

Programação com Objectos/Projecto de Programação com Objectos/Enunciado do Projecto de 2024-2025



From Wiki**3

< Programação com Objectos | Projecto de Programação com Objectos

AVISOS - Avaliação em Época Normal	[Collapse]
<p>Esclarecimento de dúvidas:</p> <ul style="list-style-type: none">Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço oficial da disciplina [1].Não utilizar fontes de informação não oficialmente associadas ao corpo docente (podem colocar em causa a aprovação à disciplina).Não são aceites justificações para violações destes conselhos: quaisquer consequências nefastas são da responsabilidade do aluno.	
<p>Requisitos para desenvolvimento, material de apoio e actualizações do enunciado (ver informação completa em Projecto de Programação com Objectos):</p> <ul style="list-style-type: none">O material de apoio é de uso obrigatório e não pode ser alterado.Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de desenvolvimento.	
<p>Processo de avaliação (ver informação completa em Avaliação do Projecto):</p> <ul style="list-style-type: none">Datas: 2024/09/27 12:00 (inicial); 2024/10/11 12:00 (intercalar); 2024/10/25 12:00 (final); 2024/10/25 (early bird) 2024/10/28 (normal) (teste prático).Todas as entregas são cruciais para o bom desenvolvimento do projecto, sendo obrigatórias: a não realização de uma entrega implica a exclusão da avaliação do projecto e, por consequência, da avaliação da disciplina.Verificar atempadamente (até 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação, incluindo a capacidade de acesso ao repositório.Apenas se consideram para avaliação os projectos existentes no repositório oficial. Apenas se considera para avaliação o ramo 'master'.Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório ou de outros materiais, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.A avaliação do projecto pressupõe o compromisso de honra de que o trabalho correspondente foi realizado pelos alunos correspondentes ao grupo de avaliação.Fraudes na execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação.	
Material de Uso Obrigatório	[Collapse]
<p>As bibliotecas po-uilib e o conteúdo inicial do repositório GIT são de uso obrigatório:</p> <ul style="list-style-type: none">po-uilib (classes de base) po-uilib-202408310000.tar.bz2 (não pode ser alterada) - javadochva-core (classes do "core") (via GIT) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados)hva-app (classes de interação) (via GIT) (deve ser completada -- os nomes das classes fornecidas não podem ser alterados) <p>A máquina virtual, fornecida para desenvolvimento do projecto, já contém todo o material de apoio.</p>	
<p>Uso Obrigatório: Repositório GIT</p> <p>Apenas se consideram para avaliação os projectos existentes no repositório GIT oficial.</p> <p>Trabalhos não presentes no repositório no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega). Não são admitidas justificações para atrasos em sincronizações do repositório. A indisponibilidade temporária do repositório, desde que inferior a 24 horas, não justifica atrasos na submissão de um trabalho.</p>	

Contents

- 1 Propriedades e funcionalidade
 - 1.1 Espécies
 - 1.2 Animais
 - 1.2.1 Satisfação dos Animais
 - 1.3 Habitats
 - 1.4 Árvores
 - 1.5 Funcionários
 - 1.5.1 Satisfação dos Veterinários
 - 1.5.2 Satisfação dos Tratadores
 - 1.6 Vacinas
 - 1.6.1 Problemas associados à vacinação
- 2 Funcionalidade da aplicação
- 3 Requisitos de desenho
- 4 Interacção com o utilizador
 - 4.1 Menu Principal
 - 4.1.1 Salvaguarda do estado actual da aplicação
 - 4.2 Avançar Estação do Ano
 - 4.3 Ver Estado de Satisfação Global
 - 4.4 Menu de Gestão de Animais
 - 4.4.1 Visualizar todos os animais
 - 4.4.2 Registar um novo animal
 - 4.4.3 Transferir um animal para um habitat
 - 4.4.4 Calcular a satisfação de um animal
 - 4.5 Menu de Gestão de Funcionários
 - 4.5.1 Visualizar todos os funcionários
 - 4.5.2 Registar um novo funcionário
 - 4.5.3 Atribuir uma nova responsabilidade a um funcionário
 - 4.5.4 Retirar uma responsabilidade a um funcionário
 - 4.5.5 Calcular a satisfação de um funcionário
 - 4.6 Menu de Gestão de Habitats
 - 4.6.1 Visualizar todos os habitats
 - 4.6.2 Registar um novo habitat
 - 4.6.3 Alterar a área de um habitat
 - 4.6.4 Alterar influência de um habitat sobre uma espécie
 - 4.6.5 Plantar uma nova árvore num habitat
 - 4.6.6 Visualizar todas as árvores de um habitat
 - 4.7 Menu de Gestão de Vacinas
 - 4.7.1 Visualizar todas as vacinas
 - 4.7.2 Registar uma nova vacina
 - 4.7.3 Vacinar um animal
 - 4.7.4 Listar o histórico de vacinações
 - 4.8 Menu de Consultas
 - 4.8.1 Consultar animais de um habitat
 - 4.8.2 Consultar vacinas de um animal
 - 4.8.3 Consultar actos médicos de um veterinário
 - 4.8.4 Consultar vacinas dadas com incúria
- 5 Inicialização por Ficheiro de Dados Textuais

O objectivo do projecto é desenvolver uma aplicação de gestão de um hotel veterinário que dispõe de tratadores, veterinários e habitats com árvores.

Propriedades e funcionalidade

Os animais de várias espécies são mantidos por tratadores que são responsáveis por distribuir a comida e pela limpeza dos habitats, e por veterinários, que têm a responsabilidade de zelar pela saúde dos animais. Quando um animal recebe uma vacina não apropriada à sua espécie, por causa de um erro veterinário, por exemplo, fica com a saúde afectada até ao fim da sua vida. Os habitats têm árvores. É possível calcular o grau de satisfação dos animais e dos funcionários.

Em todos os campos identificadores ou nomes, excepto se indicado o contrário, as diferenças entre maiúsculas e minúsculas são irrelevantes.

Espécies

As espécies são identificadas por uma chave única (cadeia de caracteres arbitrária definida na altura da criação).

Cada espécie tem um nome (cadeia de caracteres única, i.e., não existem duas espécies distintas com o mesmo nome) e um registo dos animais da espécie.

Animais

Os animais são identificados por uma chave única (cadeia de caracteres arbitrária definida na altura da criação).

Cada animal tem um nome (cadeia de caracteres não única) e informação acerca da sua espécie e do estado de saúde.

É possível calcular o nível de satisfação de um animal. Este valor depende de vários factores (ver a seguir).

Satisfação dos Animais

A satisfação dos animais é afectada positivamente quando existem mais animais da mesma espécie no mesmo habitat e afectada negativamente quando existem animais de outras espécies. O número total de animais no habitat e o próprio habitat também influenciam a satisfação dos animais que lá vivem.

A satisfação de um animal **a** no seu habitat **h** é calculada pela fórmula:

$$\text{satisfação(a)} = 20 + 3 * \text{iguais(a, h)} - 2 * \text{diferentes(a, h)} + \text{área(h)} / \text{população(h)} + \text{adequação(a, h)}$$

Nesta equação, **iguais** conta o número de animais da mesma espécie no habitat (sem contar com **a**), **diferentes** é o número de animais de espécies diferentes da de **a** no habitat, **área** é a área do habitat e **população** é a população total do habitat, **adequação** é **20** se a influência do habitat for positiva, **-20** se for negativa e **0** se a influência for neutra (condição por omissão).

Habitats

Os habitats são identificados por uma chave única (cadeia de caracteres arbitrária definida na altura da criação).

Cada habitat tem um nome (cadeia de caracteres; não única) e informação sobre a área (número inteiro). Os habitats estão associados a zero ou mais árvores e a zero ou mais animais. Estão ainda associados às espécies cuja satisfação influenciam.

Os habitats podem ser mais (ou menos) adequados para determinadas espécies. A adequação tem impacto no grau de satisfação dos animais. Por omissão, um habitat tem um impacto neutro para cada espécie.

Árvores

As árvores são identificadas por uma chave única (cadeia de caracteres arbitrária definida na altura da criação). Cada árvore tem ainda nome (cadeia de caracteres; não única) e idade em anos (número inteiro). As árvores podem ser de folha caduca ou perene e são caracterizadas pela dificuldade base de limpeza (número inteiro) que induzem no habitat onde estão implantadas (definido

no momento da introdução da árvore no habitat).

A vida das árvores segue o ciclo definido pelas estações do ano. Assim, por exemplo, as árvores de folha caduca perdem as folhas principalmente durante o Outono, ficando sem folhas no Inverno. As árvores de folha perene perdem algumas folhas durante todas as estações, mas mais durante o Inverno.

O esforço total de limpeza de uma árvore é assim determinado pelo produto de três factores: o primeiro corresponde à dificuldade base de limpeza da árvore; o segundo depende da estação do ano e do tipo de árvore, tal como indicado na tabela seguinte (função **esforço_sazonal**); e o terceiro corresponde a um factor que aumenta com a idade da árvore (ver abaixo).

Estação »	Inverno	Primavera	Verão	Outono
Folha caduca »	0	1	2	5
Folha perene »	2	1	1	1

A parte do esforço total de limpeza que é proporcional à idade da árvore cresce logaritmicamente (i.e., não muito rapidamente): **log(idade + 1)** (logaritmo natural).

Combinando os vários elementos, obtém-se a fórmula para o esforço total de limpeza:

▪ **esforço_limpeza(a) = dificuldade_limpeza(a) * esforço_sazonal(a) * log(idade(a) + 1)**

Quando uma árvore é criada, fica na estação do ano em que a aplicação estiver no momento da criação. A estação inicial da aplicação é a **Primavera**.

As árvores envelhecem com as estações, i.e., a idade aumenta em 1 unidade (1 ano) a cada 4 estações (note-se que o incremento não é necessariamente simultâneo em todas as árvores).

Funcionários

Os funcionários são identificados por uma chave única (cadeia de caracteres arbitrária definida na altura da criação). Os funcionários têm um nome (cadeia de caracteres não única).

Existem dois tipos de funcionários: tratadores e veterinários. Cada tratador está associado aos habitats da sua responsabilidade (zero ou mais). Cada veterinário está associado às espécies que sabe vacinar (zero ou mais).

Satisfação dos Veterinários

A satisfação dos veterinários é afectada positivamente pela existência de outros veterinários com a mesma responsabilidade (**n_veterinários(e)**) e negativamente pelo número de animais que estão sob a sua responsabilidade. Apenas se contabilizam as espécies que o veterinário pode tratar (**espécies(v)**).

A satisfação para um veterinário **v** é calculada pela fórmula:

▪ **satisfação(v) = 20 - trabalho(v)**
▪ **trabalho(v) = Σ (população(e) / n_veterinários(e)), e ∈ espécies(v)**

Satisfação dos Tratadores

A satisfação dos tratadores é afectada negativamente pelo trabalho que lhes é atribuído. Apenas se contabilizam os habitats sob responsabilidade do tratador (**habitats(t)**). O trabalho é também influenciado pelas árvores existentes no habitat (**árvores(h)**) (ver também secção sobre habitats e árvores).

A satisfação para um tratador **t** é calculada pela fórmula:

▪ **satisfação(t) = 300 - trabalho(t)**
▪ **trabalho(t) = Σ (trabalho_no_habitat(h) / número_de_tratadores_podem_cuidar_habitat(h)), h ∈ habitats(t)**
▪ **trabalho_no_habitat(h) = área(h) + 3 * população(h) + Σ esforço_limpeza(a), a ∈ árvores(h)**

Vacinas

As vacinas são identificadas por uma chave única (cadeia de caracteres arbitrária definida na altura da criação). Cada vacina tem um nome (cadeia de caracteres; não única) e está associada às espécies a que pode ser aplicada. Tem ainda o registo de aplicações a cada animal pelos veterinários, por ordem de vacinação.

Problemas associados à vacinação

O estado de saúde de um animal é o histórico de eventos (inicialmente vazio), relacionados com a vacinação do animal, a partir da situação inicial, entre os quais estão os danos causados por más vacinações. Se a espécie do animal está incluída nas espécies para as quais uma vacina é apropriada, então o valor do dano causado pela vacina é 0 (zero):

- **dano(v, a) = 0**
- Caso contrário, os danos causados pela má administração de uma vacina **v** a um animal **a** dependem do máximo do número de letras distintas entre o nome da espécie de **a** (**espécie(a)**) e os nomes das espécies a que se destinava **v** (**espécies(v)**).
- **dano(v, a) = MAX (tamanho_nomes(espécie(a), e) - caracteres_comuns(espécie(a), e)), e ∈ espécies(v)**
- **tamanho_nomes(espécie1, espécie2) = max(tamanho(espécie1), tamanho(espécie2))**

Por exemplo, quando uma ave recebe uma vacina destinada apenas a um mamífero, o dano resultante da má aplicação da vacina é igual a 6.

A seguinte tabela descreve os resultados associados aos potenciais danos causados por vacinações:

Dano	Termo a adicionar na apresentação do estado de saúde
0 (mesma espécie)	NORMAL
0 (espécies diferentes)	CONFUSÃO
1 a 4	ACIDENTE
5 ou mais	ERRO

Por exemplo, um animal que tenha recebido 2 vacinas com um dano entre 1 e 4 (ACIDENTE), três vacinas com um dano igual ou superior a 5 (ERRO) e uma vacina correctamente aplicada, apresenta o seguinte historial de saúde (usa-se a vírgula como separador): ACIDENTE,ACIDENTE,ERRO,ERRO,ERRO,NORMAL (i.e., concatenação ordenada por ordem de ocorrência).

Funcionalidade da aplicação


A aplicação permite:

- Gerir e operar sobre toda a informação relativa aos conceitos acima, incluindo o cálculo da satisfação;
- Preservar persistentemente o seu estado via serialização Java (não é possível manter várias versões do estado da aplicação em simultâneo);
- Efectuar pesquisas sujeitas a vários critérios e sobre as diferentes entidades geridas.
- Carregamento no início da aplicação de uma base de dados textual com conceitos pré-definidos.

As operações sobre animais, funcionários, habitats, árvores, vacinas e vacinações devem cobrir, no mínimo, os requisitos para o funcionamento dos menus descritos abaixo. As estruturas e correspondentes funcionalidades devem ter ainda em consideração os requisitos descritos a seguir.



Note-se que não é necessário nem desejável implementar de raiz a aplicação: já existem classes que representam e definem a interface geral da funcionalidade do *core* da aplicação, tal como é visível pelos comandos da aplicação.

 A interface geral do *core* já está parcialmente implementada na classe **`hva.HotelManager`** e outras fornecidas (cujos nomes devem ser mantidos), devendo ser adaptadas onde necessário. É ainda necessário criar e implementar as restantes classes que suportam a operação da aplicação.

Requisitos de desenho

- A aplicação a desenvolver deve seguir o princípio de desenho aberto-fechado por forma a aumentar a extensibilidade do seu código. Por exemplo, deve ser possível adicionar novos tipos de funcionários com um impacto mínimo no código já existente do domínio.
- Aplicação de um padrão de desenho para melhorar a legibilidade do código relacionado com o conceito árvore e permitir ao mesmo tempo que se possam adicionar novos eventos relacionados com as estações sem que isso implique alterar o código relacionado com árvores. O padrão deve ainda poder suportar novas funcionalidades dependentes da estação actual (por exemplo, qual a cor das folhas de uma árvore, que vai depender do tipo de árvore e da estação actual) sem comprometer a legibilidade da solução.
- Aplicação de um padrão de desenho que permita novas políticas de cálculo da satisfação dos funcionários sem impacto no código existente. A solução concretizada deve permitir alterar o cálculo da satisfação de um funcionário em tempo de execução. Quando é criado um funcionário, o cálculo da satisfação segue a fórmula descrita neste enunciado.
- Deve ser possível introduzir novos métodos de pesquisa com um impacto mínimo na implementação desenvolvida.
- Apesar de haver na especificação actual apenas um hotel, o código não deve assumir que não podem existir mais instâncias.

Interacção com o utilizador

- Descreve-se nesta secção a **funcionalidade máxima** da interface com o utilizador. Em geral, os comandos pedem toda a informação antes de procederem à sua validação (excepto onde indicado). Todos os menus têm automaticamente a opção **Sair** (fecha o menu).
- As operações de pedido e apresentação de informação ao utilizador **devem** realizar-se através dos objectos *form* e *display*, respectivamente, presentes em cada comando. As mensagens são produzidas pelos métodos das bibliotecas de suporte (**`po-uilib`** e **`hva-app`**). As mensagens não podem ser usadas no núcleo da aplicação (**`hva-core`**). Além disso, não podem ser definidas novas. Potenciais omissões devem ser esclarecidas antes de qualquer implementação.
- A apresentação de listas (e.g., animais, funcionários, vacinas, etc.) faz-se por ordem crescente da respectiva chave, excepto em caso de indicação contrária. A ordem das chaves alfanuméricas deve ser lexicográfica (UTF-8), não havendo distinção entre maiúsculas e minúsculas.
- De um modo geral, sempre que no contexto de uma operação com o utilizador aconteça alguma excepção, então a operação não deve ter qualquer efeito no estado da aplicação, excepto em caso de indicação contrária na operação em causa. As excepções estão na package **`hva.app.exceptions`**, excepto em caso de indicação contrária.
- As excepções usadas na interacção (subclasses de **`pt.tecnico.uilib.menus.CommandException`**), excepto em caso de indicação contrária, são lançadas pelos comandos (subclasses de **`pt.tecnico.uilib.menus.Command`**) e tratadas pelos menus (instâncias de subclasses de **`pt.tecnico.uilib.menus.Menu`**). Outras excepções não devem substituir as fornecidas nos casos descritos.
- Nos pedidos e usos dos vários identificadores, podem ocorrer as seguintes excepções, caso o identificador indicado não corresponda a um objecto conhecido (excepto no processo de registo ou em caso de indicação contrária). Note-se que estas excepções não são utilizáveis no núcleo da aplicação.

Tipo de dados	Classe	Pedido a apresentar	Excepção a lançar se desconhecido
Animal	<code>Animal</code>	<code>hva.app.animal.Prompt.animalKey()</code>	<code>hva.app.exceptions.UnknownAnimalKeyException</code>
Espécie	<code>Species</code>	<code>hva.app.animal.Prompt.speciesKey()</code>	<code>hva.app.exceptions.UnknownSpeciesKeyException</code>

Funcionário	Employee	<code>hva.app.employee.Prompt.employeeKey()</code>	<code>hva.app.exceptions.UnknownEmployeeKeyException</code>
Habitat	Habitat	<code>hva.app.habitat.Prompt.habitatKey()</code>	<code>hva.app.exceptions.UnknownHabitatKeyException</code>
Árvore	Tree	<code>hva.app.habitat.Prompt.treeKey()</code>	<code>hva.app.exceptions.UnknownTreeKeyException</code>
Vacina	Vaccine	<code>hva.app.vaccine.Prompt.vaccineKey()</code>	<code>hva.app.exceptions.UnknownVaccineKeyException</code>

Alguns casos particulares podem usar pedidos específicos não apresentados nesta tabela.



Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcial ou completamente implementados nas *packages* **`hva.app`**, **`hva.app.main`**, **`hva.app.edit`**, **`hva.app.search`**. Estas classes são de uso obrigatório e estão disponíveis no GIT (módulo **`hva-app`**).

Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação, realizar algumas operações globais e abrir submenus. A lista completa é a seguinte: Criar, Abrir, Guardar, Avançar Estação do Ano, Ver Estado de Satisfação, Gestão de Animais, Gestão de Funcionários, Gestão de Vacinas, Gestão de Habitats e Consultas. As secções abaixo descrevem pormenorizadamente as acções associadas a estas opções.



As etiquetas das opções deste menu estão definidas em **`hva.app.main.Label`**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **`hva.app.main.Prompt`** e **`hva.app.main.Message`**.



Estes comandos já estão implementados nas classes da *package* **`hva.app.main`** (disponível no GIT), respectivamente: **`DoNewFile`**, **`DoOpenFile`**, **`DoSaveFile`**, **`DoAdvanceSeason`**, **`DoShowGlobalSatisfaction`**, **`DoOpenAnimalsMenu`**, **`DoOpenEmployeesMenu`**, **`DoOpenHabitatsMenu`**, **`DoOpenVaccinesMenu`**, **`DoOpenLookupsMenu`**.

Salvaguarda do estado actual da aplicação

Inicialmente, a aplicação não tem qualquer informação, excepto quando usada a propriedade **`import`** (ver abaixo) para pré-carregar dados iniciais. O conteúdo da aplicação (toda a informação actualmente em memória) pode ser guardado para posterior recuperação (via serialização Java: **`java.io.Serializable`**). Na leitura e escrita do estado da aplicação, devem ser tratadas as excepções associadas. A funcionalidade é a seguinte:

- **Criar** -- Cria uma nova aplicação vazia: a aplicação não fica associada a nenhum ficheiro (é anónima).
- **Abrir** -- Carrega os dados de uma sessão anterior a partir de um ficheiro previamente guardado (ficando este ficheiro associado à aplicação, para futuras operações de salvaguarda). Pede-se o nome do ficheiro a abrir (**`Prompt.openFile()`**). Caso ocorra um problema na abertura ou processamento do ficheiro, deve ser lançada a excepção **`FileOpenFailedException`**. A execução bem-sucedida desta opção substitui toda a informação da aplicação.
- **Guardar** -- Guarda o estado actual da aplicação no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar, ficando a ele associado (para operações de salvaguarda subsequentes). Esta interacção realiza-se através do método **`Prompt.newSaveAs()`**. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Quando se abandona uma aplicação com modificações não guardadas (porque se cria ou abre outra), deve perguntar-se se se quer guardar a informação actual antes de a abandonar, através de **`Prompt.saveBeforeExit()`** (a resposta é obtida invocando **`readBoolean()`** ou de **`Form.confirm()`**).



Note-se que a opção **Abrir** não permite a leitura de ficheiros de texto (estes apenas podem ser utilizados no início da aplicação).



A opção **Sair** nunca implica a salvaguarda do estado da aplicação, mesmo que existam alterações.

Avançar Estação do Ano

Esta opção permite avançar a estação do ano, afectando todas as árvores em todos os habitats.

O comando deve apresentar na saída o código para a estação actual: **0** para Primavera, **1** para Verão, **2** para Outono e **3** para Inverno.


Ver Estado de Satisfação Global

O sistema apresenta o somatório dos valores de satisfação de todas as entidades registadas no hotel: animais e funcionários. O valor total a apresentar é o que resulta do arredondamento ao inteiro mais próximo (via **Math.round**).

Menu de Gestão de Animais

Este menu permite operar sobre animais. A lista completa é a seguinte: (i) visualizar todos os animais; (ii) registar um novo animal; (iii) transferir um animal para um habitat; (iv) calcular a satisfação de um animal. As secções abaixo descrevem estas opções.

 As etiquetas das opções deste menu estão definidas em **hva.app.animal.Label**. Os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **hva.app.animal.Prompt** e **hva.app.animal.Message**.

 Estes comandos já estão implementados nas classes da *package* **hva.app.animal** (disponível no GIT), respectivamente: **DoShowAllAnimals**, **DoRegisterAnimal**, **DoTransferToHabitat**, **DoShowSatisfactionOfAnimal**.

Visualizar todos os animais

O formato de apresentação de cada animal é o seguinte:

```
ANIMAL|idAnimal|nomeAnimal|idEspécie|historialDeSaúde|idHabitat
```

Se o animal nunca foi vacinado (com ou sem sucesso), o historial de saúde deve ser apresentado como **VOID**:

```
ANIMAL|idAnimal|nomeAnimal|idEspécie|VOID|idHabitat
```

O historial de saúde é apresentado como descrito em Problemas associados à vacinação, i.e., uma sequência de eventos separados por vírgulas.

Registar um novo animal

O sistema pede o identificador do novo animal. De seguida pede o nome do animal (**Prompt.animalName()**), o identificador da espécie e o identificador do habitat.

Se o identificador da espécie não existe, deve pedir-se o nome da nova espécie (**Prompt.speciesName()**) (cadeia de caracteres) e registá-la com o identificador indicado.

Se o identificador do animal já existir, lança a excepção **DuplicateAnimalKeyException**, não se processando o registo.

Transferir um animal para um habitat

Para realizar a operação, o sistema pede o identificador do animal e o identificador do habitat de destino.

Calcular a satisfação de um animal

O sistema pede o identificador do animal, sendo apresentado o valor da sua satisfação (arredondado ao inteiro mais próximo via **Math.round**).

Menu de Gestão de Funcionários

Este menu permite operar sobre funcionários. A lista completa é a seguinte: (i) visualizar todos os funcionários; (ii) registar um novo funcionário; (iii) atribuir uma nova responsabilidade a um funcionário; (iv) retirar uma responsabilidade a um funcionário; (v) calcular a satisfação de um funcionário. As secções abaixo descrevem estas opções.



As etiquetas das opções deste menu estão definidas em **`hva.app.employee.Label`**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **`hva.app.employee.Prompt`** e **`hva.app.employee.Message`**.



Estes comandos já estão implementados nas classes da *package* **`hva.app.employee`** (disponível no GIT), respectivamente:

`DoShowAllEmployees`, **`DoRegisterEmployee`**, **`DoAddResponsibility`**, **`DoRemoveResponsibility`**, **`DoShowSatisfactionOfEmployee`**.

Visualizar todos os funcionários

O formato de apresentação de cada funcionário é o seguinte:

```
tipo|id|nome|idResponsabilidades
```

Os valores para o campo *tipo* são **VET** ou **TRT**. Os valores para o campo *idResponsabilidades* são os identificadores, separados por vírgulas, dos habitats que um tratador pode limpar ou das espécies de animais que um veterinário pode tratar.

Se o funcionário não tiver responsabilidades atribuídas, o formato de apresentação é o seguinte:

```
tipo|id|nome
```

Registar um novo funcionário

O sistema pede o identificador do novo funcionário, assim como o seu nome (**`Prompt.employeeName()`**) (cadeia de caracteres). De seguida, pede-se o tipo de funcionário (**`Prompt.employeeType()`**). A resposta deve ser **VET** (é registado um veterinário) ou **TRT** (é registado um tratador). Se a resposta não corresponder a nenhum dos dois valores, a pergunta é repetida até se obter uma resposta válida. Quando um funcionário é registado fica sem qualquer responsabilidade.

Se já existir um funcionário com o mesmo identificador, deve ser lançada a excepção **`DuplicateEmployeeKeyException`**, não se realizando qualquer acção.

Atribuir uma nova responsabilidade a um funcionário

O sistema pede o identificador do funcionário e o identificador da nova responsabilidade (**`Prompt.responsibilityKey()`**): o identificador de uma espécie, se for um veterinário; ou o identificador de um habitat, se for um tratador.

Se o funcionário já tinha essa responsabilidade, não é executada nenhuma acção.

Se a responsabilidade não existir, então deve ser lançada a excepção **`NoResponsibilityException`**.

Retirar uma responsabilidade a um funcionário


O sistema pede o identificador do funcionário e o identificador da responsabilidade a retirar (**`Prompt.responsibilityKey()`**): o identificador de uma espécie, se for um veterinário; ou o identificador de um habitat, se for um tratador. Se a responsabilidade não estava atribuída ao funcionário ou não existe, é lançada a excepção **`NoResponsibilityException`**.


Calcular a satisfação de um funcionário

O sistema pede o identificador do funcionário, sendo apresentado o valor da sua satisfação (arredondado ao inteiro mais próximo via **`Math.round()`**).

Menu de Gestão de Habitats

Este menu permite operar sobre habitats. A lista completa é a seguinte: (i) visualizar todos os habitats; (ii) registar um novo habitat; (iii) alterar a área de um habitat; (iv) alterar influência de um habitat sobre uma espécie; (v) plantar uma nova árvore num habitat; (vi) visualizar todas as árvores de um habitat. As secções abaixo descrevem estas opções.

 As etiquetas das opções deste menu estão definidas em **`hva.app.habitat.Label`**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **`hva.app.habitat.Prompt`** e **`hva.app.habitat.Message`**.

 Estes comandos já estão implementados nas classes da *package* **`hva.app.habitat`** (disponível no GIT), respectivamente:
`DoShowAllHabitats`, **`DoRegisterHabitat`**, **`DoChangeHabitatArea`**, **`DoChangeHabitatInfluence`**, **`DoAddTreeToHabitat`**,
`DoShowAllTreesInHabitat`.

Visualizar todos os habitats

O formato de apresentação para cada habitat é o seguinte:

```
HABITAT|idHabitat|nome|área|númeroÁrvores
```

Esta linha inicial pode ser seguida de zero ou mais linhas, cada uma com a descrição de cada árvore pertencente ao habitat usando o seguinte formato:

```
ÁRVORE|idÁrvore|nomeÁrvore|idadeÁrvore|dificuldadeBaseLimpeza|tipoÁrvore|cicloBiológico
```

O valor de *dificuldadeBaseLimpeza* diz respeito ao valor da dificuldade base de limpeza da árvore.

O valores possíveis para *tipoÁrvore* são **PERENE** e **CADUCA**.

Os possíveis valores para o campo *cicloBiológico* são:

Tipo de Árvore	Inverno	Primavera	Verão	Outono
CADUCA	SEMFOLHAS	GERARFOLHAS	COMFOLHAS	LARGARFOLHAS
PERENE	LARGARFOLHAS	GERARFOLHAS	COMFOLHAS	COMFOLHAS

Registar um novo habitat

O sistema pede o identificador, o nome (**`Prompt.habitatName()`**) (cadeia de caracteres) e a área do habitat (**`Prompt.habitatArea()`**) (número inteiro), sendo registado o novo habitat.

Deve ser lançada a exceção **`DuplicateHabitatKeyException`**, se existir um habitat com o mesmo identificador, não se realizando qualquer acção.

Alterar a área de um habitat

O sistema pede o identificador do habitat assim como a nova área desse habitat (**`Prompt.habitatArea()`**) (número inteiro).

Alterar influência de um habitat sobre uma espécie

O sistema pede o identificador do habitat, o identificador da espécie e se a influência do habitat (**`Prompt.habitatInfluence()`**) sobre a espécie indicada é positiva (resposta: **POS**), negativa (resposta: **NEG**), ou neutra (resposta: **NEU**). Se a resposta não corresponder a nenhum dos três valores, esta última pergunta é repetida até se obter uma resposta válida.

Plantar uma nova árvore num habitat

O sistema pede o identificador do habitat e o identificador da nova árvore a plantar. De seguida, pede-se o nome (**`Prompt.treeName()`**) (cadeia de caracteres), a idade da árvore (**`Prompt.treeAge()`**), a dificuldade base de limpeza associada à árvore (**`Prompt.treeDifficulty()`**) (número inteiro) e o tipo de árvore (**`Prompt.treeType()`**) (**PERENE** para árvore de folha perene; **CADUCA** para árvores de folha caduca), registado-se a nova árvore. Se a resposta relativa ao tipo não corresponder a nenhum dos dois valores, esta última pergunta é repetida até se obter uma resposta válida.

Se já existir uma árvore com o mesmo identificador (em qualquer habitat), é lançada a exceção **`DuplicateTreeKeyException`**, não se realizando qualquer acção.

A nova árvore deve ser apresentada na saída. O formato de apresentação é como descrito acima.

Visualizar todas as árvores de um habitat

O sistema pede o identificador do habitat e apresenta as árvores que nele existem. O formato de apresentação é como descrito acima.

Menu de Gestão de Vacinas

Este menu permite operar sobre vacinas e vacinações. A lista completa é a seguinte: (i) visualizar todas as vacinas; (ii) registar uma nova vacina; (iii) vacinar um animal; (iv) listar o histórico de vacinações. As secções abaixo descrevem estas opções.



As etiquetas das opções deste menu estão definidas em **`hva.app.vaccine.Label`**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **`hva.app.vaccine.Prompt`** e **`hva.app.vaccine.Message`**.



Estes comandos já estão implementados nas classes da *package* **`hva.app.animal`** (disponível no GIT), respectivamente: **`DoShowVaccine`**, **`DoRegisterVaccine`**, **`DoVaccinateAnimal`**, **`DoShowVaccinations`**.

Visualizar todas as vacinas

O formato de apresentação de cada vacina, uma por linha, é o seguinte:

```
VACINA|idVacina|nomeVacina|numeroDeAplicações|espécies
```

O campo *espécies* contém os identificadores, separados por vírgulas, das espécies de animais que podem receber a vacina.

Registar uma nova vacina

O sistema pede o identificador da vacina, o nome (**`Prompt.vaccineName()`**) (cadeia de caracteres) da vacina e os identificadores das espécies que podem receber esta vacina (**`Prompt.listOfSpeciesKeys()`**) (lista de identificadores separados por vírgulas -- os espaços brancos são irrelevantes nesta resposta). De seguida regista a nova vacina.

Deve ser lançada a excepção **`DuplicateVaccineKeyException`**, se existir uma vacina com o mesmo identificador, não se realizando qualquer acção.

Se algum dos identificadores de espécies indicados for desconhecido, deve ser lançada a excepção **`UnknownSpeciesKeyException`**, não sendo registada a nova vacina.

Vacinar um animal

O sistema pede o identificador da vacina, o identificador do veterinário (**`Prompt.veterinarianKey()`**) e o identificador do animal a vacinar. Se o identificador não for de um veterinário, deve ser lançada a excepção **`UnknownVeterinarianKeyException`**. Se o veterinário não tiver permissão para ministrar a vacina, deve ser lançada a excepção **`VeterinarianNotAuthorizedException`** e o comando não tem qualquer efeito. Se a vacina não for adequada ao animal, deve ser apresentada uma mensagem de aviso (**`Message.wrongVaccine()`**). Note-se que, neste caso, a vacinação acontece e os danos para o animal são registados.

Listar o histórico de vacinações


O sistema lista todas as vacinas aplicadas, por ordem de aplicação, usando o seguinte formato:


```
REGISTO-VACINA|idVacina|idVeterinário|idEspécie
```

Menu de Consultas

O menu de consultas permite efectuar pesquisas sobre as entidades do domínio e suas relações. Sempre que for feita uma consulta e nenhuma entidade satisfizer as condições associadas ao pedido, nada deve ser apresentado.

A lista completa é a seguinte: (i) consultar animais de um habitat; (ii) consultar vacinas de um animal; (iii) consultar actos médicos de um veterinário; (iv) consultar vacinas dadas com incúria. As secções abaixo descrevem estas opções.

 As etiquetas das opções deste menu estão definidas em **`hva.app.search.Label`**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **`hva.app.search.Prompt`** e **`hva.app.search.Message`**.

 Estes comandos já estão implementados nas classes da *package* **`hva.app.search`** (disponível no GIT), respectivamente:
`DoShowAnimalsInHabitat`, **`DoShowMedicalActsOnAnimal`**, **`DoShowMedicalActsByVeterinarian`**, **`DoShowWrongVaccinations`**.

Consultar animais de um habitat

O sistema pede o identificador do habitat, apresentando os animais que residem nesse habitat.
O formato de apresentação é como indicado para a listagem de animais.

Consultar vacinas de um animal

O sistema pede o identificador do animal, apresentando as vacinações (por ordem de aplicação).
O formato de apresentação é como indicado para a listagem de vacinações.

Consultar actos médicos de um veterinário

O sistema pede o identificador de um funcionário. Se o identificador não existir ou não for de um veterinário, deve ser lançada a excepção **`UnknownVeterinarianKeyException`**. Caso contrário, são apresentadas todas as vacinações por ele realizadas (por ordem de aplicação).
O formato de apresentação é como indicado para a listagem de vacinações.

Consultar vacinas dadas com incúria

O sistema apresenta todas as ocorrências de vacinações que provocaram problemas de saúde aos animais.
O formato de apresentação é como indicado para a listagem de vacinações (por ordem de aplicação).

Inicialização por Ficheiro de Dados Textuais

Por omissão, quando a aplicação começa, não contém nenhuma informação e está na estação **Primavera**. No entanto, além das opções de manipulação de ficheiros descritas no menu principal, é possível iniciar exteriormente a aplicação com um ficheiro de texto especificado pela propriedade Java **`import`**. Quando se especifica esta propriedade, a aplicação é povoada com os objectos correspondentes ao conteúdo do ficheiro textual indicado.

No processamento destes dados, assume-se que não existem entradas mal-formadas e assume-se que os identificadores referidos numa entrada já foram previamente descritos. Sugere-se a utilização do método **`String.split()`** para dividir uma cadeia de caracteres em campos.

No formato abaixo: para os habitats, só os 4 primeiros campos são obrigatórios, correspondendo os opcionais a árvores implantadas no recinto; para os tratadores, só os 3 primeiros campos são obrigatórios, correspondendo os opcionais a habitats; e, para os veterinários e vacinas, só os 3 primeiros campos são obrigatórios, correspondendo os opcionais a espécies de animais. Se alguma das listas for vazia, então não se apresenta o último campo (nem o separador).

Formato do ficheiro de entrada textual

[Collapse]

```
ESPÉCIE|id|nome
ÁRVORE|id|nome|idade|dificuldade|tipo
HABITAT|id|nome|área|idÁrvore1,...,idÁrvoreN
ANIMAL|id|nome|idEspécie|idHabitat
TRATADOR|id|nome|idHabitat1,...,idHabitatN
VETERINÁRIO|id|nome|idEspécie1,...,idEspécieN
VACINA|id|nome|idEspécie1,...,idEspécieN
```

Ver abaixo a forma de utilizar estes ficheiros.

A codificação dos ficheiros a ler é garantidamente UTF-8.

```

ESPÉCIE|C10|ave
ESPÉCIE|C8|mamífero
ESPÉCIE|C1|peixe
ÁRVORE|T1|Pinheiro 1|5|20|PERENE
ÁRVORE|T4|Pinheiro 4|60|20|PERENE
ÁRVORE|T2|Oliveira|1200|10|PERENE
ÁRVORE|T6|Figueira|5|10|CADUCA
ÁRVORE|T3|Plátano|300|20|CADUCA
ÁRVORE|P1|Plátano|100|20|CADUCA
ÁRVORE|F30|Figueira|50|20|CADUCA
HABITAT|AR1|Aldeia dos Macacos|20|T3,P1,F30,T2
HABITAT|AR2|Floresta tropical|30|T1,T4,T6
HABITAT|AR3|Deserto|3
ANIMAL|A1|Alex|C10|AR1
ANIMAL|A2|Nemo|C1|AR2
ANIMAL|AA|Bobi|C8|AR1
ANIMAL|MA|Chita|C8|AR2
TRATADOR|W2S04|John Figueiredo|AR2,AR3
TRATADOR|W20|Rohit Figueiredo|AR1
TRATADOR|W24|Rohit Figueiredo
VETERINÁRIO|V2|Jorge Figueiredo|C10,C8,C1
VETERINÁRIO|V4|Filomena Figueiredo
VETERINÁRIO|VR|Abdul Figueiredo|C8
VACINA|V3|Tétano|C8
VACINA|V200|Parasitas intestinais|C10,C8,C1
VACINA|V42172|Tétano|C10
VACINA|Vexperimental|Gripe A1|C8

```

💡 Note-se que o programa nunca produz ficheiros com este formato.

Execução dos Programas e Testes Automáticos

Usando os ficheiros **test.import**, **test.in** e **test.out**, é possível verificar automaticamente o resultado correcto do programa. Note-se que é necessária a definição apropriada da variável **CLASSPATH** (ou da opção equivalente **-cp** do comando **java**), para localizar as classes do programa, incluindo a que contém o método correspondente ao ponto de entrada da aplicação (**hva.app.App.main**). As propriedades são tratadas automaticamente pelo código de apoio (**po-uilib**).

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp hva.app.App
```

Assumindo que aqueles ficheiros estão no directório onde é dado o comando de execução, o programa produz o ficheiro de saída **test.outhyp**. Em caso de sucesso, os ficheiros das saídas esperada (**test.out**) e obtida (**test.outhyp**) devem ser iguais. A comparação pode ser feita com o comando:

```
diff -b test.out test.outhyp
```

Este comando não deve produzir qualquer resultado quando os ficheiros são iguais. Note-se, contudo, que este teste não garante o correcto funcionamento do código desenvolvido, apenas verificando alguns aspectos da sua funcionalidade.

Categories: **Ensino** **PO** **Projecto de PO**