.1.5. Zoo](#_heading=h.topuv5fxgvbg) 8

[3.1.6. Classes Ocorrências](#_heading=h.dz5vzw2ia8mp) 14

[3.1.7. Classes Exceções](#_heading=h.vqn1m9fuvp7r) 14

[3.1.8. Ficheiros (Jumanji/Main)](#_heading=h.xaf8z50v7ai) 15

[**4. Conclusão**](#_heading=h.cngk2r7nbxr1) **16**

[**5. Anexo do código**](#_heading=h.v7mn0pfa66bn) **16**

[5.1. Animal](#_heading=h.j0akrhnuqdoq) 16

[5.2. Boi](#_heading=h.5d1m1iofb9jc) 20

[5.3. Cabra](#_heading=h.etfjsf5xgw3m) 22

[5.4. Cachorro](#_heading=h.8d0rnljv7ryy) 24

[5.5. Cavalo](#_heading=h.hlvljnoszx43) 26

[5.6. Cobra](#_heading=h.ev8cspzdritu) 28

[5.7. Coelho](#_heading=h.tfoo0tlldg9q) 30

[5.8. Crocodilo](#_heading=h.ks82qm8cdf7y) 31

[5.9. Dragão](#_heading=h.9f9j69858zkq) 33

[5.10. Galo](#_heading=h.a6fjv2cncfj) 36

[5.11. Gorila](#_heading=h.hikcpu39mpkg) 37

[5.12. Leão](#_heading=h.p1dc51buet) 39

[5.13. Macaco](#_heading=h.lcglof53pf95) 41

[5.14. Panda](#_heading=h.o6e0mv6e9z53) 42

[5.15. Porco](#_heading=h.jfmyph1ingav) 44

[5.16. Rato](#_heading=h.xmv7mpvbv1i) 45

[5.17. Tigre](#_heading=h.p7tphpvfzscp) 47

[5.18. UrsoPardo](#_heading=h.sutye0jlcjcy) 49

[5.19. UrsoPolar](#_heading=h.jgkgmbudxehl) 50

[5.20. Vaca](#_heading=h.ox5qzs4ktroq) 52

[5.21. AnimalIdAlreadyUsedF](#_heading=h.xafylztp46b5) 53

[5.22. AnimalNotFound](#_heading=h.grnpp36q0fup) 54

[5.23. InsAlreadyExistsF](#_heading=h.ioy78hb0v0mz) 54

[5.24. InsNotFound](#_heading=h.zgqjo2vwhmrm) 54

[5.25. Bovino](#_heading=h.c9iqh9suoyq4) 55

[5.26. Canis](#_heading=h.ghe5h6mdxss1) 56

[5.27. Capra](#_heading=h.b9aameja44rp) 56

[5.28. Crocodyloidea](#_heading=h.50pxi61fffzc) 57

[5.29. Dragon](#_heading=h.k0ne3wp2rt78) 57

[5.30. Equus](#_heading=h.8oh8fxx1glld) 58

[5.31. Gallus](#_heading=h.d9uyjhn5ve7y) 58

[5.32. Ophidia](#_heading=h.7ok79hng5es8) 59

[5.33. Oryctolagus](#_heading=h.hp45e3ro79ba) 60

[5.34. Panthera](#_heading=h.yloev7cxgrpm) 60

[5.35. Primata](#_heading=h.xbezos7hxtdp) 61

[5.36. Roedor](#_heading=h.581zpe76qt0g) 62

[5.37. Suíno](#_heading=h.nzqsjnqqj482) 62

[5.38. Urso](#_heading=h.qwu52vq5qqh9) 63

[5.39. Albinismo](#_heading=h.2g15z53bo96p) 64

[5.40. Dragonismo](#_heading=h.porv0019cv7u) 64

[5.41. Heterocromia](#_heading=h.db33qzzd3s7y) 64

[5.42. Adquirido](#_heading=h.untg7gl7g2ly) 65

[5.43. Colocação](#_heading=h.6sw5vwe0kjpg) 65

[5.44. Construção](#_heading=h.yhjjqn89ltyv) 66

[5.45. Fugiu](#_heading=h.38iw44fujiqg) 67

[5.46. Nascimento](#_heading=h.cv4vvbo10830) 68

[5.47. Ocorrência](#_heading=h.144t6dd5walz) 69

[5.48. Óbito](#_heading=h.skxpqq3ep428) 69

[5.49. Instalação](#_heading=h.lghk0sm8escu) 70

[5.50. Jumanji](#_heading=h.ha8dr49l0mb9) 74

[5.51. Zoo](#_heading=h.y8hzcqsm5010) 115

[**6. Anexo do Diagrama UML**](#_heading=h.7qnzdkw7hsmb) **168**

# Introdução

Este relatório tem por objetivo demonstrar a capacidade de aplicação dos conceitos adquiridos ao longo da unidade curricular de Programação Orientada por Objetos, desenvolvendo para isso uma aplicação que permita gerir um Zoo, utilizando a linguagem de programação *Java* e a plataforma “*NetBeans*”.

No final do relatório encontra-se incluído, em anexo, todo o código de todas as classes e o *diagrama UML* que descreve o modelo do projeto realizado.

# Ideia principal do projeto

Este projeto tem como ideia principal o desenvolvimento de uma aplicação que permita gerir um Zoo, podendo o utilizador gerir os animais presentes nas suas instalações, tanto como o seu balanço contabilístico e verificar as ocorrências num histórico.

# Procedimento e implementação do código

## 3.1. Classes

### 3.1.1. Classes Animais

Estas classes estão todas dentro da *package* Animais.

#### 3.1.1.1. Animal

Esta classe tem todos os métodos e variáveis essenciais que um animal do zoo tem, por isso é uma classe abstrata.

Nas suas variáveis temos, a “id” que é um inteiro que representa o id do animal no zoo, a “nome” que é uma String que representa o nome artístico do animal, a “idade” que é um inteiro que representa a idade do animal, a “bonus” que é um double que representa o bónus de atratividade que o animal tem (este depende das mutações e pode ser negativo já que o animal o perde ao longo dos anos), a “HETEROCROMIA” que é um booleano que é constante (logo é “final”) e representa a mutação de heterocromia no animal (será “true” se o animal a conter e “false” se não), a “ALBINISMO” que também é um booleano constante e como previsto representa a mutação de albinismo no animal (funciona igual a “HETEROCROMIA”) , e por fim “animaisZoo” que é um inteiro que é a única de classe (logo estática) e representa o número de animais já adicionados no zoo (DESDE O INÍCIO) e serve como uma forma de atribuir um id a um animal que temos a certeza que não existe no zoo (e neste caso nunca existiu). Estas variáveis são todas privadas de modo a ter encapsulamento.

Temos dois construtores nesta classe (ou seja um que é um *overload* do outro).

Um que recebe “nome” (uma String que será atribuída ao “nome” do animal) e “atratividadeBase” (um inteiro que representa uma atratividade de um animal que servirá adicionar ao bónus se tiver mutações) . Outro que recebe “nome” (já descrito em cima), “atratividadeBase” um inteiro que representa a atratividade base do animal e servirá adicionar ao bónus se tiver mutações), “genoma” (uma String que representa a característica genética do animal), “esperancaVida” (um inteiro que representa a esperança de vida do animal e servirá para calcular a idade do animal se adquirido) e “apetiteReprodutivo” (um inteiro que representa o apetite reprodutivo do animal).

Em ambos, o valor de “nome” é guardado na variável “nome” do animal, na variável “id” é guardado o valor de “animaisZoo” mais 1 (sendo que o valor dessas variaveis será realmente mudado no construtor das subclasses), damos o valor 0 a “bonus” e usamos o método “**incrementaBonus**” para adicionar a “bonus” a atratividade base (“atratividadeBase”) se tiver mutações (depende do número destas). Antes de usar esse método, na parte de calcular as mutações e idade que os dois diferem, para a idade no primeiro construtor é simplesmente dado o valor 0 a “idade”, mas no segundo usando o valor de “esperancaVida” e um valor random entre -3 e 3 calculamos a idade do animal (dividimos a esperança de vida do animal por 3 e adicionamos um inteiro que varia entre -3 e 3, e o valor é atribuído a “idade” do animal, assim temos variedade nas idades dos animais adquiridos) e verificamos se essa fica negativa e se sim é igualada a 0, para a parte das mutações são usados os métodos “**temHeterocromia**” e “**temAlbinismo**” para calcular se o animal tem essas mutações e para dar valor as constantes “HETEROCROMIA” e “ALBINISMO”, e no primeiro construtor esses métodos recebem só variáveis presentes na classe (“id”, “nome”, “bonus” e “idade”), mas no segundo eles recebem também as variáveis específicas desse animal vindas como argumento nesse (“atratividadeBase”, “genoma”, “esperancaVida” e “apetiteReprodutivo”).

O método “**toString**” imprime todas as variáveis desta classe correspondentes ao animal (“id”, “nome”, “bonus”, “idade”, “HETEROCROMIA” e “ALBINISMO”).

Os métodos “**temHeterocromia**” e “**temAlbinismo**” são ambos herdados das interfaces **Heterocromia** e **Albinismo** (logo são feitos dois *overrides*) , nestes são usadas os métodos “**Math.*abs***” e “**Objects.*hash***” em conjunto, pois com o “**Objects.*hash***” conseguimos um número inteiro dependendo dos seus argumentos (que vão ser os que referimos em cima passados como argumentos nos métodos “**temHeterocromia**” e “**temAlbinismo**”) e fazendo o resto da divisão inteira por 100 e aí temos meio que uma percentagem, mas como esse número pode ser negativo usamos o “**Math.*abs***” para ter a certeza que é positivo. Com essa percentagem vemos se ela é menor que um número fixo (que será a probabilidade de ter aquela mutação) e retornamos essa comparação.

O método “**incrementaAnimais**” aumenta o variável de classe “animaisZoo” por um (logo este método também é estático).

O método “**incrementaBonus**” recebe um argumento double “atratividadeBase” (que neste caso será o valor da atratividade base do animal em questão) e adiciona esse valor ao bónus do animal (“bonus”) se a variável “HETEROCROMIA” e/ou “ALBINISMO” for “true” (logo se o animal tiver heterocromia e/ou albinismo).

O método “**incrementaIdade**” aumenta o valor da variável “idade” por um (logo aumenta a idade do animal por um ano).

Os métodos “**comerAnual**” e “**cuidarAnual**” recebem o zoo em questão como argumento e são ambos abstratos e implementados em cada subclasse de **Animal** (isto é em cada animal específico).

#### 3.1.1.2. Subclasses de Animal

As subclasses de **Animal** são todas construídas de forma semelhante e só mudam os valores das suas variáveis (pois são todas tipos de animais diferentes).

Todas as suas variáveis são de classe (pois são iguais para todos os animais de cada tipo) e temos, “atratividadeBase” que é uma variável inteira, representa a atratividade base do animal (isto é a sua atratividade para os visitantes do zoo) e em animais fora do calendário chinês é uma constante (“ATRATIVIDADEBASE”) pois nunca vai mudar, “GENOMA” que é uma String constante e representa a característica genética do animal, “ESPERANCAVIDA” que é uma constante inteira e representa a esperança média de vida do animal e por fim “APETITEREPRODUTIVO” que é uma constante inteira que representa o apetite reprodutivo do animal (isto é, a percentagem de ter filhos ao longo de um ano). Estas variáveis são todas privadas de modo a ter encapsulamento.

O construtor destas classes recebe dois argumentos, “nome” que é uma String e será usada como nome artístico do animal (vai ser atribuída à variável da superclasse “nome”) e “adiciona” que é um booleano que representa a intenção de adicionar o animal ao zoo (se for “true” o animal é adicionado ao zoo e se for “false” não é). Neste construtor é chamado o “super” da superclasse e neste é passado o argumento “nome” e todas as variáveis deste animal (“atratividadeBase”, “GENOMA”, “ESPERANCAVIDA”, “APETITEREPRODUTIVO”), depois desse é verificado o valor do argumento “adiciona”, se for “true” é aumentado o valor da variável de classe da superclasse “animaisZoo” por um (já que tem mais um animal no zoo) e depois é guardado o valor deste na variável da superclasse “id” (desta forma teremos sempre um id diferente e nunca usado para cada animal).

O método “**toString**” imprime o nome do tipo de animal (por exemplo “Boi”), junta com o “toString” da superclasse **Animal** (“+super.toString()+”) e incrementa as suas variáveis a esse (“atratividadeBase”, “GENOMA”, “ESPERANCAVIDA”, “APETITEREPRODUTIVO”).

Os métodos “**comerAnual**” e “**cuidarAnual**” são ambos herdados da superclasse **Animal** e implementados em cada animal (pois cada animal come e necessita de cuidados diferentes) , nestes é recebido o zoo como argumento e ao saldo deste zoo é retirado o valor da comida e dos cuidados anuais desse tipo de animal.

Nas classes dos animais que estão no calendário chinês (**Boi**, **Cabra**, **Cachorro**, **Cavalo**, **Cobra**, **Coelho**, **Dragão**, **Galo**, **Macaco**, **Porco**, **Rato**, **Tigre**) existe o método “**incrementaAtratividadeBase**” que duplica a atratividade base (“atratividadeBase”) do animal.

Nestas classes todas a única que se difere é **Dragão**, pois este tem mais uma mutação possível (dragonismo) que é específica para os dragões, logo tem uma variável “DRAGONISMO” que é um booleano constante e representa a mutação de dragonismo no dragão, esta é inicializada no construtor do **Dragão** usando o método herdado (logo é feito um *Overrride*) da interface **Dragonismo** (que só é implementada em **Dragão**) “**temDragonismo**” onde o seu funcionamento é semelhante aos métodos das outras mutações (explicadas em cima na classe **Animal**), continuando no construtor , é verificado se o dragão tem dragonismo e aí é adicionada a atratividade base deste (“atratividadeBase”) ao seu bónus (que está na superclasse, (“bonus”)). Essa variável também será imprimida somente no “**toString**” de **Dragão**.

### 3.1.2. Classes Genomas

Estas classes estão todas dentro da *package* Genomas.

Para a criação dos genomas de cada animal, utilizou-se interfaces, construídas de forma semelhante, só mudando o valor da sua variável (pois são animais diferentes), “preco” que é uma static double, que representa o custo do animal com esse genoma. Esta variável é privada de modo a ter encapsulamento.

Como são interfaces não tem construtor, e só tem um único método static Animal, responsável por criar de forma aleatória animais com o seu respectivo genoma, por exemplo o método **“criaBovino”**  cria um animal com genoma **Bovino**, dependendo do número random, que pode ser **Boi** no caso do número ser 0 ou **Vaca** no caso do número ser diferente de 0, outro exemplo pode ser o método **“criaUrso”** que cria animais com genoma **Urso**, de igual forma utilizando números random e dependendo se o número é 0 cria um panda, se é 1 cria um urso pardo e se é 2 cria um urso polar, outro exemplo é o método **“criaDragon”** que neste caso só cria um **Dragão**, pois é o único animal com esse genoma.

### 3.1.3. Classes Mutações

Estas classes estão todas dentro da *package* Mutações.

Para a criação das mutações optou-se por utilizar interfaces para cada uma delas, com um único método abstract boolean **“temAlbinismo”,** para animais com albinismo, **“temHeterocromia”**, para animais com heterocromia e **“temDragonismo”**, para animais com dragonismo (que no caso é só **Dragão**), este método recebe como argumento um conjunto de objetos.

### 3.1.4. Instalação

Esta classe está dentro da *package* principal (com.mycompany.jumanji).

Esta é a classe para as instalações do zoo, que contém todas as variáveis e métodos relacionados a tal.

Nas suas variáveis temos, “numeroInstalacao” que é um inteiro e representa o id da instalação, “lotacao” que também é um inteiro e representa o tamanho da instalação (quantos animais consegue ter instalados), “animaisInstalados” que também é um inteiro e representa os animais instalados na instalação (de modo a sabermos se a instalação está cheia), “precoInstalacao” que também é um inteiro e representa o preço de construção da instalação, “animais” que é uma ArrayList e é onde serão guardados os animais da instalação e por fim “instalacoes” que é um inteiro que é o único estático (logo de classe) e representa o número de instalações já adicionados no zoo e serve como uma forma de atribuir um número a uma instalação que temos a certeza que não existe no zoo. Estas variáveis são todas privadas de modo a ter encapsulamento.

O construtor desta classe recebe um booleano “constroi” que representa a intenção de adicionar a instalação ao zoo (se for “true” a instalação é adicionada ao zoo e se for “false” não é). Neste construtor é gerado um valor para “lotacao” utilizando o método “**geraLotacao**”, depois usando esse valor inicializamos a ArrayList “animais” (usando “lotacao” como sua capacidade inicial), damos valor 0 a “animaisInstalados” e geramos um preço para a construção da instalação (“preco”) usando o método “**geraPreco**”, dizemos também que o valor do número da instalação (“numeroInstalacao”) é o número de instalações no zoo (“instalacoes”) mais 1 (sendo que este será mesmo atribuído à instalação no “if” no final do construtor) , seguidamente verificamos o valor de “constroi” e se for “true” é aumentado a variável de classe “instalacoes” por 1 (já é construída uma nova instalação no zoo) e é dado esse valor a “numeroInstalacao” (atribuindo o id á instalação).

O método “**toString**” imprime todas as variáveis desta classe correspondentes a uma instalação (“numeroInstalacao”, “lotacao”, “animaisInstalados”, “preco” e “animais”).

O método “**incrementaInstalacoes**” aumenta o variável de classe “instalacoes” por um (logo este método também é estático, e aumenta o número de instalações inseridos no zoo desde o início).

Os métodos “**geraLotacao**” e “**geraPreco**” são usados para gerar uma lotação e um preço de construção para a instalação respetivamente, os dois têm um funcionamento parecido, o “**geraLotacao**” com ajuda de um random gera um inteiro de 1 a 10 e retorna esse inteiro (atribuindo esse valor à variável “lotacao” no construtor) e o “**geraPreco**” também com a ajuda de um random gera um double entre 250 e 749, e retorna esse valor multiplicado pela lotação da instalação (“lotacao”) (assim temos uma pequena variação de preços para as instalações mas que depende também das suas lotações).

O método “**adicionaAnimal**” serve para adicionar um animal à instalação, este recebe dois argumentos, “zoo” que é o zoo em que a instalação se encontra e “animal” que é o animal a adicionar à instalação. Neste método verificamos se “animaisInstalados” (animais na instalação) é menor que “lotacao” (capacidade da instalação) , se for simplesmente adicionamos o animal a ArrayList da instalação e adicionamos um valor à variável “animaisInstalados”, se não for então quer dizer que a instalação está cheia, imprimimos todos os animais na instalação e pedimos ao utilizador (utilizando um scanner) para escolher um para substituir pelo que vai inserir, depois pegamos no animal que o utilizador escolheu, o guardamos na ArrayList dos animais fora das instalações, o retiramos da ArrayList da instalação e adicionamos o novo nessa ArrayList.

O método “**manutencaoInstalacaoAno**” recebe o zoo como argumento e retira ao saldo deste o valor da manutenção da instalação durante um ano, que é calculado multiplicando a sua lotação (“lotacao”) por 200.

### 3.1.5. Zoo

Esta classe está dentro da *package* principal (com.mycompany.jumanji).

Esta classe é a classe mais extensa do nosso projeto e é a mais importante pois contém maior parte das funções principais e relaciona todas as classes, é a classe do zoo e contém todas as suas variáveis e métodos.

Nas suas variáveis temos, “nomeZoo” que é uma String e representa o nome do zoo em funcionamento, “anoZoo” que é um inteiro e representa o ano atual, “zoo” que é um HashMap de instalações onde a sua key é o número da instalação e o seu value é a instalação em si, “animaisFI” que é uma ArrayList de **Animal** onde são armazenados os animais fora das instalações, “obituario” que é uma ArrayList de **Animal** onde são armazenados os animais mortos, “historico” que é uma ArrayList de **Ocorrência** onde são armazenadas as ocorrências desde o começo do zoo e por fim “saldo” que é a única de classe, é inteiro e representa o saldo do zoo. Estas variáveis são todas privadas de modo a ter encapsulamento.

O construtor desta classe recebe um booleano “receba” que representa a intenção de pedir ou não por nome e ano ao utilizador. Neste é inicializadas todas as ArrayLists (“animaisFI”, “historico” e “obituario”), o HashMap (“zoo”) e também o saldo do zoo ("saldo"), sendo que este começa a 0, depois verificamos o valor de “receba” e se for “true” executamos os métodos “**recebeNome**” e “**recebeAno**” (para pedir um nome e um ano ao utilizador) e atribuimos seus valores a “nomeZoo” e a “anoZoo” respetivamente, por fim inicializamos o menu do zoo (“**menu**”).

Os métodos “**recebeNome**” e “**recebeAno**” são usados para pedir um nome e um ano ao utilizador para o zoo (respetivamente), e têm um funcionamento parecido, o “**recebeNome**” utiliza uma classe **Scanner** para obter a String que o utilizador escreve na consola, verifica se essa é mesmo uma String e se é composta só por letras e depois dá o valor dessa String a variável “nomeZoo”, e o “**recebeAno**” também usando uma classe **Scanner** obtém o inteiro que o utilizador escreve, verifica também se é inteiro e dá esse valor à variável “anoZoo”.

Os métodos “**adicionarFI**” e “**removerFI**” são usados para manipular a ArrayList “animaisFI” e como o nome indica, o primeiro serve para adicionar um animal a essa ArrayList (pois este recebe o animal como argumento) e o segundo remove um animal da ArrayList (pois recebe um inteiro como argumento, e este é a posição do animal a remover).

O método “**adicionarObitos**” recebe um animal como argumento e adiciona esse animal a ArrayList “obituario” do zoo.

O método “**adicionarInstalacao**” recebe uma instalação como argumento e adiciona essa instalação ao HashMap “zoo” do zoo.

O método “**adicionarAI**” recebe um inteiro e um animal como argumento, sendo este inteiro o número de uma instalação, ele obtém a instalação utilizando o seu número e o HashMap “zoo” e insere na ArrayList desta instalação o animal.

O método “**adicionarOcorrencia**” recebe uma ocorrência como argumento e adiciona essa ocorrência a ArrayList “historico” do zoo.

O método “**adicionarSaldo**” recebe um double como argumento e adiciona ao saldo do zoo (“saldo”) esse double (e como “saldo” é uma variável de classe o método é estático).

O método “**aumentarAno**” quando chamado aumenta por um valor a variável “anoZoo”.

O método “**menu**” é o método responsável do menu do zoo, que tem todas as opções de execução do programa que o utilizador pode invocar, é o método que invoca os métodos principais do programa necessários e pedidos para o projeto (trata-se do switch enorme com todas as opções pedidas e dentro deste maior parte dos métodos do programa).

O método “**voltarMenu**” serve simplesmente como um método auxiliar para voltar ao menu original do zoo, não recebe argumentos e com a ajuda de uma classe **Scanner** pede ao utilizador para escrever o número 0 para voltar para o menu, verifica se esse é um inteiro e verifica se é 0 (se não for executa o método denovo).

Os métodos “**constroiAnimalID**” e “**constroiAnimal**” têm o mesmo propósito, que é de completar um animal, ambos recebem como argumento o tal animal (“animal”) e a String da ocorrência relacionada a esse (“ocorrencia”), depois pedem ao utilizador um nome para o animal e verificam se esse nome é uma String e se é uma String de letras, e é nesta parte final que os dois se diferem, o “**constroiAnimalID**” serve para os animais que ainda não foram adicionados ao zoo (isto é não têm id) por isso ele aumenta o número de animais no zoo (“animaisZoo”) utilizando o método “**incrementaAnimais**” (método de classe de **Animal**), usando esse valor dá um id ao animal (“id”), usando o nome dado pelo utilizador dá um nome artístico ao animal (“nome”), adiciona esse animal a ArrayList dos animais fora das instalações (“animaisFI”) utilizando o “**adicionarFI**” e cria uma ocorrência usando o mé todo “**criaOcorrencia**” e a String recebida como argumento (“ocorrencia”), e o “**constroiAnimal**” é igual só que menos completo pois este serve para os animais que já estão no zoo, por isso não aumenta o número de animais no zoo nem dá um id ao animal.

O método “**constroiInstalacao**” serve para construir uma instalação no zoo, neste caso completar um instalação, recebe como argumento a tal instalação, depois aumenta o número de instalações no zoo usando o método “**incrementaInstalacoes**” (método de classe de **Instalação**), usando esse valor atribui um valor ao “numero” da instalação, adiciona essa instalação ao HashMap “zoo” usando o método “**adicionarInstalacao**”, retira ao saldo do zoo (“saldo”) o preço de construção da instalação (“preco”) usando o método “**adicionarSaldo**” e por fim cria a ocorrência de construção de uma instalação usando o método “**criaOcorrencia**”.

O método “**instalarAnimal**” é um dos métodos importantes do zoo, é com este método que vamos conseguir colocar um animal numa instalação, este não tem argumentos, e começa por verificar se existem animais fora das instalações e se existem instalações no zoo (pois sem um ou outro o processo de instalar um animal não é possível), depois de verificar os dois imprime todas as instalações do zoo (todos os elementos do HashMap “zoo”) e utilizando uma classe **Scanner** pede ao utilizador para escolher uma (verifica se o que o utilizador escreveu é um inteiro), deste ponta para a frente na função temos um bloco try-catch (devido a duas exceções neste código) dentro deste começamos por verificar se existe uma instalação com o número dado pelo utilizador percorrendo o HashMap e com a ajuda de uma variável local “existe” (se não houver é lançada uma exceção **InsNotFound**), e se houver continuamos e imprime todos os animais fora das instalações (percorrendo a ArrayList “animaisFI”) e utilizando outra vez o scanner pedimos ao utilizador para escolher uma animal para instalar (verificando se o que o utilizador escreveu é um inteiro), depois verificamos se existe (na ArrayList dos animais fora das instalações “animaisFI”) o animal escolhido (se não existir é lançada uma exceção **AnimalNotFound**), depois de todas as verificações o animal é adicionado a ArrayList da isntalação usando o método “**adicionarAI**”, imprime as informações do animal e da instalação dizendo que este foi adicionado, cria uma ocorrência relacionada com o animal e a instalacao (que neste caso é a colocação do animal) usando o método “**criarOcorrencia**” e por fim retira o animal da ArrayList dos animais fora das instalações (“animaisFI”) usando o método “**removerFI**”.

O método “**comprarAnimal**” é usado para retirar ao saldo do zoo o valor para adquirir determinado animal, este recebe o tal animal como argumento e verificar de que tipo é (utilizando a sua classe) e depois de descobrir qual é, utilizando o método “**adicionarSaldo**”, retira ao saldo metade da atratividade base desse tipo de animal (obtendo essa com um método estático get, pois essa variável também é estática).

O método “**comprarGenoma**” é usado para retirar ao saldo do zoo o valor para adquirir determinado animal de um certo genoma, este recebe uma String “genoma” que o respetivo nome do genoma, depois verifica qual é este genoma e retira ao saldo do zoo (“saldo”) o preço desse genoma (“preco”) usando o método “**adicionarSaldo**” em conjunto com o método de classe “**getPreco**” da classe do genoma.

O método “**animalRandomizado**” é usado para criar um animal randomizado do zoo, utilizando um random que vai de 0 a 18 (sendo que o zoo tem 19 animais) escolhe retornar um animal qualquer do zoo sempre que este método é chamado.

O método “**animalGenoma**” é usado para criar um animal de um certo genoma, este recebe uma String “genoma” que o respetivo nome do genoma, depois verifica qual é este genoma e invoca o método estático “**cria…**” da classe do genoma para criar um animal randomizado com esse genoma, se não existir este genoma no zoo é retornado “null”.

O método “**anoChines**” serve para aumentar a atratividade base dos animais do zoo dependendo do calendário chines, guarda numa variável inteira o resto da divisão por 12 do ano atual, dependendo desse valor, utilizando um switch case, seleciona-se o animal desse ano e incrementa-se a sua atratividade base de forma permanente, invocando ao método “**incrementaAtratividadeBase**”implementado em cada classe dos animais do zoo, de seguida imprime o ano atual e o animal, que aumentou a sua atratividade base.

Para listar os animais, que se encontram em todo o zoo criou-se o método, “**listarAnimais”,** este método percorre o array dos animais que se encontram fora das instalações, se o array se encontra vazio imprime "Não existem animais no zoo", no caso de existir animais no zoo, imprime cada animal, também percorre o hashmap das instalações do zoo, imprime o número da instalação e no caso da instalação se encontrar vazia imprime "Não existem animais dentro desta instalação", caso contrário imprime cada animal que se encontra nessa instalação.

Para listar os animais com dada mutação, criou-se o método **“listarAnimaisM()”,** o qual verifica se existem animais no zoo, utilizando o “**isEmpty**”, se o zoo não tem animais imprime "Não existem animais no zoo", se existem animais, pede ao utilizador para inserir o tipo de mutação que desejar e é guardado utilizando o Scanner numa variável String mutacao, primeiro procura animais com a mutação desejada fora das instalações, se não tem animais fora das instalações imprime "Não existem animais fora das instalações", no caso de existir utilizando um boolean existe, booleano que serve para saber se existe pelo menos um animal com a mutação desejada fora das instalações, inicializado em false, depois percorre o array de animais fora das instalações, se a mutação é Heteronomia imprime os animais com a mutação e existe passa para true, se a mutação é Albinismo imprime os animais com a mutação e existe passa para true, se a mutação é Dragonismo, como só acontece na classe **Dragão,** é utilizado um downCast, pois precisamos de métodos da subclasse **Dragão ,** existe passa para true e imprime o dragão com Dragonismo. se existir for false quer dizer que não existem animais com a mutação fora das instalações, logo imprime "Não existem animais com essa mutação fora das instalações", de seguida percorre o hashmap das instalações e realiza o mesmo procedimento.

Para listar os animais com dada caraterística genética criou-se o método **“listarAnimaisG”,** o qual verifica se existem animais no zoo, utilizando o “**isEmpty**”, se o zoo não tem animais imprime "Não existem animais no zoo", se existem animais, pede ao utilizador para inserir o tipo de genoma que desejar e é guardado utilizando o Scanner numa variável String genoma, primeiro procura animais com o genoma desejado fora das instalações, se não tem animais fora das instalações imprime "Não existem animais fora das instalações", no caso de existir utilizando um boolean existe, booleano que serve para saber se existe pelo menos um animal do genoma fora das instalações, inicializado em false, depois percorre o array de animais fora das instalações, e utiliza um switch (genoma), para verificar que genoma o utilizador escreveu, depois verifica se o animal tem o genoma que o utilizador deseja, se tem imprime o animal e existir passa para true, se nenhum animal tem o genoma desejado existir fica em false e imprime "Não existe nenhum animal com esse genoma fora das instalações", de seguida percorre o hashmap das instalações e realiza o mesmo procedimento.

O método “**listarInstalacoes**” é usado para imprimir todas as instalações do zoo, primeiro verifica se existem instalações no zoo, depois se tiver percorre o HashMap “zoo” (onde se situam as instalações do zoo) e imprime todas.

O método “**criarOcorrencia**” é usado para simplificar a criação das ocorrências, recebe uma String (“ocorrencia”) que é o nome da ocorrência a criar, também recebe um animal (“animal”) e uma instalação (“instalacao”) e vai ou não usar estes dependendo da ocorrência (por exemplo a ocorrência **Nascimento** só precisa de animal), esta ocorrência será criada usando a String recebida e se esta não existir imprime que ocorreu um erro a criar a ocorrência.

O método “**periodoContabilistico”,**  é um método que simula o período contabilístico do zoo no qual se contabilizam os custos e os proveitos, se atualiza o saldo do zoo e registam-se os nacimentos e os óbitos dos animais, para isto este método, invoca aos métodos: **“animaisSaldo()”, “animaisISaldo()”, “instalacoesSaldo()”, “aumentarIdade()”, “aumentarAno()”, “nascerAnimal()”, “morrerAnimal()” e “ prejuizo()”.**

O método “**nascerAnimal**” serve para que possam nascer animais no zoo, percorre o hashmap com os arrays dos animais que estão nas instalações do zoo, utilizando um ciclo foreach, de seguida utilizando outro ciclo foreach, utilizou-se uma variável Animal a que guarda classe do animal da posição em que se encontra, depois compara essa classe com todas as classes de animais, quando encontrar a igualdade, gera um inteiro número random entre 0 e o 99 e guarda numa variável inteira probNacer , a seguir compara probNacer com o apetite reprodutivo do animal, APETITEREPRODUTIVO variável static que está em todas as classes de animais, se probNacer é menor ou igual a APETITEREPRODUTIVO, cria um novo animal dessa classe pois nasce um animal, de seguida invoca ao método “**constroiAnimal**”, o qual recebe como argumento um animal e o nome da ocorrência, este método é implementado para atribuir um nome ao animal criado, adicionar o animal ao array dos animais criados e criar uma ocorrência utilizando o método “**criaOcorrencia**”.

Para as mortes dos animais foram criados os métodos “**morrerAnimal**” e “**aumentarIdade**”.

O método “**aumentarIdade**”, percorre o array dos animais criados e utilizando o método **“incrementaIdade”** (definido em cada classe dos animais) aumenta a idade de todos os animais do array, além disso chama ao método “**incrementaBonus**”, utilizado para diminuir a atratividade base de cada animal à medida que vai ficando velho, por último volta a fazer o mesmo procedimento, mas para o hashmap dos animais das instalações.

O método **“morrerAnimal”,** percorre o array dos animais criados utilizando um bucle for, depois compara cada classe de cada animal do array, posição por posição, com todas as classes de animais, quando se verificar a igualdade, compara a ESPERANCAVIDA (variável static que se encontra definido em cada classe dos animais, e guarda o valor dos anos que vive normalmente um animal ) com a idade atual do animal, se a idade do animal for maior o animal morre, logo é indicado o animal que morreu e é guardado no array Óbitos que guarda todos os animais que morreram e por último retira o animal do array dos animais criados, por último volta a fazer o mesmo procedimento, mas para o hashmap dos animais das instalações.

Para saber os ganhos e as perdas do zoo, criou-se os métodos **“animaisSaldo()”, ”animaisISaldo()”, ”instalacoesSaldo”**.

O método **“animaisSaldo”,** é um método criado para calcular os gastos dos animais que não estão em exibição nas instalações, pelo que não produzem ganhos ao zoo,este métodopercorre o array com os animais criados no zoo utilizando um bucle foreach , depois compara cada classe de cada animal do array, posição por posição, com todas as classes de animais, quando se verificar a igualdade invoca aos métodos **“comerAnual”,**  que subtrai ao saldo o preço da comida do animal e o **“cuidarAnimal”**, que subtrai ao saldo os custos do cuidado do animal.

O método **”animaisISaldo”**, é um método que calcula os gastos dos animais, que estão nas instalações, e os ganhos que eles produzem ao zoo, este método percorre o array com os animais nas instalações utilizando um bucle foreach, depois compara cada classe de cada animal do array, posição por posição, com todas as classes de animais, quando se verificar a igualdade invoca aos métodos **“comerAnual”,**  que subtrai ao saldo o preço da comida do animal e o **“cuidarAnimal”,**  que subtrai ao saldo os custos do cuidado do animal, e por último é adicionado ao saldo a atratividade base e o bónus.

O método **”instalacoesSaldo”**, é um método utilizado para retirar ao saldo as despesas das instalações do zoo, este método percorre o hashmap dos instalações do zoo e retira ao saldo o valor das manutenções da instalação invocando ao método **“manutencaoInstalacaoAno”.**

Para saber se o zoo tem prejuízos, criou-se o um método **“prejuizo”,** que verifica se o saldo é negativo ou positivo, se o saldo é negativo imprime "Saldo negativo, logo pode haver prejuízos" e invoca ao método **“jumanji()”**, se o saldo é positivo imprime "Saldo Positivo, logo não há prejuízos".

O método “**jumangi**” serve para criar uma confusão no zoo em que alguns animais podem fugir deste, este método percorre o array dos animais fora das instalações (“animaisFI”) e o HashMap das instalações (”zoo”) e as suas ArrayLists de animais, depois utilizando um random de 0 a 99 verifica se esse deu maior que 30, e se sim o animal fugiu, depois verifica também se esse é maior que 75 e se for esse animal fugiu do zoo permanentemente e é criada a ocorrência de tal mas se não esse animal fugiu mas foi capturado e não é criada a ocorrência. Ao fugir do zoo o animal é retirado da ArrayList de onde estava (a dos animais fora das instalações ou a das instalações) mas se for capturado volta ao zoo e é armazenado na ArrayList dos animais fora das instalações.

O método “**retratoF**” serve simplesmente para imprimir todo o zoo, este imprime todos os animais no zoo (usando “**listarAnimais**”), todas as instalações (usando “**listarInstalacoes**”), o obituário (“obituario”) e o histórico (“historico”).

### 3.1.6. Classes Ocorrências

Estas classes estão todas dentro da *package* Ocorrências.

Estas são as classes das ocorrências no zoo, e servem para guardar e criar ocorrências de modo a ter histórico do zoo.

Existem 6 ocorrências possíveis no zoo, **Adquirido** (quando é adquirido um animal), **Nascimento** (quando nasce um animal), **Óbito** (quando morre um animal), **Fugiu** (quando foge um animal), **Colocação** (quando é instalado um animal), **Construção** (quando é construída uma instalação) e são todas subclasses de **Ocorrência**.

A superclasse **Ocorrência** tem uma variável “zoo” que se refere ao zoo onde a ocorrência aconteceu. O seu construtor recebe um argumento “zoo”, este que vem com o valor do zoo da ocorrência e será guardado na sua variável “zoo”. Esta classe só tem um método relevante (sem ser getter nem setter) que é o “**toString**” que imprime a informação do zoo da ocorrência.

As ocorrências que são estritamente do animal (“**Adquirido**”, “**Nascimento**”, “**Óbito**” e “**Fugiu**”) só têm uma variável “animal” que é um **Animal** e representa o animal relacionado àquela determinada ocorrência, a que é estritamente das instalações (“**Construção**”) também só tem uma variável “instalacao” que é uma **Instalação** e representa a instalação relacionada à ocorrência e por fim a que implica os dois (“**Colocação**”) tem duas variáveis, uma para o animal e outra para a instalação (“animal” e “instalacao”). Nos seus construtores é sempre recebido como argumento o zoo (pois é necessário para o super) e as que têm “animal” recebem esse animal como argumento as que têm “instalacao” recebem essa instalação como argumento (pois dão esses valores as suas variáveis). Como a sua superclasse, só têm um método relevante que é o “**toString**” e este (dependendo da ocorrência) imprime os atributos das suas variáveis (valores do animal e/ou da instalação). As variáveis são todas privadas de modo a ter encapsulamento.

### 3.1.7. Classes Exceções

Estas classes estão todas dentro da *package* Exceções.

No nosso código existem quatro classes que estendem de **Exception** e essas são as nossas exceções especiais.

A primeira é a **AnimalIdAlreadyUsedF** que é lançada quando quando ao ler o ficheiro é verificado que o id do animal a ser lido é igual a um id de um animal já no zoo, ao ser lançada é mostrada uma mensagem específica da exceção.

E a segunda é a **InsAlreadyExistsF** que é lançada quando ao ler o ficheiro é verifica que o número da instalação a ser lida é igual a um de uma instalação já no zoo, ao ser lançada também mostra uma mensagem específica da exceção.

A terceira é a **AnimalNotFound** que é lançada quando é instanciado um animal que não existe (isto é um índice fora da ArrayList “animaisFI”) ao tentar colocá-lo numa instalação, ao ser lançada também mostra uma mensagem específica da exceção.

E a última é a **InstNotFound** que é lançada quando é instanciada uma instalação que não existe (isto é, tentar meter um animal numa instalação que não foi criada por exemplo), ao ser lançada também mostra uma mensagem específica da exceção.

### 3.1.8. Ficheiros (Jumanji/Main)

Esta classe está dentro da *package* principal (com.mycompany.jumanji).

A nossa classe principal é a Jumanji, neste temos o nosso “main” que é onde o código é executado, é a primeira classe a ser executada no programa.

O método “**main**” é começado pela escolha da opção de início (se o utilizador quer carregar dados ou quer criar um novo zoo), é utilizada uma classe **Scanner** para obter a resposta (que é verificada se é inteira), se o utilizador escolher criar um novo zoo, simplesmente é criada uma nova instância de Zoo com o argumento booleano a “true” (quer dizer que é para pedir o nome e o ano deste zoo), mas se o utilizador decidir carregar os dados, são executados os métodos responsáveis por isso (“**carregaZoo**”, “**carregaAnimaisFI**”, “**carregaInstalação**” e “**carregaAnimaisI**”) que serão falados em baixo e inicializado o menu desse zoo (“**menu**”). Também é neste método onde são apanhadas todas as exceções dos respectivos métodos de carregar e são imprimidas as mensagens de erro dos mesmos.

O método “**carregaZoo**” é o responsável por carregar as informações de nome do zoo e ano do zoo de um ficheiro Zoo.txt, para o programa, para isto o método recebe por referência o zoo e utiliza duas variáveis auxiliares String “nomeZoo” e int “anoZoo”, que são utilizadas para guardar as informações do ficheiro, para abrir e ler o ficheiro utilizou-se um **FileReader** e um **BufferredReader** ( para ler linha a linha do ficheiro), depois é guardada nas variáveis auxiliares e posteriormente guardadas no zoo.

O método “**carregaAnimaisFI**” é responsável por carregar os animais que estão fora das instalações, este carrega as informações principais de cada animal a adicionar ao zoo (“id”, “nome” e “idade”), recebe uma referência do zoo e utiliza um **FileReader** em conjunto com um **BufferredReader** para ler as linhas do ficheiro txt (“AnimaisZoo.txt”) uma a uma, guardando essas informações sequencialmente nas variáveis, neste método ao contrário do anterior verificamos a primeira linha para saber qual é o animal a criar (por exemplo “Leão:”) com um switch, depois verificamos a segunda linha (que será o id) para saber se já tem algum animal no zoo com esse id (e se tiver lançamos uma exceção **AnimalIdAlreadyUsedF**) e depois na parte de mudar o id do animal verificamos se o seu id é maior que o número de animais no zoo (“animaisZoo”), e se for o igualamos ao valor do id (de modo que quando criamos um novo animal no zoo não haja repetição de ids).

O método “**carregaInstalacoes**” é responsável por carregar as instalações do zoo, este carrega todas as informações principais de cada instalação (“numeroInstalacao” e “lotacao”), recebe uma referência do zoo e como os métodos anteriores utiliza um **FileReader** em conjunto com um **BufferredReader** para ler as linhas do ficheiro txt (“ZooInstalação.txt”) uma a uma sequencialmente, é verificada a primeira linha lida do ficheiro (“numeroInstalacao”) para verificar se já existe uma instalação com esse número (se existir é lançada uma exceção **InsAlreadyExistsF**), e como no método acima é verificado se o número de instalações (“instalacoes”) é menor que o id da instalação lida, e se for o número de instalações é igualado ao id (de modo que quando criamos uma nova instalação no zoo não haja repetição de números).

O método “**carregaAnimaisI**” é responsável por carregar os animais que estão nas instalações, este carrega as informações principais de cada animal a adicionar ao zoo (“id”, “nome” e “idade”), recebe uma referência do zoo e utiliza um **FileReader** em conjunto com um **BufferredReader** para ler as linhas do ficheiro txt (“AnimaisZooI.txt”) uma a uma, guardando essas informações sequencialmente nas variáveis, neste método ao contrário do anterior verificamos a primeira linha, para saber qual é o número da instalação do animal a criar, se a instalação não existe(lançamos a exceção InsNotFound), depois cria-se o animal (por exemplo “Leão:”) com um switch, verificamos a terceira linha (que será o id) para saber se já tem algum animal no zoo com esse id (e se tiver lançamos uma exceção **AnimalIdAlreadyUsedF**) e depois na parte de mudar o id do animal verificamos se o seu id é maior que o número de animais no zoo (“animaisZoo”), e se for o igualamos ao valor do id (de modo que quando criamos um novo animal no zoo não haja repetição de ids).

# Conclusão

Com este projeto conseguimos aprender a elaborar diagramas de modo a facilitar a interpretação de classes em Java.

Concluímos que ao utilizar classes conseguimos organizar e elaborar código a um nível profissional assim conseguindo simplificar o manuseamento e a localização de certos elementos.

Em geral a elaboração deste projeto foi razoavelmente acessível, e ajudou-nos a recordar e reforçar conhecimentos acerca de Programação Orientada a Objetos.

# Anexo do código

## 5.1. Animal

package Animais;

import Mutações.Albinismo;

import Mutações.Heterocromia;

import com.mycompany.jumanji.Zoo;

import java.util.Objects;

import java.util.Random;

/\*

Classe dos animais do zoo

\*/

public abstract class Animal implements Albinismo, Heterocromia{

private int id; //Id do animal

private String nome; //Nome do animal

private int idade; //Idade do animal

private double bonus; //Bonus de atratividade do animal (pode ser negativo porque o animal perde este ao longo dos anos)

private final boolean HETEROCROMIA; //Booleano que define se o animal tem heterocromia

private final boolean ALBINISMO; //Booleano que define se o animal tem albinismo

private static int animaisZoo; //Quantos animais foram inseridos no zoo desde o início

/\*Recebe 3 argumentos,

"nome" é o nome artístico do animal,

"atratividadeBase" é a atratividade base do animal em questão e

"esperancaVida" que é a esperança de vida também do animal em questão

\*/

public Animal(String nome, double atratividadeBase) {

this.nome = nome;

idade = 0;

bonus = 0;

id = animaisZoo+1; //Diz que o id do animal é os animas já inseridos no zoo mais 1 (O seu id e animaisZoo vão realmente ser mudados no construtor das subclasses)

HETEROCROMIA = temHeterocromia(nome,idade,bonus,id); //Calcula e verifica se o animal tem heterocromia utilizando as caraterísticas presentes

ALBINISMO = temAlbinismo(nome,idade,bonus,id, HETEROCROMIA); //Calcula e verifica se o animal é albino utilizando as caraterísticas presentes

incrementaBonus(atratividadeBase); //Incrementa ao bonus do animal a sua atratividade dependendo das mutações

}

/\*Recebe 5 argumentos,

"nome" é o nome artístico do animal,

"atratividadeBase" é a atratividade base do animal em questão,

"genoma" é a caraterística genética também do animal em questão,

"esperancaVida" é a esperança de vida também do animal em questão e

"apetiteReprodutivo" que é o apetite reprodutivo também do animal em questão

\*/

public Animal(String nome, double atratividadeBase, String genoma, int esperancaVida, int apetiteReprodutivo){

Random rand = new Random();

this.nome = nome;

idade = (esperancaVida/3)+(rand.nextInt(3+3)-3); //Calcula a idade do animal dividindo a sua esperança de vida por 3 e adicionando a esse número outro que varia entre -3 e 3

if(idade<0)idade=0; //Se o calculo da idade for menor que 0, a idade do animal fica a 0

bonus = 0;

id = animaisZoo+1; //Diz que o id do animal é os animas já inseridos no zoo mais 1 (O seu id e animaisZoo vão realmente ser mudados no construtor das subclasses)

HETEROCROMIA=temHeterocromia(id, nome,idade,bonus,atratividadeBase, genoma, esperancaVida, apetiteReprodutivo); //Calcula e verifica se o animal tem heterocromia utilizando as caraterísticas presentes

ALBINISMO=temAlbinismo(id, nome,idade,bonus,atratividadeBase, genoma, esperancaVida, apetiteReprodutivo, HETEROCROMIA); //Calcula e verifica se o animal é albino utilizando as caraterísticas presentes

incrementaBonus(atratividadeBase); //Incrementa ao bonus do animal a sua atratividade dependendo das mutações

}

public int getId() {

return id;

}

public String getNome() {

return nome;

}

public int getIdade() {

return idade;

}

public double getBonus() {

return bonus;

}

public static int getAnimaisZoo() {

return animaisZoo;

}

public boolean isHETEROCROMIA() {

return HETEROCROMIA;

}

public boolean isALBINISMO() {

return ALBINISMO;

}

public void setId(int id) {

this.id = id;

}

public void setNome(String nome) {

this.nome = nome;

}

public void setIdade(int idade) {

this.idade = idade;

}

public void setBonus(double bonus) {

this.bonus = bonus;

}

public static void setAnimaisZoo(int animaisZoo) {

Animal.animaisZoo = animaisZoo;

}

@Override

public String toString() {

return "Id= "+id+" Nome:"+nome+" Idade:"+idade+" Bonus:"+bonus+" Heterocromia:"+HETEROCROMIA+" Albinismo:"+ALBINISMO;

}

/\*

Método vindo da interface Heterocromia que retorna "true" se o número que é calculado usando o hash for menor que 5 e "false" se não,

Assim tencnicamente temos uma probabilidade de o animal ter heterocromia

\*/

@Override

public boolean temHeterocromia (Object...obj){

int prob = (Math.abs(Objects.hash(obj))%100);

return prob < 5;

}

/\*

Método vindo da interface Albinismo que retorna "true" se o número que é calculado usando o hash for menor que 10 e "false" se não,

Assim tencnicamente temos uma probabilidade de o animal ter albinismo

\*/

@Override

public boolean temAlbinismo (Object...obj){

int prob = (Math.abs(Objects.hash(obj))%100);

return prob < 10;

}

/\*

Método para simplificar a adição de um animal aumentando por um a variável de animais

\*/

public static void incrementaAnimais(){

animaisZoo++;

}

/\*

Método que verifica se o animal tem alguma mutação e ai incrementa ao bonús desse animal o valor da sua atratividade base

Com isto, dependendo do número de mutações que tem o animal a sua atratividade em geral multiplica essas vezes

\*/

public void incrementaBonus (double atratividadeBase){

if (HETEROCROMIA){

bonus+=atratividadeBase;

}

if (ALBINISMO){

bonus+=atratividadeBase;

}

}

/\*

Método para simplificar o aumento da idade do animal

\*/

public void incrementaIdade(){

idade++;

}

/\*

Método para retirar ao saldo do zoo o custo da comida do animal durante um ano

\*/

public abstract void comerAnual(Zoo zoo);

/\*

Método para retirar ao saldo do zoo o custo dos cuidadores do animal durante um ano

\*/

public abstract void cuidarAnual(Zoo zoo);

}

## 5.2. Boi

package Animais;

import Genomas.Bovino;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Boi

\*/

public class Boi extends Animal implements Bovino{

private static double atratividadeBase = 450; //Atratividade base do animal

private static final String GENOMA = "Bovino"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 50; //Apetite reprodutivo do animal

public Boi(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Boi.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Boi (" +super.toString()+" AtratividadeBase:"+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Boi durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Boi durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-200);

}

}

## 5.3. Cabra

package Animais;

import Genomas.Capra;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Cabra

\*/

public class Cabra extends Animal implements Capra{

private static double atratividadeBase = 350; //Atratividade base do animal

private static final String GENOMA = "Capra"; //Genoma do animal

private static final int ESPERANCAVIDA = 18; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 20; //Apetite reprodutivo do animal

public Cabra(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Cabra.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Cabra (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de uma Cabra durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-100);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de uma Cabra durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

}

## 5.4. Cachorro

package Animais;

import Genomas.Canis;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Cachorro

\*/

public class Cachorro extends Animal implements Canis {

private static double atratividadeBase = 200; //Atratividade base do animal

private static final String GENOMA = "Canis"; //Genoma do animal

private static final int ESPERANCAVIDA = 10; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 30; //Apetite reprodutivo do animal

public Cachorro(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Cachorro.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Cachorro (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Cachorro durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-50);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Cachorro durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-50);

}

}

## 5.5. Cavalo

package Animais;

import Genomas.Equus;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Cavalo

\*/

public class Cavalo extends Animal implements Equus{

private static double atratividadeBase = 500; //Atratividade base do animal

private static final String GENOMA = "Equus"; //Genoma do animal

private static final int ESPERANCAVIDA = 30; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 20; //Apetite reprodutivo do animal

public Cavalo(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Cavalo.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Cavalo (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Cavalo durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-200);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Cavalo durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-130);

}

}

## 5.6. Cobra

package Animais;

import Genomas.Ophidia;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Cobra

\*/

public class Cobra extends Animal implements Ophidia{

private static double atratividadeBase = 400; //Atratividade base do animal

private static final String GENOMA = "Ophidia"; //Genoma do animal

private static final int ESPERANCAVIDA = 25; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 50; //Apetite reprodutivo do animal

public Cobra(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Cobra.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Cobra (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de uma Cobra durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-50);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de uma Cobra durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-50);

}

}

## 5.7. Coelho

package Animais;

import Genomas.Oryctolagus;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Coelho

\*/

public class Coelho extends Animal implements Oryctolagus{

private static double atratividadeBase = 250; //Atratividade base do animal

private static final String GENOMA = "Oryctolagus"; //Genoma do animal

private static final int ESPERANCAVIDA = 9; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 70; //Apetite reprodutivo do animal

public Coelho(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Coelho.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Coelho (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Coelho durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-20);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Coelho durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-50);

}

}

## 5.8. Crocodilo

package Animais;

import Genomas.Crocodyloidea;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Crocodilo

\*/

public class Crocodilo extends Animal implements Crocodyloidea{

private static final double ATRATIVIDADEBASE = 750; //Atratividade base do animal

private static final String GENOMA = "Crocodyloidea"; //Genoma do animal

private static final int ESPERANCAVIDA = 70; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 50; //Apetite reprodutivo do animal

public Crocodilo(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Crocodilo (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Crocodilo durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-150);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Crocodilo durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-200);

}

}

## 5.9. Dragão

package Animais;

import Genomas.Dragon;

import Mutações.Dragonismo;

import com.mycompany.jumanji.Zoo;

import java.util.Objects;

/\*

Classe do animal Dragão

\*/

public class Dragão extends Animal implements Dragonismo, Dragon{

private static double atratividadeBase = 5000; //Atratividade base do animal

private static final String GENOMA = "Dragon"; //Genoma do animal

private static final int ESPERANCAVIDA = 100; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 1; //Apetite reprodutivo do animal

private final boolean DRAGONISMO; //Booleano que define se o dragão tem dragonismo, que é uma mutação uníca dos dragões

public Dragão(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

DRAGONISMO = temDragonismo();

if (DRAGONISMO){

super.setBonus(super.getBonus()+atratividadeBase);

}

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public boolean isDRAGONISMO() {

return DRAGONISMO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Dragão.atratividadeBase = atratividadeBase;

}

/\*

Método vindo da interface Dragonismo que retorna "true" se o número que é calculado usando o hash for menor que 20 e "false" se não,

Assim tencnicamente temos uma probabilidade de o animal ter dragonismo

\*/

@Override

public boolean temDragonismo (Object...obj){

int prob = (Math.abs(Objects.hash(obj))%100);

return prob < 20;

}

@Override

public String toString() {

return "Dragão (" +super.toString()+" Dragonismo:"+DRAGONISMO+" AtratividadeBase:"+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Dragão durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-1000);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Dragão durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-2000);

}

}

## 5.10. Galo

package Animais;

import Genomas.Gallus;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Galo

\*/

public class Galo extends Animal implements Gallus{

private static double atratividadeBase = 200; //Atratividade base do animal

private static final String GENOMA = "Gallus"; //Genoma do animal

private static final int ESPERANCAVIDA = 7; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 40; //Apetite reprodutivo do animal

public Galo(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Galo.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Galo (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Galo durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-30);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Galo durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-20);

}

}

## 5.11. Gorila

package Animais;

import Genomas.Primata;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Gorila

\*/

public class Gorila extends Animal implements Primata {

private static final double ATRATIVIDADEBASE = 800; //Atratividade base do animal

private static final String GENOMA = "Primata"; //Genoma do animal

private static final int ESPERANCAVIDA = 40; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 50; //Apetite reprodutivo do animal

public Gorila(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Gorila (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Gorila durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-150);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Gorila durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-300);

}

}

## 5.12. Leão

package Animais;

import Genomas.Panthera;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Leão

\*/

public class Leão extends Animal implements Panthera{

private static final double ATRATIVIDADEBASE = 1100; //Atratividade base do animal

private static final String GENOMA = "Panthera"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 20; //Apetite reprodutivo do animal

public Leão(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Leão (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Leão durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-300);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Leão durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-350);

}

}

## 5.13. Macaco

package Animais;

import Genomas.Bovino;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Vaca

\*/

public class Vaca extends Animal implements Bovino {

private static final double ATRATIVIDADEBASE = 250; //Atratividade base do animal

private static final String GENOMA = "Bovino"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 60; //Apetite reprodutivo do animal

public Vaca(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Vaca (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Vaca durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Vaca durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-100);

}

}

## 5.14. Panda

package Animais;

import Genomas.Urso;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Panda

\*/

public class Panda extends Animal implements Urso {

private static final double ATRATIVIDADEBASE = 1350; //Atratividade base do animal

private static final String GENOMA = "Urso"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 1; //Apetite reprodutivo do animal

public Panda(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Panda (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Panda durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-300);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Panda durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-200);

}

}

## 5.15. Porco

package Animais;

import Genomas.Bovino;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Vaca

\*/

public class Vaca extends Animal implements Bovino {

private static final double ATRATIVIDADEBASE = 250; //Atratividade base do animal

private static final String GENOMA = "Bovino"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 60; //Apetite reprodutivo do animal

public Vaca(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Vaca (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Vaca durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Vaca durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-100);

}

}

## 5.16. Rato

package Animais;

import Genomas.Roedor;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Rato

\*/

public class Rato extends Animal implements Roedor{

private static double atratividadeBase = 30; //Atratividade base do animal

private static final String GENOMA = "Roedor"; //Genoma do animal

private static final int ESPERANCAVIDA = 5; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 20; //Apetite reprodutivo do animal

public Rato(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,atratividadeBase,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getAtratividadeBase() {

return atratividadeBase;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

public static void setAtratividadeBase(double atratividadeBase) {

Rato.atratividadeBase = atratividadeBase;

}

@Override

public String toString() {

return "Rato (" +super.toString()+" AtratividadeBase: "+atratividadeBase+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método para simplificar o duplicamento da atratividade base por causa do ano chinês

\*/

public static void incrementaAtratividadeBase(){

atratividadeBase=atratividadeBase\*2;

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Rato durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-3);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Rato durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-5);

}

}

## 5.17. Tigre

package Animais;

import Genomas.Bovino;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Vaca

\*/

public class Vaca extends Animal implements Bovino {

private static final double ATRATIVIDADEBASE = 250; //Atratividade base do animal

private static final String GENOMA = "Bovino"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 60; //Apetite reprodutivo do animal

public Vaca(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Vaca (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Vaca durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Vaca durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-100);

}

}

## 5.18. UrsoPardo

package Animais;

import Genomas.Urso;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Urso-Pardo

\*/

public class UrsoPardo extends Animal implements Urso {

private static final double ATRATIVIDADEBASE = 900; //Atratividade base do animal

private static final String GENOMA = "Urso"; //Genoma do animal

private static final int ESPERANCAVIDA = 25; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 5; //Apetite reprodutivo do animal

public UrsoPardo(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Urso-Pardo (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Urso-Pardo durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-300);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Urso-Pardo durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-200);

}

}

## 5.19. UrsoPolar

package Animais;

import Genomas.Urso;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Urso-Polar

\*/

public class UrsoPolar extends Animal implements Urso {

private static final double ATRATIVIDADEBASE = 1400; //Atratividade base do animal

private static final String GENOMA = "Urso"; //Genoma do animal

private static final int ESPERANCAVIDA = 30; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 2; //Apetite reprodutivo do animal

public UrsoPolar(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Urso-Polar (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Urso-Polar durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-350);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Urso-Polar durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-300);

}

}

## 5.20. Vaca

package Animais;

import Genomas.Bovino;

import com.mycompany.jumanji.Zoo;

/\*

Classe do animal Vaca

\*/

public class Vaca extends Animal implements Bovino {

private static final double ATRATIVIDADEBASE = 250; //Atratividade base do animal

private static final String GENOMA = "Bovino"; //Genoma do animal

private static final int ESPERANCAVIDA = 20; //Esperança de vida média do animal

private static final int APETITEREPRODUTIVO = 60; //Apetite reprodutivo do animal

public Vaca(String nome, boolean adiciona){ //Recebe dois argumentos "nome" é o nome artistico do animal e "adiciona" e se é ou não para adicionar o animal ao zoo

super(nome,ATRATIVIDADEBASE,GENOMA,ESPERANCAVIDA,APETITEREPRODUTIVO);

if(adiciona){//Se for para adicionar o numero de animais do zoo aumenta, é dado um "id" ao animal

Animal.setAnimaisZoo(Animal.getAnimaisZoo()+1);

super.setId(Animal.getAnimaisZoo());

}

}

public static double getATRATIVIDADEBASE() {

return ATRATIVIDADEBASE;

}

public static String getGENOMA() {

return GENOMA;

}

public static int getESPERANCAVIDA() {

return ESPERANCAVIDA;

}

public static int getAPETITEREPRODUTIVO() {

return APETITEREPRODUTIVO;

}

@Override

public String toString() {

return "Vaca (" +super.toString()+" AtratividadeBase: "+ATRATIVIDADEBASE+" Genoma:"+GENOMA+" EsperançaDeVida:"+ESPERANCAVIDA+" ApetiteReprodutivo:"+APETITEREPRODUTIVO+")";

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo da comida de um Vaca durante um ano

\*/

@Override

public void comerAnual(Zoo zoo){

zoo.adicionarSaldo(-80);

}

/\*

Método vindo da superclasse Animal para retirar ao saldo do zoo o custo dos cuidadores de um Vaca durante um ano

\*/

@Override

public void cuidarAnual(Zoo zoo){

zoo.adicionarSaldo(-100);

}

}

## 5.21. AnimalIdAlreadyUsedF

package Exceções;

/\*

Classe da exceção quando o animal a ser inserido do ficheiro tem o mesmo id que um já no zoo

\*/

public class AnimalIdAlreadyUsedF extends Exception{

public AnimalIdAlreadyUsedF(){

super("Houve repetição de ids no animais presentes nos documentos de texto");

}

}

## 5.22. AnimalNotFound

package Exceções;

/\*

Classe quando utilizador instancia um animal que não existe

\*/

public class AnimalNotFound extends Exception {

public AnimalNotFound(){

super("Não foi encontrado o animal ou não existe");

}

}

## 5.23. InsAlreadyExistsF

/\*

Classe da exceção quando instalações no ficheiro têm números iguais

\*/

public class InsAlreadyExistsF extends Exception {

public InsAlreadyExistsF(){

super("Existem instalações no ficheiro que repetem o seu número");

}

}

## 5.24. InsNotFound

package Exceções;

/\*

Classe da exceção quando utilizador instancia uma instalação que não existe

\*/

public class InsNotFound extends Exception {

public InsNotFound(){

super("Não foi encontrada a instalação correspondente ao número ou essa não existe");

}

}

## 5.25. Bovino

package Genomas;

import Animais.Boi;

import Animais.Vaca;

import Animais.Animal;

import java.util.Random;

/\*

Classe da característica genética Bovino

\*/

public interface Bovino {

public static double preco = 275;

/\*

Método que retorna um animal com genoma Bovino

\*/

public static Animal criaBovino (){

Random rand = new Random();

int a = rand.nextInt(2);

if (a == 0){

return new Boi(null, true);

}

else{

return new Vaca(null, true);

}

}

public static double getPreco() {

return preco;

}

}

## 5.26. Canis

package Genomas;

import Animais.Cachorro;

import Animais.Animal;

/\*

Classe da característica genética Canis

\*/

public interface Canis {

public static double preco = 150;

/\*

Método que retorna um animal com genoma Canis

\*/

public static Animal criaCanis (){

return new Cachorro(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.27. Capra

package Genomas;

import Animais.Cabra;

import Animais.Animal;

/\*

Classe da característica genética Capra

\*/

public interface Capra {

public static double preco = 225;

/\*

Método que retorna um animal com genoma Capra

\*/

public static Animal criaCapra (){

return new Cabra(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.28. Crocodyloidea

package Genomas;

import Animais.Crocodilo;

import Animais.Animal;

/\*

Classe da característica genética Crocodyloidea

\*/

public interface Crocodyloidea {

public static double preco = 425;

/\*

Método que retorna um animal com genoma Crocodyloidea

\*/

public static Animal criaCrocodyloidea (){

return new Crocodilo(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.29. Dragon

package Genomas;

import Animais.Dragão;

import Animais.Animal;

/\*

Classe da característica genética Dragon

\*/

public interface Dragon {

public static double preco = 3000;

/\*

Método que retorna um animal com genoma Dragon

\*/

public static Animal criaDragon (){

return new Dragão(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.30. Equus

package Genomas;

import Animais.Cavalo;

import Animais.Animal;

/\*

Classe da característica genética Equus

\*/

public interface Equus {

public static double preco = 300;

/\*

Método que retorna um animal com genoma Equus

\*/

public static Animal criaEquus (){

return new Cavalo(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.31. Gallus

package Genomas;

import Animais.Galo;

import Animais.Animal;

/\*

Classe da característica genética Gallus

\*/

public interface Gallus {

public static double preco = 150;

/\*

Método que retorna um animal com genoma Gallus

\*/

public static Animal criaGallus (){

return new Galo(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.32. Ophidia

package Genomas;

import Animais.Cobra;

import Animais.Animal;

/\*

Classe da característica genética Ophidia

\*/

public interface Ophidia {

public static double preco = 250;

/\*

Método que retorna um animal com genoma Ophidia

\*/

public static Animal criaOphidia (){

return new Cobra(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.33. Oryctolagus

package Genomas;

import Animais.Coelho;

import Animais.Animal;

/\*

Classe da característica genética Oryctolagus

\*/

public interface Oryctolagus {

public static double preco = 150;

/\*

Método que retorna um animal com genoma Oryctolagus

\*/

public static Animal criaOryctolagus (){

return new Coelho(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.34. Panthera

package Genomas;

import Animais.Leão;

import Animais.Tigre;

import Animais.Animal;

import java.util.Random;

/\*

Classe da característica genética Panthera

\*/

public interface Panthera {

public static double preco = 560;

/\*

Método que retorna um animal com genoma Panthera

\*/

public static Animal criaPanthera (){

Random rand = new Random();

int a = rand.nextInt(2);

if (a == 0){

return new Leão(null, true);

}

else{

return new Tigre(null, true);

}

}

public static double getPreco() {

return preco;

}

}

## 5.35. Primata

package Genomas;

import Animais.Macaco;

import Animais.Gorila;

import Animais.Animal;

import java.util.Random;

/\*

Classe da característica genética Primata

\*/

public interface Primata {

public static double preco = 450;

/\*

Método que retorna um animal com genoma Primata

\*/

public static Animal criaPrimata (){

Random rand = new Random();

int a = rand.nextInt(2);

if (a == 0){

return new Macaco(null, true);

}

else{

return new Gorila(null, true);

}

}

public static double getPreco() {

return preco;

}

}

## 5.36. Roedor

package Genomas;

import Animais.Rato;

import Animais.Animal;

/\*

Classe da característica genética Roedor

\*/

public interface Roedor {

public static double preco = 20;

/\*

Método que retorna um animal com genoma Roedor

\*/

public static Animal criaRoedor (){

return new Rato(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.37. Suíno

package Genomas;

import Animais.Porco;

import Animais.Animal;

/\*

Classe da característica genética Suíno

\*/

public interface Suíno {

public static double preco = 150;

/\*

Método que retorna um animal com genoma Suíno

\*/

public static Animal criaSuíno (){

return new Porco(null,true);

}

public static double getPreco() {

return preco;

}

}

## 5.38. Urso

package Genomas;

import Animais.UrsoPardo;

import Animais.Panda;

import Animais.UrsoPolar;

import Animais.Animal;

import java.util.Random;

/\*

Classe da característica genética Urso

\*/

public interface Urso {

public static double preco = 550;

/\*

Método que retorna um animal com genoma Urso

\*/

public static Animal criaUrso (){

Random rand = new Random();

int a = rand.nextInt(3);

return switch (a) {

case 0 -> new Panda(null, true);

case 1 -> new UrsoPardo(null, true);

default -> new UrsoPolar(null, true);

};

}

public static double getPreco() {

return preco;

}

}

## 5.39. Albinismo

package Mutações;

/\*

Classe da mutação Albinismo

\*/

public interface Albinismo {

public abstract boolean temAlbinismo (Object...obj);

}

## 5.40. Dragonismo

package Mutações;

/\*

Classe da mutação Dragonismo

\*/

public interface Dragonismo {

public abstract boolean temDragonismo (Object...obj);

}

## 5.41. Heterocromia

package Mutações;

/\*

Classe da mutação Heterocromia

\*/

public interface Heterocromia {

public abstract boolean temHeterocromia (Object...obj);

}

## 5.42. Adquirido

package Ocorrências;

import Animais.Animal;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de adiquirir um animal

\*/

public class Adquirido extends Ocorrência {

private Animal animal; //Animal adiquirido

public Adquirido (Animal animal, Zoo zoo){ //Recebe o animal adiquirido e o seu zoo

super(zoo);

this.animal = animal;

}

public Animal getAnimal() {

return animal;

}

public void setAnimal(Animal animal) {

this.animal = animal;

}

@Override

public String toString(){

return super.toString()+" Foi adquirido um "+animal.getClass()+" chamado "+animal.getNome()+" que ficou com id "+animal.getId();

}

}

## 5.43. Colocação

package Ocorrências;

import Animais.Animal;

import com.mycompany.jumanji.Instalação;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de colocar um animal numa instalação

\*/

public class Colocação extends Ocorrência{

private Animal animal; //Animal instalado

private Instalação instalacao; //Instalação onde foi instalado

public Colocação (Animal animal, Instalação instalacao, Zoo zoo){ //Recebe o animal instalado, a tal instalação e o seu zoo

super(zoo);

this.animal = animal;

this.instalacao = instalacao;

}

public Animal getAnimal() {

return animal;

}

public Instalação getInstalacao() {

return instalacao;

}

public void setAnimal(Animal animal) {

this.animal = animal;

}

public void setInstalacao(Instalação instalacao) {

this.instalacao = instalacao;

}

@Override

public String toString(){

return super.toString()+" Foi colocado na instalação de número "+instalacao.getNumeroInstalacao()+" o "+animal.getClass()+" de id "+animal.getId();

}

}

## 5.44. Construção

package Ocorrências;

import com.mycompany.jumanji.Instalação;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de construir uma instalação

\*/

public class Construção extends Ocorrência {

private Instalação instalacao; //Instalação construida

public Construção (Instalação instalacao, Zoo zoo){ //Recebe a instalação construida e o seu zoo

super(zoo);

this.instalacao = instalacao;

}

public Instalação getInstalacao() {

return instalacao;

}

public void setInstalacao(Instalação instalacao) {

this.instalacao = instalacao;

}

@Override

public String toString(){

return super.toString()+" Foi construida uma instalação com lotação "+instalacao.getLotacao()+" que custou "+instalacao.getPrecoInstalacao()+" e ficou com o número "+instalacao.getNumeroInstalacao();

}

}

## 5.45. Fugiu

package Ocorrências;

import Animais.Animal;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de fugir um animal

\*/

public class Fugiu extends Ocorrência {

private Animal animal; //Animal que fugiu

public Fugiu (Animal animal, Zoo zoo){ //Recebe o animal que fugiu e o seu zoo

super(zoo);

this.animal = animal;

}

public Animal getAnimal() {

return animal;

}

public void setAnimal(Animal animal) {

this.animal = animal;

}

@Override

public String toString(){

return super.toString()+" Fugiu um "+animal.getClass()+" chamado "+animal.getNome()+" de id "+animal.getId();

}

}

## 5.46. Nascimento

package Ocorrências;

import Animais.Animal;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de nascer um animal

\*/

public class Nascimento extends Ocorrência {

private Animal animal; //Animal que nasceu

public Nascimento (Animal animal, Zoo zoo){ //Recebe o animal nascido e o seu zoo

super(zoo);

this.animal = animal;

}

public Animal getAnimal() {

return animal;

}

public void setAnimal(Animal animal) {

this.animal = animal;

}

@Override

public String toString(){

return super.toString()+" Nasceu um "+animal.getClass()+" chamado "+animal.getNome()+" que ficou com id "+animal.getId();

}

}

## 5.47. Ocorrência

package Ocorrências;

import com.mycompany.jumanji.Zoo;

/\*

Classe das ocorrências

\*/

public class Ocorrência {

private Zoo zoo; //Zoo onde aconteceu a ocorrência

public Ocorrência(Zoo zoo){ //Recebe o seu zoo

this.zoo = zoo;

}

public Zoo getZoo() {

return zoo;

}

public void setZoo(Zoo zoo) {

this.zoo = zoo;

}

@Override

public String toString(){

return "\n No zoo "+zoo.getNomeZoo()+" no ano "+zoo.getAnoZoo()+":";

}

}

## 5.48. Óbito

package Ocorrências;

import Animais.Animal;

import com.mycompany.jumanji.Zoo;

/\*

Classe da ocorrência de morrer um animal

\*/

public class Óbito extends Ocorrência {

private Animal animal; //Animal que morreu

public Óbito (Animal animal, Zoo zoo){ //Recebe o animal que morreu e o seu zoo

super(zoo);

this.animal = animal;

}

public Animal getAnimal() {

return animal;

}

public void setAnimal(Animal animal) {

this.animal = animal;

}

@Override

public String toString(){

return super.toString()+" Morreu um "+animal.getClass()+" chamado "+animal.getNome()+" que tinha o id "+animal.getId();

}

}

## 5.49. Instalação

package com.mycompany.jumanji;

import Animais.Animal;

import java.util.ArrayList;

import java.util.Random;

import java.util.Scanner;

/\*

Classe das instalações do zoo

\*/

public class Instalação {

private int numeroInstalacao; //Inteiro que cada instalação tem par serem distingidas, ou seja o seu id

private int lotacao; //Tamanho da instalação (quantos animais pode ter, isto é, o tamanho do array)

private int animaisInstalados; //Número de animais na instalação

private double precoInstalacao; //Preço de construção da instalação

private ArrayList<Animal> animais; //Array de animais da instalação (onde são guardados os animas que estão na instalação)

private static int instalacoes; //Número de instalações adicionadas ao zoo, serve para criar seus numeros de instalação

public Instalação(boolean constroi){ //O construtor recebe um boleano "constroi" que serve para saber se é ou não para adicionar a instalação ao zoo

lotacao = geraLotacao(); //Gera uma lotação qualquer

animais = new ArrayList<>(lotacao);

animaisInstalados = 0;

precoInstalacao = geraPreco(); //Gera um preço de construção para a instalação

numeroInstalacao = instalacoes+1; //Diz que o numero da instalação é o número de instalações no zoo mais 1 (O seu numero e instalacoes vão realmente ser mudados se for para a construir)

if(constroi){ //Se for para construir, aumenta o número de instalações no zoo e usa esse número como número da nova instalação

instalacoes++;

numeroInstalacao = instalacoes;

}

}

public ArrayList<Animal> getAnimais() {

return animais;

}

public int getNumeroInstalacao() {

return numeroInstalacao;

}

public static int getInstalacoes() {

return instalacoes;

}

public double getPrecoInstalacao() {

return precoInstalacao;

}

public int getLotacao() {

return lotacao;

}

public int getAnimaisInstalados() {

return animaisInstalados;

}

public void setAnimais(ArrayList<Animal> animais) {

this.animais = animais;

}

public void setNumeroInstalacao(int numeroInstalacao) {

this.numeroInstalacao = numeroInstalacao;

}

public static void setInstalacoes(int instalacoes) {

Instalação.instalacoes = instalacoes;

}

public void setPrecoInstalacao(double precoInstalacao) {

this.precoInstalacao = precoInstalacao;

}

public void setLotacao(int lotacao) {

this.lotacao = lotacao;

}

public void setAnimaisInstalados(int animaisInstalados) {

this.animaisInstalados = animaisInstalados;

}

@Override

public String toString() {

return "Instalação{ NúmeroDaInstalação="+numeroInstalacao+", Lotação="+lotacao+", Animais instalados="+animaisInstalados+", Preço="+precoInstalacao+", Animais="+animais+"}";

}

/\*

Método para simplificar a adição de uma instalação aumentando por um a variável de instalações

\*/

public static void incrementaInstalacoes(){

Instalação.setInstalacoes(Instalação.getInstalacoes()+1);

}

/\*

Método que usando um random retorna uma lotação para a instalação que vai de 1 a 10

\*/

private int geraLotacao (){

Random rand = new Random();

int a = rand.nextInt(10)+1;

return a;

}

/\*

Método que usando um random e a lotação retorna um preço de construção para a instalação

\*/

private double geraPreco (){

Random rand = new Random();

double a = rand.nextInt(500)+250;

return a\*lotacao; //Random que vai de 250 a 749, vezes, a lotação da instalação que vai de 1 a 10

}

/\*

Método para adicionar um animal á instalação

Verifica se a instalação está cheia e se estiver pede ao utilizador para escolher um animal para substituir pelo que quer inserir

O animal substituído vai para o ArrayList dos animais fora do zoo e o a inserir é inserido na ArrayList da instalação

\*/

public void adicionaAnimal (Zoo zoo, Animal animal){

if(animaisInstalados<lotacao){ //Se a instalação não estiver cheia

animais.add(animal);

animaisInstalados++;

}

else{ //Se estiver cheia

System.out.println("Esta instalação está cheia, qual animal quer trocar?");

for(int i = 0;i<animais.size();i++){

System.out.println(i+") "+animais.get(i));

}

Scanner scan = new Scanner(System.in);

System.out.print("Animal:");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para escolher uma opção tem de escrever um dos números inteiros apresentados no inicio de cada animal");

System.out.print("Animal:");

scan.next();

}

int opcao = scan.nextInt();

zoo.adicionarFI(animais.get(opcao)); //Adiciona o animal substituído ao ArrayList dos animais fora das instalações

animais.remove(opcao); //Remove o animal substituído da instalação

animais.add(animal); //Adiciona o animal a inserir á ArrayList da instalação

}

}

/\*

Método para retirar ao saldo do zoo o custo da manutenção em geral da instalação

Depende da lotação desta, pois numa instalação maior lógicamente será mais cara a manutenção

\*/

public void manutencaoInstalacaoAno(Zoo zoo){

zoo.adicionarSaldo(-(lotacao\*200));

}

}

## 5.50. Jumanji

package com.mycompany.jumanji;

import Animais.Animal;

import Animais.Boi;

import Animais.Cabra;

import Animais.Cachorro;

import Animais.Cavalo;

import Animais.Cobra;

import Animais.Coelho;

import Animais.Crocodilo;

import Animais.Dragão;

import Animais.Galo;

import Animais.Gorila;

import Animais.Leão;

import Animais.Macaco;

import Animais.Panda;

import Animais.Porco;

import Animais.Rato;

import Animais.Tigre;

import Animais.UrsoPardo;

import Animais.UrsoPolar;

import Animais.Vaca;

import Exceções.AnimalIdAlreadyUsedF;

import Exceções.InsAlreadyExistsF;

import Exceções.InsNotFound;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.Map;

import java.util.Scanner;

/\*

Classe principal

\*/

public class Jumanji {

public static void main(String[] args) throws IOException {

System.out.println("\n---------------------Opções de Início---------------------");

System.out.println("1. Carregar dados");

System.out.println("2. Novo Zoo");

int opcao;

Scanner scan = new Scanner(System.in);

System.out.print("Selecione uma opção: ");

while(!scan.hasNextInt()){

System.out.println("Para escolher uma opção tem de escrever um dos dois números inteiros apresentados no início de cada opção de início");

System.out.print("Selecione uma opção: ");

scan.next();

}

opcao = scan.nextInt(); //Utilizamos scanner para saber a opção que o utilizador escolheu

switch (opcao){

case 1: //Se for para carregar zoo apartir dos ficheiros

Zoo zoo = new Zoo(false);

try{

carregaZoo(zoo); //Carrega as informações do zoo apartir do ficheiro

carregaAnimaisFI(zoo); //Carrega as informações dos animais fora das instalações apartir do ficheiro

carregaInstalaçao(zoo); //Carrega as informações das instalações apartir do ficheiro

carregarAnimaisI(zoo); //Carrega as informações dos animais dentro das instalações apartir do ficheiro

zoo.menu(); //Inicializa o menu

}

catch(IOException i){ //Se houver algum erro no carregamento

System.out.println("Erro no carregamento");

}

catch (AnimalIdAlreadyUsedF | InsAlreadyExistsF | InsNotFound ex){

System.out.println(ex.getMessage());

}

break;

case 2: //Se for para criar novo zoo

zoo = new Zoo(true);

break;

default:

System.out.println("Opção inválida");

break;

}

}

/\*

Método para carregar as informações do zoo

\*/

public static void carregaZoo (Zoo zoo) throws IOException {

String nomeZoo; //Nome do zoo

int anoZoo; //Ano do zoo

String line;

FileReader inStream = new FileReader("Zoo.txt");

try (BufferedReader lerDados = new BufferedReader(inStream)) { //Usamos um buffer para ler linha a linha do ficheiro

line = lerDados.readLine();

while (line != null) {

nomeZoo = lerDados.readLine();

anoZoo = Integer.parseInt(lerDados.readLine().trim());

zoo.setNomeZoo(nomeZoo);

zoo.setAnoZoo(anoZoo);

line = lerDados.readLine();

}

lerDados.close();

}

}

/\*

Método para carregar as informações dos animais fora da instalações no zoo e inserilos nesse ArrayList

\*/

public static void carregaAnimaisFI(Zoo zoo) throws IOException, AnimalIdAlreadyUsedF {

Animal animal;

int id; //Id do animal

String nome; //Nome do animal

int idade; //Idade do animal

String line;

FileReader inStream = new FileReader("AnimaisZoo.txt");

try (BufferedReader lerDados = new BufferedReader(inStream)) { //Usamos um buffer para ler linha a linha do ficheiro

line = lerDados.readLine();

while (line != null) {

switch (line) {

case "Leão:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Leão(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Boi:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Boi(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Cabra:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cabra(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Cachorro:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cachorro(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Cavalo:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cavalo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Coelho:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Coelho(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Cobra:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cobra(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Crocodilo:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Crocodilo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Dragão:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Dragão(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Galo:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Galo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Gorila:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Gorila(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Macaco:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Macaco(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Panda:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Panda(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Porco:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Porco(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Rato:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Rato(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Tigre:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Tigre(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "UrsoPardo:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new UrsoPardo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "UrsoPolar:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new UrsoPolar(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

case "Vaca:":

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Vaca(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarFI(animal);

break;

}

line = lerDados.readLine();

}

lerDados.close();

}

}

/\*

Método para carregar as informações das instalações do zoo

\*/

public static void carregaInstalaçao(Zoo zoo)throws IOException,InsAlreadyExistsF{

Instalação inst;

int numeroInstalacao;

int lotacao;

String line;

FileReader inStream = new FileReader("ZooInstalação.txt");

try (BufferedReader lerDados = new BufferedReader(inStream)) { //Usamos um buffer para ler linha a linha do ficheiro

line = lerDados.readLine();

while (line != null) {

numeroInstalacao = Integer.parseInt(lerDados.readLine().trim());

for (Map.Entry instalacao : zoo.getZoo().entrySet()){

int key = (int)instalacao.getKey();

if(numeroInstalacao == key){

throw new InsAlreadyExistsF();

}

}

lotacao = Integer.parseInt(lerDados.readLine().trim());

inst = new Instalação(false);

inst. setLotacao(lotacao);

inst.setNumeroInstalacao(numeroInstalacao);

if (Instalação.getInstalacoes()<numeroInstalacao){

Instalação.setInstalacoes(numeroInstalacao);

}

zoo.adicionarInstalacao(inst);

line = lerDados.readLine();

}

lerDados.close();

}

}

/\*

Método para carregar as informações dos animais dentro da instalações do zoo e inserir-los na ArayList dessa instalação

\*/

public static void carregarAnimaisI(Zoo zoo) throws IOException,AnimalIdAlreadyUsedF,InsNotFound{

int numeroI ;//Número da instalçao

boolean existe;

Animal animal;

int id; //Id do animal

String nome; //Nome do animal

int idade; //Idade do animal

String line;

FileReader inStream = new FileReader("AnimaisZooI.txt");

try (BufferedReader lerDados = new BufferedReader(inStream)) { //Usamos um buffer para ler linha a linha do ficheiro

line = lerDados.readLine();

while (line != null) {

switch (line) {

case "Leão:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceção

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Leão(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Boi:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Boi(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Cabra:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cabra(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Cachorro:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cachorro(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Cavalo:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cavalo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Coelho:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Coelho(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Cobra:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Cobra(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Crocodilo:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Crocodilo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Dragão:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Dragão(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Galo:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Galo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Gorila:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Gorila(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Macaco:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Macaco(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Panda:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Panda(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Porco:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Porco(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Rato:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Rato(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Tigre:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Tigre(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "UrsoPardo:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new UrsoPardo(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "UrsoPolar:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new UrsoPolar(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

case "Vaca:":

numeroI=Integer.parseInt(lerDados.readLine().trim());

existe = false;

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numeroI == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

id= Integer.parseInt(lerDados.readLine().trim());

for(Animal a : zoo.getAnimaisFI()){ //Percorre a ArrayList dos animais fora das instalações do zoo

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

for (Map.Entry instalacao : zoo.getZoo().entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

if(a.getId() == id){ //Se houver um id igual ao do animal que estamos a ler

throw new AnimalIdAlreadyUsedF(); //Lança a exceçaõ

}

}

}

nome = lerDados.readLine().trim();

idade = Integer.parseInt(lerDados.readLine().trim());

animal=new Vaca(null, false);

animal.setNome(nome);

animal.setId(id);

if (Animal.getAnimaisZoo()<id){ //Se o id do animal que estamos a ler é maior que o número de animais ja adicionados no zoo

Animal.setAnimaisZoo(id); //Igualamos o o número de animais ao id

}

animal.setIdade(idade);

zoo.adicionarAI(numeroI,animal);

break;

}

line = lerDados.readLine();

}

lerDados.close();

}

}

}

## 5.51. Zoo

package com.mycompany.jumanji;

import Ocorrências.Ocorrência;

import Ocorrências.Fugiu;

import Ocorrências.Construção;

import Ocorrências.Adquirido;

import Ocorrências.Colocação;

import Ocorrências.Nascimento;

import Ocorrências.Óbito;

import Animais.Animal;

import Animais.Boi;

import Animais.Cabra;

import Animais.Cachorro;

import Animais.Cavalo;

import Animais.Cobra;

import Animais.Coelho;

import Animais.Crocodilo;

import Animais.Dragão;

import Animais.Galo;

import Animais.Gorila;

import Animais.Leão;

import Animais.Macaco;

import Animais.Panda;

import Animais.Porco;

import Animais.Rato;

import Animais.Tigre;

import Animais.UrsoPardo;

import Animais.UrsoPolar;

import Animais.Vaca;

import Genomas.Bovino;

import Genomas.Canis;

import Genomas.Capra;

import Genomas.Crocodyloidea;

import Genomas.Dragon;

import Genomas.Equus;

import Genomas.Gallus;

import Genomas.Ophidia;

import Genomas.Oryctolagus;

import Genomas.Panthera;

import Genomas.Primata;

import Genomas.Roedor;

import Genomas.Suíno;

import Genomas.Urso;

import Exceções.InsNotFound;

import Exceções.AnimalNotFound;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.Random;

import java.util.Scanner;

/\*

Classe do zoo

\*/

public class Zoo {

private String nomeZoo; //Nome do zoo

private int anoZoo; //Ano presente

private HashMap<Integer, Instalação> zoo; //HashMap onde as "keys" são os numeros das instalações e os "values" saõ as instalações

private ArrayList<Animal> animaisFI; //Array de animais onde são armazenados os animais recem-criados e/ou fora das instalações

private ArrayList<Animal> obituario; //Array da animais onde são armazenados os animais mortos

private ArrayList<Ocorrência> historico; //Array de ocorrências onde são armazenadas todas as ocorrências no zoo

private double saldo; //Saldo geral do zoo

public Zoo(boolean receba){ //Recebe um argumento "receba" booleano que diz é ou não para o zoo receber o nome e o ano do utilizador

zoo = new HashMap<>();

animaisFI = new ArrayList<>();

obituario = new ArrayList<>();

historico = new ArrayList<>();

saldo = 0;

if (receba){ //Se for para receber o nome e o ano do zoo pelo o utilizador

recebeNome(); //Pede ao utilizador o nome do zoo

recebeAno(); //Pede ao utilizador o ano do zoo

menu(); //Abre o menu do zoo

}

}

public String getNomeZoo() {

return nomeZoo;

}

public int getAnoZoo() {

return anoZoo;

}

public HashMap<Integer, Instalação> getZoo() {

return zoo;

}

public ArrayList<Animal> getAnimaisFI() {

return animaisFI;

}

public ArrayList<Animal> getObituario() {

return obituario;

}

public ArrayList<Ocorrência> getHistorico() {

return historico;

}

public double getSaldo() {

return saldo;

}

public void setNomeZoo(String nomeZoo) {

this.nomeZoo = nomeZoo;

}

public void setAnoZoo(int anoZoo) {

this.anoZoo = anoZoo;

}

public void setZoo(HashMap<Integer, Instalação> zoo) {

this.zoo = zoo;

}

public void setAnimaisFI(ArrayList<Animal> animaisFI) {

this.animaisFI = animaisFI;

}

public void setObituario(ArrayList<Animal> obituario) {

this.obituario = obituario;

}

public void setHistorico(ArrayList<Ocorrência> historico) {

this.historico = historico;

}

public void setSaldo(double saldo) {

this.saldo = saldo;

}

/\*

Método que pede o nome do zoo ao utilizador

\*/

private void recebeNome (){

Scanner scan = new Scanner(System.in);

System.out.print("Nome do seu Zoo:");

while(!scan.hasNext()){ //Verificar que o que o utilizador escreveu é uma string

System.out.println("O nome do zoo tem de ser uma String!");

System.out.print("Nome do seu Zoo:");

scan.next();

}

nomeZoo = scan.next();

while (!nomeZoo.matches("[a-zA-Z]+")){ //Verificar se essa string é uma string de letras

System.out.println("O nome do zoo tem de ser uma String de só letras!");

System.out.print("Nome do seu Zoo:");

nomeZoo = scan.next();

}

}

/\*

Método que pede o ano do zoo ao utilizador

\*/

private void recebeAno (){

Scanner scan = new Scanner(System.in);

System.out.print("Qual o ano de fundação:");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("O ano de fundalção tem de ser um número inteiro");

System.out.print("Qual o ano de fundação:");

scan.next();

}

anoZoo = scan.nextInt();

}

/\*

Método que adiciona um animal ao array dos animais criados/fora das instalações

\*/

public void adicionarFI (Animal animal){

animaisFI.add(animal);

}

/\*

Método que remove um animal do array dos animais criados/fora das instalações

\*/

public void removerFI (int animal){

animaisFI.remove(animal);

}

/\*

Método que adiciona uma animal ao array dos animais mortos

\*/

public void adicionarObitos (Animal animal){

obituario.add(animal);

}

/\*

Método que adiciona uma instalação ao zoo/HashMap

\*/

public void adicionarInstalacao (Instalação instalacao){

zoo.put(instalacao.getNumeroInstalacao(), instalacao);

}

/\*

Método que adiciona um animal a uma instalação utilizando o seu número

\*/

public void adicionarAI (int numeroInstalacao, Animal animal){

zoo.get(numeroInstalacao).adicionaAnimal(this, animal);

}

/\*

Método que adiciona uma ocorrência ao histórico

\*/

public void adicionarOcorrencia (Ocorrência ocorrencia){

historico.add(ocorrencia);

}

/\*

Método que adiciona ao saldo um número

\*/

public void adicionarSaldo (double saldo){

this.saldo+=saldo;

}

/\*

Método que adiciona um ano ao zoo

\*/

public void aumentarAno(){

anoZoo++;

}

/\*

Método do menu do zoo

\*/

public void menu(){

System.out.println("\n---------------------Menu Zoo "+nomeZoo+"---------------------");

System.out.println("1. Adquirir animal aleatório");

System.out.println("2. Adquirir animal com característica genética");

System.out.println("3. Construir instalação");

System.out.println("4. Colocar animal em instalação");

System.out.println("5. Calendário chinês");

System.out.println("6. Listar animais");

System.out.println("7. Listar animais com dada característica genética");

System.out.println("8. Listar animais com dada mutação");

System.out.println("9. Listar instalações");

System.out.println("10. Retrato de família animal");

System.out.println("11. Obituário");

System.out.println("12. Histórico");

System.out.println("13. Período Contabilístico");

System.out.println("14. Jumanji");

System.out.println("15. Sair da aplicação");

int opcao;

Scanner scan = new Scanner(System.in);

System.out.print("Selecione uma opção: ");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para escolher uma opção tem de escrever um dos números inteiros apresentados no inicio de cada opção");

System.out.print("Selecione uma opção: ");

scan.next();

}

opcao = scan.nextInt(); //Utilizamos scanner para saber a opção que o utilizador escolheu

switch (opcao) {

case 1:

//Criamos 3 animais "parciais" randomizados, mas só um será escolhido pelo utilizador

Animal animal1 = animalRandomizado();

Animal animal2 = animalRandomizado();

Animal animal3 = animalRandomizado();

System.out.println("Escolha um animal a adiquirir!");

System.out.println("1. "+animal1);

System.out.println("2. "+animal2);

System.out.println("3. "+animal3);

System.out.print("Selecione uma opção: ");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para escolher uma opção tem de escrever um três dos número inteiros apresentados no inicio de cada animal");

System.out.print("Selecione uma opção: ");

scan.next();

}

opcao = scan.nextInt(); //Utilizamos scanner para saber a opção que o utilizador escolheu

switch (opcao) {

case 1 :

constroiAnimalID(animal1, "adquirido"); //Construimos o animal que ainda não tem id nem esta adicionado ao zoo

comprarAnimal(animal1); //Tiramos ao saldo o custo desse animal

System.out.println("Foi criado um novo "+animal1.getClass()+" chamado "+animal1.getNome());

voltarMenu();

break;

case 2 :

constroiAnimalID(animal2, "adquirido"); //Construimos o animal que ainda não tem id nem esta adicionado ao zoo

comprarAnimal(animal2); //Tiramos ao saldo o custo desse animal

System.out.println("Foi criado um novo "+animal2.getClass()+" chamado "+animal2.getNome());

voltarMenu();

break;

case 3 :

constroiAnimalID(animal3, "adquirido"); //Construimos o animal que ainda não tem id

comprarAnimal(animal3); //Tiramos ao saldo o custo desse animal

System.out.println("Foi criado um novo "+animal3.getClass()+" chamado "+animal3.getNome()+"");

voltarMenu();

break;

default: //Se escolher uma opção inválida

System.out.println("Opção inválida.");

voltarMenu();

}

break;

case 2 :

String genoma;

System.out.print("Escreva o genoma que vai querer que o animal tenha: ");

genoma = scan.next(); //Utilizamos scanner para saber o genoma que o utilizador escolheu

Animal animalC = animalGenoma(genoma); //Cria animal randomizado, mas com genoma que o utilizador deu

if(animalC == null){ //Se o animal for nulo (quer dizer que no o método "animalGenoma" não está presente o genoma dado pelo utilizador)

System.out.println("O genoma referido não existe!");

}

else{

constroiAnimal(animalC, "adquirido"); //Construimos o animal sendo que este ja tem id

comprarGenoma(genoma); //Tiramos ao saldo o custo de um animal desse genoma

System.out.println("Foi criado um novo "+animalC.getClass()+" chamado "+animalC.getNome()+"");

}

voltarMenu();

break;

case 3 :

//Criamos 3 instalções "parciais" randomizadas, mas só uma ou nenhuma será escolhido pelo utilizador

Instalação inst1 = new Instalação(false);

Instalação inst2 = new Instalação(false);

Instalação inst3 = new Instalação(false);

System.out.println("Escolha um animal a adiquirir!");

System.out.println("1. Primeiro Candidato: "+inst1);

System.out.println("2. Segundo Candidato: "+inst2);

System.out.println("3. Terceiro Candidato: "+inst3);

System.out.println("0. Não escolher nenhuma");

System.out.print("Selecione uma opção: ");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para escolher uma opção tem de escrever um dos quatro números inteiros apresentados no inicio de cada instalação");

System.out.print("Selecione uma opção: ");

scan.next();

}

opcao = scan.nextInt(); //Utilizamos scanner para saber a opção que o utilizador escolheu

switch (opcao) {

case 0 : //Se escolher não construir nenhuma

System.out.println("Não foi construida nenhuma nova instalação");

voltarMenu();

break;

case 1 :

constroiInstalacao(inst1); //Construimos a instalação que ainda não tem id nem está no zoo

System.out.println("Foi escolhida e construida a instalação proposta pelo primeiro candidato");

voltarMenu();

break;

case 2 :

constroiInstalacao(inst2); //Construimos a instalação que ainda não tem id nem está no zoo

System.out.println("Foi escolhida e construida a instalação proposta pelo segundo candidato");

voltarMenu();

break;

case 3 :

constroiInstalacao(inst3); //Construimos a instalação que ainda não tem id nem está no zoo

System.out.println("Foi escolhida e construida a instalação proposta pelo terceiro candidato");

voltarMenu();

break;

default: //Se escolher uma opção inválida

System.out.println("Opção inválida.");

voltarMenu();

}

break;

case 4 :

instalarAnimal(); //Colocamos um animal dos que temos no array dos criados numa instalação

voltarMenu();

break;

case 5 :

anoChines(); //Verificamos o ano e aumentamos a atratividade base do animal desse ano no calendário chinês

voltarMenu();

break;

case 6 :

listarAnimais(); //Listamos todos os animais fora e dentro das instalações

voltarMenu();

break;

case 7 :

listarAnimaisG(); //Listamos todos os animais de um certo genoma fornecido pelo utilizador fora e dentro das instalações

voltarMenu();

break;

case 8 :

listarAnimaisM(); //Listamos todos os animais com uma certa mutação fornecida pelo utilizador fora e dentro das instalações

voltarMenu();

break;

case 9 :

listarInstalacoes(); //Listamos todas as instalações do zoo

voltarMenu();

break;

case 10 :

retratoF(); //Imprimimos todo o zoo

voltarMenu();

break;

case 11 :

System.out.println("Obituário :\n"+obituario); //Imprimimos o obituário

voltarMenu();

break;

case 12 :

System.out.println("Histórico :\n"+historico); //Imprimimos o histórico

voltarMenu();

break;

case 13 :

System.out.println("Período contabilistico do ano "+anoZoo);

periodoContabilistico();

voltarMenu();

break;

case 14 :

System.out.println("Devido a uma confusão no zoo foram perdidos de vista os animais!");

jumanji();

voltarMenu();

break;

case 15 :

System.out.println("Saiu da aplicação!");

break;

default :

System.out.println("Opção inválida!");

voltarMenu();

break;

}

}

/\*

Método usado para voltar ao menu principal do zoo

\*/

public void voltarMenu(){

System.out.println("Clique [0] para voltar ao menu");

Scanner scan = new Scanner(System.in);

int option;

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para voltar tem de clicar no NÚMERO [0] do seu teclado");

scan.next();

}

option = scan.nextInt();

if(option == 0){ //Se utilizador escreu 0

menu();

}

else{ //Se não executa o método denovo

voltarMenu();

}

}

/\*

Método usado para contruir um animal (neste caso completar um animal incompleto)

Este método recebe um animal que ainda não tem nome nem esta no zoo (logo não tem id)

\*/

public void constroiAnimalID (Animal animal, String ocorrencia){

Scanner scan = new Scanner(System.in);

String nomeA;

System.out.print("Digite o nome do animal: ");

while(!scan.hasNext()){ //Verifica se o que o utilizador escreveu é uma string

System.out.println("O nome do animal tem de ser uma String!");

System.out.print("Digite o nome do animal: ");

scan.next();

}

nomeA = scan.next(); //Utilizamos scanner para saber o nome artístico que o utilizador escolheu para o animal

while (!nomeA.matches("[a-zA-Z]+")){ //Verifica se o que o utilizador escreveu é uma string de letras

System.out.println("O nome do animal tem de ser uma String de só letras!");

System.out.print("Digite o nome do animal: ");

nomeA = scan.next();

}

Animal.incrementaAnimais(); //Aumenta o número de animais já criados no zoo

animal.setId(Animal.getAnimaisZoo()); //Dá um id ao animal escolhido, dependendo do numero de animais já criados no zoo

animal.setNome(nomeA); //Dá o nome artístico que o utilizador escolheu ao animal escolhido

adicionarFI(animal); //Adiciona esse animal ao array dos animais fora das instalações

criaOcorrencia(ocorrencia, animal,null); //Cria a ocorrência relacionada a este animal (se foi adquirido ou nasceu)

}

/\*

Método usado para contruir um animal (neste caso completar um animal incompleto)

Este método recebe um animal que ainda não tem nome mas está no zoo (logo já tem id)

\*/

public void constroiAnimal (Animal animal, String ocorrencia){

Scanner scan = new Scanner(System.in);

String nomeA;

System.out.print("Digite o nome do animal: ");

while(!scan.hasNext()){ //Verifica se o que o utilizador escreveu é uma string

System.out.println("O nome do animal tem de ser uma String!");

System.out.print("Digite o nome do animal: ");

scan.next();

}

nomeA = scan.next(); //Utilizamos scanner para saber o nome artístico que o utilizador escolheu para o animal

while (!nomeA.matches("[a-zA-Z]+")){ //Verifica se o que o utilizador escreveu é uma string de letras

System.out.println("O nome do animal tem de ser uma String de só letras!");

System.out.print("Digite o nome do animal: ");

nomeA = scan.next();

}

animal.setNome(nomeA); //Dá o nome artístico que o utilizador escolheu ao animal escolhido

adicionarFI(animal); //Adiciona esse animal ao array dos animais fora das instalações

criaOcorrencia(ocorrencia, animal,null); //Cria a ocorrência relacionada a este animal (se foi adquirido ou nasceu)

}

/\*

Método usado para construir uma instalação (neste caso completar uma instalação incompleta)

Este método recebe uma instalação que ainda não está no zoo (logo não tem número)

\*/

public void constroiInstalacao (Instalação instalacao){

Instalação.incrementaInstalacoes(); //Aumenta o número de instalações já criadas no zoo

instalacao.setNumeroInstalacao(Instalação.getInstalacoes()); //Dá um número á instalação, dependendo do número de instalações já criadas no zoo

adicionarInstalacao(instalacao); //Adiciona a instalação ao HashMap do zoo

adicionarSaldo(-(instalacao.getPrecoInstalacao())); //Retira ao saldo do zoo o preço de construção da instalação

criaOcorrencia("construcao",null,instalacao); //Cria a ocorrência relacionada á instalação (que neste caso é a sua construção)

}

/\*

Método usado para instalar um animal numa instalação

\*/

public void instalarAnimal(){

if(animaisFI.isEmpty()){ //Se não existir animais fora das instalações

System.out.println("Não existem animais para ser instalados");

}

else if (zoo.isEmpty()){ //Se não houver instalações no zoo

System.out.println("Não existem instalações para o animal ser instalado");

}

else{

Scanner scan = new Scanner(System.in);

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

System.out.println(key+" : "+value); //Imprime todas as instalações do zoo

}

int numero;

System.out.print("Escreva o numero da instalação a adicionar o animal:");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("O número da instalação tem de ser inteiro!");

System.out.print("Escreva o numero da instalação a adicionar o animal:");

scan.next();

}

numero = scan.nextInt();

try{

boolean existe = false;

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

if(numero == key){ //Se houver um número de instalação igual

existe = true;

}

}

if(!existe){

throw new InsNotFound();

}

for(int i = 0; i<animaisFI.size(); i++){ //Percorre os animais fora das instalações

System.out.println(i+") "+animaisFI.get(i)); //Imprime os animais fora das instalações

}

int animal;

System.out.print("Escoha o animal que quer adicionar:");

while(!scan.hasNextInt()){ //Verifica se o que o utilizador escreveu é um número inteiro

System.out.println("Para escolher uma opção tem de escrever um dos número inteiros apresentados no inicio de cada animal");

System.out.print("Escoha o animal que quer adicionar:");

scan.next();

}

animal = scan.nextInt();

if(animal<0 || animal>=animaisFI.size()){ //Verifica se o inteiro que o utilizador escreveu não corresponde a um animal na ArrayList dos animais fora das instalações

throw new AnimalNotFound();

}

adicionarAI(numero, animaisFI.get(animal)); //Adiciona o animal à ArrayList da instalação

System.out.println(animaisFI.get(animal).getNome()+" o "+animaisFI.get(animal).getClass()+" foi adicionado à instalção de número "+numero);

criaOcorrencia("colocacao",animaisFI.get(animal),zoo.get(numero)); //Cria a ocorrência relacionada a este animal e a essa instalação (que neste caso é a colocação do animal na instalação)

removerFI(animal); //Remove o animal da ArrayList dos animal fora das instalações

}

catch (InsNotFound | AnimalNotFound ex){

System.out.println(ex.getMessage());

}

}

}

/\*

Método usado para retirar ao saldo do zoo o valor para adquirir determinado animal

Recebe um animal, verifica qual é e retira o devido valor ao saldo do zoo (que é a sua atratividade base a dividir por 2)

\*/

public void comprarAnimal(Animal animal){

if(animal.getClass() == Boi.class){

adicionarSaldo(-Boi.getAtratividadeBase()/2);

}

else if (animal.getClass() == Cabra.class){

adicionarSaldo(-Cabra.getAtratividadeBase()/2);

}

else if (animal.getClass() == Cachorro.class){

adicionarSaldo(-Cachorro.getAtratividadeBase()/2);

}

else if (animal.getClass() == Cavalo.class){

adicionarSaldo(-Cavalo.getAtratividadeBase()/2);

}

else if (animal.getClass() == Cobra.class){

adicionarSaldo(-Cobra.getAtratividadeBase()/2);

}

else if (animal.getClass() == Coelho.class){

adicionarSaldo(-Coelho.getAtratividadeBase()/2);

}

else if (animal.getClass() == Crocodilo.class){

adicionarSaldo(-Crocodilo.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == Dragão.class){

adicionarSaldo(-Dragão.getAtratividadeBase()/2);

}

else if (animal.getClass() == Galo.class){

adicionarSaldo(-Galo.getAtratividadeBase()/2);

}

else if (animal.getClass() == Gorila.class){

adicionarSaldo(-Gorila.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == Leão.class){

adicionarSaldo(-Leão.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == Macaco.class){

adicionarSaldo(-Macaco.getAtratividadeBase()/2);

}

else if (animal.getClass() == Panda.class){

adicionarSaldo(-Panda.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == Porco.class){

adicionarSaldo(-Porco.getAtratividadeBase()/2);

}

else if (animal.getClass() == Rato.class){

adicionarSaldo(-Rato.getAtratividadeBase()/2);

}

else if (animal.getClass() == Tigre.class){

adicionarSaldo(-Tigre.getAtratividadeBase()/2);

}

else if (animal.getClass() == UrsoPardo.class){

adicionarSaldo(-UrsoPardo.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == UrsoPolar.class){

adicionarSaldo(-UrsoPolar.getATRATIVIDADEBASE()/2);

}

else if (animal.getClass() == Vaca.class){

adicionarSaldo(-Vaca.getATRATIVIDADEBASE()/2);

}

}

/\*

Método usado para retirar ao saldo do zoo o valor para adquirir um animal de um certo genoma

Recebe uma string que será o genoma do animal

\*/

public void comprarGenoma(String genoma){

genoma = genoma.toLowerCase();

switch (genoma) {

case "bovino" -> adicionarSaldo(-Bovino.getPreco());

case "canis" -> adicionarSaldo(-Canis.getPreco());

case "capra" -> adicionarSaldo(-Capra.getPreco());

case "crocodyloidea" -> adicionarSaldo(-Crocodyloidea.getPreco());

case "dragon" -> adicionarSaldo(-Dragon.getPreco());

case "equus" -> adicionarSaldo(-Equus.getPreco());

case "gallus" -> adicionarSaldo(-Gallus.getPreco());

case "ophidia" -> adicionarSaldo(-Ophidia.getPreco());

case "oryctolagus" -> adicionarSaldo(-Oryctolagus.getPreco());

case "panthera" -> adicionarSaldo(-Panthera.getPreco());

case "primata" -> adicionarSaldo(-Primata.getPreco());

case "roedor" -> adicionarSaldo(-Roedor.getPreco());

case "suíno" -> adicionarSaldo(-Suíno.getPreco());

case "urso" -> adicionarSaldo(-Urso.getPreco());

}

}

/\*

Método usado para criar um animal qualquer sem nome e que ainda não está no zoo

\*/

public Animal animalRandomizado(){

Random rand = new Random();

int a = rand.nextInt(19); //Utiliza um random que vai de 0 a 18

return switch (a) {

case 0 -> new Boi(null, false);

case 1 -> new Cabra(null, false);

case 2 -> new Cachorro(null, false);

case 3 -> new Cavalo(null, false);

case 4 -> new Cobra(null, false);

case 5 -> new Coelho(null, false);

case 6 -> new Crocodilo(null, false);

case 7 -> new Dragão(null, false);

case 8 -> new Galo(null, false);

case 9 -> new Gorila(null, false);

case 10 -> new Leão(null, false);

case 11 -> new Macaco(null, false);

case 12 -> new Panda(null, false);

case 13 -> new Porco(null, false);

case 14 -> new Rato(null, false);

case 15 -> new Tigre(null, false);

case 16 -> new UrsoPardo(null, false);

case 17 -> new UrsoPolar(null, false);

default -> new Vaca(null, false);

};

}

/\*

Método usado para criar um animal qualquer de um certo genoma

Recebe uma string que será o genoma do animal e dependendo dessa chama o método estático ("cria...") da classe do genoma

\*/

public Animal animalGenoma (String genoma){

genoma = genoma.toLowerCase();

return switch (genoma) {

case "bovino" -> Bovino.criaBovino();

case "canis" -> Canis.criaCanis();

case "capra" -> Capra.criaCapra();

case "crocodyloidea" -> Crocodyloidea.criaCrocodyloidea();

case "dragon" -> Dragon.criaDragon();

case "equus" -> Equus.criaEquus();

case "gallus" -> Gallus.criaGallus();

case "ophidia" -> Ophidia.criaOphidia();

case "oryctolagus" -> Oryctolagus.criaOryctolagus();

case "panthera" -> Panthera.criaPanthera();

case "primata" -> Primata.criaPrimata();

case "roedor" -> Roedor.criaRoedor();

case "suíno" -> Suíno.criaSuíno();

default -> Urso.criaUrso();

};

}

/\*

Método usado para duplicar a atratividade base de um animal no calêndário chinês dependendo do ano presente

\*/

public void anoChines(){

int ano = anoZoo%12; //Usando o resto da divisão por 12 conseguimos saber qual animal é suposto duplicar

switch (ano) {

case 6 -> {

Tigre.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Tigres duplicou");

}

case 7 -> {

Coelho.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Coelhos duplicou");

}

case 8 -> {

Dragão.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Dragões duplicou");

}

case 9 -> {

Cobra.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base das Cobras duplicou");

}

case 10 -> {

Cavalo.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Cavalos duplicou");

}

case 11 -> {

Cabra.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base das Cabras duplicou");

}

case 0 -> {

Macaco.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Macacos duplicou");

}

case 1 -> {

Galo.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Galos duplicou");

}

case 2 -> {

Cachorro.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Cachoros duplicou");

}

case 3 -> {

Porco.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Porcos duplicou");

}

case 4 -> {

Rato.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Ratos duplicou");

}

case 5 -> {

Boi.incrementaAtratividadeBase();

System.out.println("Como estamos no ano "+anoZoo+" a atratividade base dos Bois duplicou");

}

}

}

/\*

Método que serve para listar todos os animais do zoo (dentro e fora das instalações)

\*/

public void listarAnimais(){

if(animaisFI.isEmpty() && zoo.isEmpty()){ //Se não tem animais dentro nem fora das instalações

System.out.println("Não existem animais no zoo");

}

else{

System.out.println("Fora das instalações:");

if(animaisFI.isEmpty()){ //Se não tem animais fora das instalações

System.out.println("Não existem animais não instalados");

}

else{

for (Animal a : animaisFI){ //Percorre a ArrayList dos animais fora das instalações

System.out.println(a);

}

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

System.out.println("Dentro da instalação número "+key+":");

if(value.getAnimais().isEmpty()){ //Se não tem animais dentro da instalação

System.out.println("Não existem animais dentro desta instalação");

}

else{

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais dentro da instalação

System.out.println(a);

}

}

}

}

}

/\*

Método que serve para listar todos os animais do zoo (dentro e fora das instalações) de um certo genoma

\*/

public void listarAnimaisG(){

if(animaisFI.isEmpty() && zoo.isEmpty()){ //Se não tem animais dentro nem fora das instalações

System.out.println("Não existem animais no zoo");

}

else{

Scanner scan = new Scanner(System.in);

System.out.print("Quer listar os animais com qual genoma? Genoma: ");

String genoma;

while(!scan.hasNext()){ //Verifica se o que o utilizador escreveu é uma string

System.out.println("O nome do genoma tem de ser uma String!");

System.out.print("Genoma: ");

scan.next();

}

genoma = scan.next();

while (!genoma.matches("[a-zA-Z]+")){ //Verifica se o que o utilizador escreveu é uma string de letras

System.out.println("O nome do genoma tem de ser uma String de só letras!");

System.out.print("Genoma: ");

genoma = scan.next();

}

genoma = genoma.toLowerCase();

System.out.println("Fora das instalações:");

if(animaisFI.isEmpty()){ //Se não tem animais fora das instalações

System.out.println("Não existem animais fora das instalações");

}

else{

boolean existe= false; //Booleano que serve para sber se existe pelo menos um animal do genoma fora das instalações

for (Animal a : animaisFI){ //Percorre a ArrayList dos animais fora das instalações

switch (genoma) { //Verifica que genoma o utilizador escreveu

case "bovino" :

if(a.getClass() == Vaca.class || a.getClass() == Boi.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "canis" :

if(a.getClass() == Cachorro.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "capra" :

if(a.getClass() == Cabra.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "crocodyloidea" :

if(a.getClass() == Crocodilo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "dragon" :

if(a.getClass() == Dragão.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "equus" :

if(a.getClass() == Cavalo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "gallus" :

if(a.getClass() == Galo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "ophidia" :

if(a.getClass() == Cobra.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "oryctolagus" :

if(a.getClass() == Coelho.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "panthera" :

if(a.getClass() == Leão.class || a.getClass() == Tigre.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "primata" :

if(a.getClass() == Macaco.class || a.getClass() == Gorila.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "roedor" :

if(a.getClass() == Rato.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "suíno" :

if(a.getClass() == Porco.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "urso" :

if(a.getClass() == Panda.class || a.getClass() == UrsoPardo.class || a.getClass() == UrsoPolar.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

}

}

if(!existe){ //Se for false quer dizer que não existe um animal fora das instalações do genoma

System.out.println("Não existe nenhum animal com esse genoma fora das instalações");

}

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

System.out.println("Dentro da instalação número "+key+":");

if(value.getAnimais().isEmpty()){ //Se não tiver animais na ArrayList da instalação

System.out.println("Não existem animais dentro desta instalação");

}

else{

boolean existe= false; //Booleano que serve para saber se existe pelo menos um animal do genoma dentro da instalação

for (Animal a : value.getAnimais()){ //Percorre a ArrayList da instalação

switch (genoma) {

case "bovino" :

if(a.getClass() == Vaca.class || a.getClass() == Boi.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "canis" :

if(a.getClass() == Cachorro.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "capra" :

if(a.getClass() == Cabra.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "crocodyloidea" :

if(a.getClass() == Crocodilo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "dragon" :

if(a.getClass() == Dragão.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "equus" :

if(a.getClass() == Cavalo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "gallus" :

if(a.getClass() == Galo.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "ophidia" :

if(a.getClass() == Cobra.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "oryctolagus" :

if(a.getClass() == Coelho.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "panthera" :

if(a.getClass() == Leão.class || a.getClass() == Tigre.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "primata" :

if(a.getClass() == Macaco.class || a.getClass() == Gorila.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "roedor" :

if(a.getClass() == Rato.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "suíno" :

if(a.getClass() == Porco.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

case "urso" :

if(a.getClass() == Panda.class || a.getClass() == UrsoPardo.class || a.getClass() == UrsoPolar.class){ //Verifica se o animal é algum do genoma

System.out.println(a);

existe = true;

}

break;

}

}

if(!existe){ //Se for false quer dizer que não existe um animal na instalação do genoma

System.out.println("Não existe nenhuma animal com esse genoma dentro desta instalação");

}

}

}

}

}

/\*

Método que serve para listar todos os animais do zoo (dentro e fora das instalações) com uma certa mutação

\*/

public void listarAnimaisM(){

if(animaisFI.isEmpty() && zoo.isEmpty()){ //Se não tem animais dentro nem fora das instalações

System.out.println("Não existem animais no zoo");

}

else{

Scanner scan = new Scanner(System.in);

System.out.print("Quer listar os animais com qual mutação? Mutação: ");

String mutacao;

while(!scan.hasNext()){ //Verifica se o que o utilizador escreveu é uma string

System.out.println("O nome da mutação tem de ser uma String!");

System.out.print("Mutação: ");

scan.next();

}

mutacao = scan.next();

while (!mutacao.matches("[a-zA-Z]+")){ //Verifica se o que o utilizador escreveu é uma string de letras

System.out.println("O nome da mutação tem de ser uma String de só letras!");

System.out.print("Mutação: ");

mutacao = scan.next();

}

mutacao = mutacao.toLowerCase();

//Se não existir a mutação dada pelo utiliador

if (!mutacao.equals("heterocromia") && !mutacao.equals("albinismo") && !mutacao.equals("dragonismo")){

System.out.println("Não existe essa mutação");

}

else{

System.out.println("Fora das instalações:");

if(animaisFI.isEmpty()){ //Se não tem animais fora das instalações

System.out.println("Não existem animais não instalados");

}

else{

boolean existe = false; //Booleano usado para saber se existe pelo menos um animal fora das instalações com a mutação dada

for (Animal a : animaisFI){ //Percorre ArrayList dos animais fora das instalações

if(mutacao.equals("heterocromia") && a.isHETEROCROMIA()){ //Se a mutação é heterocromia e o animal a tem

System.out.println(a);

existe = true;

}

else if (mutacao.equals("albinismo") && a.isALBINISMO()){ //Se a mutação é albinismo e o animal a tem

System.out.println(a);

existe = true;

}

else if (mutacao.equals("dragonismo") && a.getClass()==Dragão.class){ //Se a mutação é dragonismo e o animal é um dragão

Dragão b = (Dragão) a;

if (b.isDRAGONISMO()){ //Se o dragão tem dragonismo

System.out.println(b);

existe = true;

}

}

}

if(!existe){ //Se for false quer dizer que não existe um animal fora das instalações com a mutação

System.out.println("Não existem animais com essa mutação fora das instalações");

}

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre HashMap de instalações

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

System.out.println("Dentro da instalação número "+key+":");

if(value.getAnimais().isEmpty()){ //Se não tiver animais na ArrayList da instalação

System.out.println("Não existem animais dentro desta instalação");

}

else{

boolean existe = false; //Booleano usado para saber se existe pelo menos um animal com a mutação na instalação

for (Animal a : value.getAnimais()){ //Percorre a ArrayList da instalação

if(mutacao.equals("heterocromia") && a.isHETEROCROMIA()){ //Se a mutação é heterocromia e o animal a tem

System.out.println(a);

existe = true;

}

else if (mutacao.equals("albinismo") && a.isALBINISMO()){ //Se a mutação é albinismo e o animal a tem

System.out.println(a);

existe = true;

}

else if (mutacao.equals("dragonismo") && a.getClass()==Dragão.class){ //Se a mutação é dragonismo e o animal é um dragão

Dragão b = (Dragão) a;

if (b.isDRAGONISMO()){ //Se o dragão tem dragonismo

System.out.println(b);

existe = true;

}

}

}

if(!existe){ //Se for false quer dizer que não existe um animal na instalação com a mutação

System.out.println("Não existe nenhuma animal com essa mutação nesta instalação");

}

}

}

}

}

}

/\*

Método usado para listar as instalações do zoo

\*/

public void listarInstalacoes(){

if(zoo.isEmpty()){ //Se não existe instalações no zoo

System.out.println("Não existem istalações no zoo");

}

else{

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações no zoo

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

System.out.println(key + " : " + value); //Imprime as instalações

}

}

}

/\*

Método usado para simplificar a criação das ocorrências do zoo

Recebe uma string que corresponde a uma ocorrência, e se tiverem relacionados com esta, um animal e uma instalação

\*/

public void criaOcorrencia (String ocorrencia, Animal animal, Instalação instalacao){

ocorrencia = ocorrencia.toLowerCase();

switch (ocorrencia) {

case "nascimento" -> adicionarOcorrencia(new Nascimento(animal,this));

case "obito" -> adicionarOcorrencia(new Óbito(animal,this));

case "adquirido" -> adicionarOcorrencia(new Adquirido(animal,this));

case "construcao" -> adicionarOcorrencia(new Construção(instalacao,this));

case "colocacao" -> adicionarOcorrencia(new Colocação(animal,instalacao,this));

case "fugiu" -> adicionarOcorrencia(new Fugiu(animal,this));

default -> System.out.println("Ocorreu um erro a criar a ocorrência");

}

}

/\*

Método usado para verificar e criar um novo animal (recém-nascido) no zoo

Só nasce um animal se já existir um do seu tipo instalado numa instalação no zoo

\*/

public void nascerAnimal(){

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

Random rand = new Random();

int probNacer=rand.nextInt(100); //Inteiro random de 0 a 99

//Verifica o tipo de animal

//Verifica se o número randomizado é menor ou igual ao apetite reprodutivo desse

if (a.getClass() == Tigre.class){

if(probNacer <= Tigre.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um tigre na instalação de número "+key);

Animal p =new Tigre(null, true); //Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass() == Leão.class){

if(probNacer<=Leão.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um leão na instalação de número "+key);

Animal p =new Leão(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Boi.class){

if(probNacer<=Boi.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um boi na instalação de número "+key);

Animal p =new Boi(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Cabra.class){

if(probNacer<=Cabra.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu uma cabra na instalação de número "+key);

Animal p =new Cabra(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Cachorro.class){

if(probNacer<=Cachorro.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um cachorro na instalação de número "+key);

Animal p =new Cachorro(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Cavalo.class){

if(probNacer<=Cavalo.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um cavalo na instalação de número "+key);

Animal p =new Cavalo(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Cobra.class){

if(probNacer<=Cobra.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um cobra na instalação de número "+key);

Animal p =new Cobra(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Coelho.class){

if(probNacer<=Coelho.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um coelho na instalação de número "+key);

Animal p =new Coelho(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Crocodilo.class){

if(probNacer<=Crocodilo.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um crocodilo na instalação de número "+key);

Animal p =new Crocodilo(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Dragão.class){

if(probNacer<=Cavalo.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um dragão na instalação de número "+key);

Animal p =new Dragão(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Galo.class){

if(probNacer<=Galo.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um galo na instalação de número "+key);

Animal p =new Galo(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Gorila.class){

if(probNacer<=Gorila.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um gorila na instalação de número "+key);

Animal p =new Gorila(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Macaco.class){

if(probNacer<=Macaco.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um macaco na instalação de número "+key);

Animal p =new Macaco(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Panda.class){

if(probNacer<=Panda.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um panda na instalação de número "+key);

Animal p =new Panda(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Porco.class){

if(probNacer<=Porco.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um porco na instalação de número "+key);

Animal p =new Porco(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Rato.class){

if(probNacer<=Rato.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um rato na instalação de número "+key);

Animal p =new Rato(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==UrsoPolar.class){

if(probNacer<=UrsoPolar.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um urso polar na instalação de número "+key);

Animal p =new UrsoPolar(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==Vaca.class){

if(probNacer<=Vaca.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu uma vaca na instalação de número "+key);

Animal p =new Vaca(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

else if (a.getClass()==UrsoPardo.class){

if(probNacer<=UrsoPardo.getAPETITEREPRODUTIVO()){

System.out.println("Nasceu um urso pardo na instalação de número "+key);

Animal p =new UrsoPardo(null, true);//Cria um novo animal já instalado no zoo (isto é , que ja tem id)

constroiAnimal(p,"nascimento"); //Constroi esse animal, sendo que já tem id, e cria a ocorrência

p.setIdade(0); //Muda a sua idade para 0

}

}

}

}

}

/\*

Método usado para aumentar a idade dos animais no zoo

\*/

public void aumentarIdade(){

for (Animal a : animaisFI){ //Percorre os animais fora das instalações

a.incrementaIdade();

a.incrementaBonus(-50);

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HasMap da sinstalações no zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais na instalação

a.incrementaIdade();

a.incrementaBonus(-50);

}

}

}

/\*

Método usado para verificar e "matar" os animais que já passaram da sua esperança de vida média

\*/

public void morrerAnimal(){

for(int i = 0; i<animaisFI.size(); i++){ //Percorre a ArrayList dos animais fora do zoo

//Verifica o tipo de animal

//Verifica se a sua idade já passou da sua esperança média de vida

if(animaisFI.get(i).getClass()==Tigre.class){

if(Tigre.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o tigre chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Leão.class){

if(Leão.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o leão chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Boi.class){

if(Boi.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o boi chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Cabra.class){

if(Cabra.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu a cabra chamada "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Cachorro.class){

if(Cachorro.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o cachorro chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Cavalo.class){

if(Cavalo.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o cavalo chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Cobra.class){

if(Cobra.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu a cobra chamada "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Coelho.class){

if(Coelho.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o coelho chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Crocodilo.class){

if(Crocodilo.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o crocodilo chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Dragão.class){

if(Dragão.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o dragão chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Galo.class){

if(Galo.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o galo chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Gorila.class){

if(Gorila.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o gorila chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Macaco.class){

if(Macaco.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o macaco chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Panda.class){

if(Panda.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o panda chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Porco.class){

if(Porco.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o porco chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Rato.class){

if(Rato.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o rato chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==UrsoPolar.class){

if(UrsoPolar.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o urso polar chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==Vaca.class){

if(Vaca.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu a vaca chamada "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

else if (animaisFI.get(i).getClass()==UrsoPardo.class){

if(UrsoPardo.getESPERANCAVIDA()<animaisFI.get(i).getIdade()){

System.out.println("Morreu o urso pardo chamado "+animaisFI.get(i).getNome()+" de id "+animaisFI.get(i).getId());

criaOcorrencia("obito", animaisFI.get(i),null); //Cria ocorrência de óbito

adicionarObitos(animaisFI.get(i)); //Adiciona o animal a ArrayList do obituário

removerFI(i); //Remove o animal da ArrayList dos animais fora do zoo

}

}

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (int i = 0; i<value.getAnimais().size();i++){ //Percorre a ArrayList dos animais na instalação

//Verifica o tipo de animal

//Verifica se a sua idade já passou da sua esperança média de vida

if(value.getAnimais().get(i).getClass()==Tigre.class){

if(Tigre.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o tigre chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Leão.class){

if(Leão.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o leão chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Boi.class){

if(Boi.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o boi chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Cabra.class){

if(Cabra.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu a cabra chamada "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Cachorro.class){

if(Cachorro.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o cachorro chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Cavalo.class){

if(Cavalo.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o cavalo chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Cobra.class){

if(Cobra.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu a cobra chamada "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Coelho.class){

if(Coelho.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o coelho chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Crocodilo.class){

if(Crocodilo.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o crocodilo chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Dragão.class){

if(Dragão.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o dragão chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Galo.class){

if(Galo.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o galo chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Gorila.class){

if(Gorila.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o gorila chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Macaco.class){

if(Macaco.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o macaco chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Panda.class){

if(Panda.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o panda chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Porco.class){

if(Porco.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o porco chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Rato.class){

if(Rato.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o rato chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==UrsoPolar.class){

if(UrsoPolar.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o urso polar chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==Vaca.class){

if(Vaca.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu a vaca chamada "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

else if (value.getAnimais().get(i).getClass()==UrsoPardo.class){

if(UrsoPardo.getESPERANCAVIDA()<value.getAnimais().get(i).getIdade()){

System.out.println("Morreu o urso pardo chamado "+value.getAnimais().get(i).getNome()+" de id "+value.getAnimais().get(i).getId());

criaOcorrencia("obito", value.getAnimais().get(i),null); //Cria ocorrência de óbito

adicionarObitos(value.getAnimais().get(i)); //Adiciona o animal a ArrayList do obituário

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList dos animais na istalação

}

}

}

}

}

/\*

Método usado para retirar ao saldo do zoo as despesas de cada animal fora das instalações do zoo

Estes só dão despesa

\*/

public void animaisSaldo(){

for (Animal a : animaisFI){ //Percorre a ArrayList dos animais fora das instalações

//Verifica o tipo do animal

if(a.getClass() == Boi.class){

Boi b = (Boi) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Cabra.class){

Cabra b = (Cabra) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Cachorro.class){

Cachorro b = (Cachorro) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Cavalo.class){

Cavalo b = (Cavalo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Cobra.class){

Cobra b = (Cobra) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Coelho.class){

Coelho b = (Coelho) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Crocodilo.class){

Crocodilo b = (Crocodilo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Dragão.class){

Dragão b = (Dragão) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Galo.class){

Galo b = (Galo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Gorila.class){

Gorila b = (Gorila) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Leão.class){

Leão b = (Leão) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Macaco.class){

Macaco b = (Macaco) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Panda.class){

Panda b = (Panda) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Porco.class){

Porco b = (Porco) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Rato.class){

Rato b = (Rato) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Tigre.class){

Tigre b = (Tigre) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == UrsoPardo.class){

UrsoPardo b = (UrsoPardo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == UrsoPolar.class){

UrsoPolar b = (UrsoPolar) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

else if (a.getClass() == Vaca.class){

Vaca b = (Vaca) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

}

}

}

/\*

Método usado para retirar e adicionar ao saldo do zoo as despesas e os ganhos de cada animal dentro das instalações do zoo

Estes dão despesas e ganhos

\*/

public void animaisISaldo (){

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HasMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

for (Animal a : value.getAnimais()){ //Percorre a ArrayList dos animais da instalação

//Verifica o tipo do animal

if(a.getClass() == Boi.class){

Boi b = (Boi) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Cabra.class){

Cabra b = (Cabra) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Cachorro.class){

Cachorro b = (Cachorro) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Cavalo.class){

Cavalo b = (Cavalo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Cobra.class){

Cobra b = (Cobra) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Coelho.class){

Coelho b = (Coelho) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Crocodilo.class){

Crocodilo b = (Crocodilo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Dragão.class){

Dragão b = (Dragão) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Galo.class){

Galo b = (Galo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Gorila.class){

Gorila b = (Gorila) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Leão.class){

Leão b = (Leão) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Macaco.class){

Macaco b = (Macaco) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Panda.class){

Panda b = (Panda) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Porco.class){

Porco b = (Porco) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Rato.class){

Rato b = (Rato) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Tigre.class){

Tigre b = (Tigre) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == UrsoPardo.class){

UrsoPardo b = (UrsoPardo) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == UrsoPolar.class){

UrsoPolar b = (UrsoPolar) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

else if (a.getClass() == Vaca.class){

Vaca b = (Vaca) a;

b.comerAnual(this); //Retira o valor da comida ao saldo do zoo

b.cuidarAnual(this); //Retira o valor dos cuidados ao saldo do zoo

saldo+=Boi.getAtratividadeBase()+b.getBonus(); //Adiciona a sua atratividade base e seu bonus

}

}

}

}

/\*

Método usado para retirar ao saldo do zoo as despesas das instalações do zoo

Estas só dão despesas

\*/

public void instalacoesSaldo (){

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HashMap das instalações do zoo

Instalação value = ((Instalação)instalacao.getValue());

value.manutencaoInstalacaoAno(this); //Retira o valor da manutenção da instalação ao saldo do zoo

}

}

/\*

Método usado para verificar se o zoo está em prejuizo, e se estiver invoca o método "jumangi" onde podem fugir animais

\*/

public void prejuizo(){

if (saldo < 0){

System.out.println("Saldo negativo, logo pode haver prejuisos");

jumanji();

}

else{

System.out.println("Saldo Positivo, logo não há prejuisos");

}

}

/\*

Método usado para juntar e ordenar todos os métodos usados num período contabilistico

\*/

public void periodoContabilistico (){

animaisSaldo(); //Retira ao saldo do zoo as despesas dos animais fora das instalações

animaisISaldo(); //Retira e adiciona ao saldo so zoo as despesas e os ganhos dos animais dentro das instalações

instalacoesSaldo(); //Retira ao saldo do zoo as despesas das instalaçoes do zoo

aumentarIdade(); //Aumenta a idade de todos os animais do zoo

aumentarAno(); //Aumenta o ano do zoo

nascerAnimal(); //Cria animais que nascem

morrerAnimal(); //Mata animais que morrem

System.out.println("Passando do ano "+(anoZoo-1)+" para o ano "+anoZoo+" o saldo constabilístico resume-se em: "+saldo);

prejuizo(); //Dependendo do saldo executa o método "jumangi"

}

/\*

Método usado para verificar a probabilidade do animal fugir e fazer ele fugir do zoo

\*/

public void jumanji(){

for(int i = 0; i<animaisFI.size();i++){ //Percorre a ArrayList dos animais fora das instalações

Random rand = new Random();

int r = rand.nextInt(100); //Cria um inteiro randomizado de 0 a 99

if(r > 30){ //Se esse random for maior que 30 (se foge do zoo)

if(r > 75){ //Se esse random for maior que 75 (foge do zoo e nunca volta)

System.out.println("O "+animaisFI.get(i).getClass()+" chamado "+animaisFI.get(i).getNome()+" fugiu do zoo");

criaOcorrencia("fugiu",animaisFI.get(i),null); //Cria a ocorrência do animal fugir do zoo

animaisFI.remove(animaisFI.get(i)); //Remove o animal da ArrayList dos animais fora das instalações

}

else{ //Se não (foge do zoo mas é apanhado)

System.out.println("O "+animaisFI.get(i).getClass()+" chamado "+animaisFI.get(i).getNome()+" fugiu do zoo mas por sorte foi apanhado.");

animaisFI.add(animaisFI.get(i)); //Adiciona o animal na ArrayList dos animais fora das instalações

animaisFI.remove(animaisFI.get(i)); //Retira o animal da ArrayList dos animais fora das instalações

}

}

}

for (Map.Entry instalacao : zoo.entrySet()){ //Percorre o HasMap das instalações do zoo

int key = (int)instalacao.getKey();

Instalação value = ((Instalação)instalacao.getValue());

for (int i = 0; i<value.getAnimais().size();i++){ //Percorre a ArrayList dos animais na instalação

Random rand = new Random();

int r = rand.nextInt(100); //Cria um inteiro randomizado de 0 a 99

if(r > 30){ //Se esse random for maior que 30 (se foge do zoo)

if(r > 75){ //Se esse random for maior que 75 (foge do zoo e nunca volta)

System.out.println("O "+value.getAnimais().get(i).getClass()+" chamado "+value.getAnimais().get(i).getNome()+" fugiu da instalação número "+key);

criaOcorrencia("fugiu",animaisFI.get(i),null); //Cria a ocorrência do animal fugir do zoo

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList de animais da instalação

value.setAnimaisInstalados(value.getAnimaisInstalados()-1); //Diminui o número de animais na instalação

}

else{ //Se não (foge do zoo mas é apanhado)

System.out.println("O "+value.getAnimais().get(i).getClass()+" chamado "+value.getAnimais().get(i).getNome()+" fugiu da instalação número "+key+" mas por sorte foi apanhado.");

animaisFI.add(value.getAnimais().get(i)); //Adiciona o animal na ArrayList dos animais fora das instalações

value.getAnimais().remove(value.getAnimais().get(i)); //Remove o animal da ArrayList de animais da instalação

value.setAnimaisInstalados(value.getAnimaisInstalados()-1); //Diminui o número de animais na instalação

}

}

}

}

}

/\*

Método usado para juntar todo o zoo de modo a imprimir-lo todo

\*/

public void retratoF(){

System.out.println("Retrato de família animal:");

System.out.println("\nAnimais do zoo:");

listarAnimais(); //Lista todos os animais do zoo

System.out.println("Instalações do zoo:");

listarInstalacoes(); //Lista todas as instalações do zoo

System.out.println("Obituário :\n"+obituario);

System.out.println("Histórico :\n"+historico);

}

}

# Anexo do Diagrama UML

Como o diagrama é muito grande vamos também enviar na pasta do projeto a imagem e o link para abrir no site diagrams.net

