UNIVERSIDADE
FEDERAL DO CEARÁ

Programação Orientada a Objetos
Trem vol. 2

David Sena Oliveira

Quixadá, Outubro de 2014

1 Descrição

Essa é a primeira evolução do Trabalho do Trem. Você precisará implementar herança, classes abstratas e interfaces.

2 Diagrama

O diagrama da Figura 1 apresenta as classes a serem implementadas.

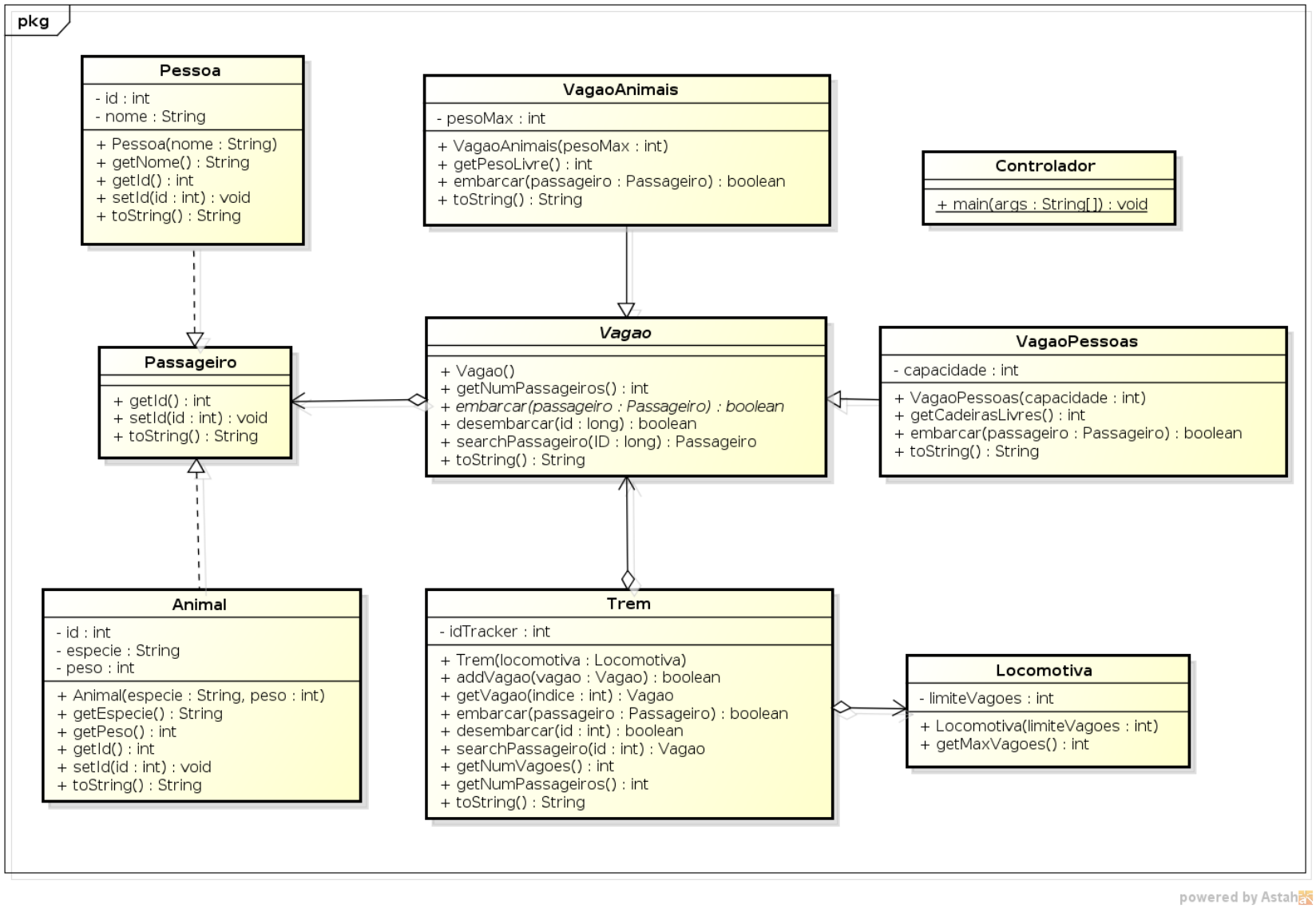

Figura 1: Diagrama de Classes

3 Métodos

A maioria dos métodos, principalmente os get e set são de implementação trivial. Os métodos não triviais estão descritos abaixo.

3.1 Controlador

O controlador mais uma vez é brinde. Dado, o controlador apresentado na Listings 1, a saída gerada deve ser algo como:

```
Trem{
    ( 4:Do )
    [ 1:cao:20 3:gato:10  _:5 ]
    ( 5:Re 6:Mi )
    [ 2:cao:30  _:0 ]
}
```

Listing 1: Controlador

```
package controle;
import passageiros.Animal;
import passageiros.Pessoa;
import trem.Locomotiva;
import trem.Trem;
import vagoes.VagaoAnimais;
import vagoes.VagaoPessoas;

public class Controlador {
    public static void main(String[] args) {
        Locomotiva locomotiva = new Locomotiva(10);

        Trem trem = new Trem(locomotiva);

        trem.addVagao(new VagaoPessoas(1));
        trem.addVagao(new VagaoAnimais(35));
        trem.addVagao(new VagaoPessoas(2));
        trem.addVagao(new VagaoAnimais(30));

        trem.embarcar(new Animal("cao", 20));
        trem.embarcar(new Animal("cao", 30));
        trem.embarcar(new Animal("gato", 10));
        trem.embarcar(new Animal("cobra", 10));

        trem.embarcar(new Pessoa("Do"));
        trem.embarcar(new Pessoa("Re"));
        trem.embarcar(new Pessoa("Mi"));
        trem.embarcar(new Pessoa("Fa"));

        System.out.println(trem);

    }
}
```

3.2 Trem e Locomotiva

- A classe Locomotiva é igual ao volume 1.
- Na classe Trem o único método alterado é o `toString()`. Ele contrói a String interando sobre os vagões e montando a String usando

  `"\t" + vagao.toString() + "\n"`

3.3 Passageiro

- O Passageiro se tornou uma interface.
- Método `toString`() retorna todos os atributos da classe no formato

  `"id:atr1:atr2"`

3.4 Pessoa

- Pessoa implementa Passageiro. Possui além do id, uma atributo nome.
- O método `String toString`() retorna `"id:nome"`

3.5 Animal

- Animal implementa Passageiro. Possui além do id, uma atributo espécie e um peso.
- O método `String toString`() retorna `"id:especie:peso"`

3.6 Vagao

- Vagão se tornou classe abstrata. Perdeu o método `getCapacidade` e o atributo `capacidade`. A classe `embarcar` tornou-se abstrata.
- O Vetor de passageiros se tornou `protected` para que possa ser acessado pelas classes derivadas.
- Método `String toString`() retorna uma String contendo a descrição dos passageiros do vagão. Deve concatenar as chamadas `toString`() dos passageiros.

  `"1:Carlos 2:Mario"`

3.7 VagaoPessoas

- VagaoPessoas extende a classe abstrata Vagao. Possui um atributo capacidade que define quantas cadeiras disponíveis existem.
- Método `boolean embarcar(Passageiro passageiro)` verifica se existe cadeira vaga, verifica se o passageiro é do tipo pessoa e então adiciona o passageiro. Use `passageiro instanceof Pessoa`.
- Método `String toString`() inicia e termina com parênteses, chama o método `super.toString`() para pegar a lista de passageiros e adiciona um `_` para cada cadeira vaga. Um exemplo de saída para um vagão de capacidade 5, com 2 pessoas seria:

  `( 4:Carlos 5:Mario _ _ _ )`

3.8 VagaoAnimais

- `VagaoAnimais` extende a classe abstrata `Vagao`. Possui um atributo pesoMax que define qual o peso total que ele comporta.
- Método `boolean embarcar(Passageiro passageiro)` verifica se o passageiro é do tipo animal, verifica se o animal ainda cabe e então adiciona o passageiro. Use `passageiro instanceof Animal`.
- Método `String toString()` inicia e termina com colchetes, chama o método `super.toString()` para pegar a lista de passageiros e adiciona um `_:pesoLivre` para cada cadeira vaga. Um exemplo de saída para um vagão de capacidade 500, com 3 animais de 100 kilos seria:

  ```
  [ 1:Cao:100 2:Gato:100 3:Porco:100 _:200 ]
  ```