**Descrição:**

**Main.java:**

Aqui que a execução do programa ocorre. Recebendo os import das classes auxiliares e subclasses,

Esta classe possui um menu para poder realizar as ações desejadas(Adicionar animal, alimentá-lo, verificar se ele corre risco de extincao, etc.)\* Ela possui dois métodos: "ShowMenu()" que mostra as opções que veresão ser tomadas, e o "cls()", que "emula" a função

system("cls") do c++. No mais, a classe possui diversos atributos que auxiliarão na criação dos objetos, e a maior parte do programa

**Animal.java**

SuperClasse responsável por definir todo Animal deste programa. Estende para duas classes: "Animal\_Domesticado" e "Animal\_Selvagem". Decidi usar estas subclasses pois poderia criar facilmente subclasses a partir delas (Animal\_Domesticado derivaria em cão, gato, coelho, etc.).

Assumindo que cada animal tenha um nome Científico (ex: Panthera tigris), uma classe (ex: mamífero, inseto), um codigo(000000432812), idade(ex: 13 anos, 35 anos), um peso, nomePopular(Pantheras tigris == Tigre), um contador de doenças( crece conforme o animal come alimentos estragados) e um verificador se ele está doente ou não, é possível criar um objeto desta classe. Nos métodos, temos o "Comer(Comida)", onde, dada uma comida X, o animal alimenta-se dela, um método "Verificar\_perigo\_Extincao" que retorna um indicador de qual o perigo de extinção de um animal, "Velhice(int Expectativa Vida)", que dada uma expectativa de vida, retorna uma mensagem que varia dependendo da (idade atual/expectativa vida), além de um método que indica se ele está doente.

Existem outros métodos, como o "toString(Animal)", responsável por realizar uma função parecida com o operador "<<" do c++, e o "Habitat()" indicando qual lugar será mais provável encontrar estes animais, e este será sobreescrito por suas subclasses.

**Animal\_Domesticado.java**

Sendo uma "filha" da classe Animal, Animal\_Domesticado herda todos os atributos da classe Añimal. Portanto não explicarei dos detalhes da superclasse.

Então, o que a torna especial? Não muito infelizmente. Alem de possuir um static final para ter o controle do que esta criando, cada animal possui um apelido

(cachorro = Totó), e o objeto dono e auxDono indica quais são as informações do Dono. A classe tem um método para verificar qual a qualidade do

criador, sem contar que ele sobreescreve as funções "Habitat" e "toString"

**Animal\_Domesticado.java**

"Irmã" de Animal\_Domesticado, a classe ANumal\_Selvagem é mais uma classe que herda da classe animal. Não tem tantos atributos quanto a superclasse, mas possui uns importantes.

Cada animal da classe "AnimalSelvagem" possui uma variável indicando se ele é o líder do grupo(Caso já tenha um líder, nenhum outro poderá ser, conforme esta a condição no main), e existe o vetor grupo, constituído de várioas objetos Animal\_Selvagens, além do atributo MAXGRUPO, que delimita o tamanho máximo do grupo.

Para os métodos, temos um método para adicionar algum animal novato ao grupo, além dos "ToString()" e dos "Habitat()"

Todo animal deve ter um dono, certo? Teoricamente, não (E na pratica também não, mas vamos que vamos). Esta classe

vai ser importante na classe Dono, para indicar que todo animal tem um dono (Um mundo melhor seria este imagino).

**Dono.java**

Todo animal deve ter um dono, certo? Teoricamente, não (E na pratica também não, mas vamos que vamos). Esta classe vai ser importante na classe Dono, para indicar que todo animal tem um dono (Um mundo melhor seria este imagino).

O que o dono possui? Primeiramente, um nome e o numero de animais que este possui, a qualidade do tratamento será definida através de um método, aonde entra uma nota entre 0 e 10 e sai um conceito (Parecido com o sistema e notas de uma Universidade)

**Comida.java**

A classe Comida contém as informações mínimas para uma refeição. Não possui muitos atributos "nomeComida" recebe o nome da cdomida, que é de certa forma desnecessário, diferente das "calorias", que definirão quanto o animal vai engordar. MesesProdução indica quanto tempo esta ração/comida foi produzida ou encontra-se morte, sendo este atributo útil para verificar a "qualidade" da comida, que impacta na saúde do animal

Além dos velhos conhecidos getters e setters, tempos o método "verificar\_Qualidade(int)", retornando se a comida ainda é comestível ou não.

**Requisitos funcionais**

***Todos os atributos e funções membros devem estar relacionados a classe***

1. Pelomenos 4 atributos
2. Pelo menos 4 funções membros sem incluir get e set

**Requisitos de implementação**

1. Todos os atributos devem ser inicializados. Fez validação de dados
2. Três construtores, incluindo um construtor de cópia e construtor com parâmetros defaults. Verificaalocaçãodentro do construtor de cópia.
3. Deveter um atributo string
4. Um atributo static. Correta modelagem dos statics?
5. Um atributoconst static
6. Dois métodos constantes (não pode ser get)
7. Um array
8. Uma função inline (não pode ser get ou set)
9. Método com passagem por referência usando ponteiro
10. Método static – deve ser chamado no main
11. Composição com a classe Data. Fez uso do objeto criado?
12. O que é const deve ser const.
13. Alocação dinâmica de memória. A memória é desalocada?
14. friendOperator<<
15. Operator=
16. vectorpush\_back

Requisitos para as classes adicionais (pelo menos duas)

* Operator =
* Alocação dinâmica - se houver vazamento de memória a classe toda é desconsiderada
* Usar o destrutor
* Construtor de cópia
* Operator <<friend
* Um conststatic

**Requisitosherança**

* Diagrama de classes (obrigatório salvar também o png do diagrama no gitHub)
  + Herança pública
  + Construtor de cópia, e sobrecargas dos operadores de atribuição (=) e << (cout<< base) para a classe base e derivada
  + Usar Protected acessando diretamente os atributos na classe derivada
  + Alocação dinâmica de memória na classe base e derivada
  + Sobrescrita de método: chamar dentro do método da classe derivada o método correspondente da classe base usando ::
  + No main: criar um ponteiro da classe base para alocar memória para a classe derivada e chamar os vários métodos implementados