19 - A classe TipoEspecialidade foi feita para evitar erros na hora de inserção no banco e para limitar à apenas as especialidades que o domínio atende;

A classe TipoAtendimento limita a somente dois tipos de atendimento, que são Primeira Vez e Retorno;

A classe TipoEspecie limita a seleção somente aos tipos de animais que são atendidos no domínio;

A classe TipoPagamento aceita apenas as formas de pagamento aceitas pelo domínio;

20 - public enum NomeClasseEnumarada {

Enum1, enum2, enum3, ..., enumN;

}

**public** **enum** *TipoEspecialidade* {

***Cardiologista***, ***Oftalmologista***, ***Anestesiologia***, ***CirurgiaoGeral***, ***Dermatologia***;

}

**public** **enum** *TipoAtendimento* {

***Retorno***, ***PrimeiraVez***;

}

**public** **enum** *TipoEspecie* {

***Cachorro***, ***Gato***, ***Passaro***, ***Roedor***;

}

**public** **enum** *TipoPagamento* {

***aVista***, ***Cartao***, ***Convenio***;

}

21 – O atributo mediaSalarial na classe Veterinario, guarda uma

média de salário entre todos os objetos do tipo Veterinario;

O atributo horarioPico na classe Consulta foi feito, pensando em qual horário a clinica atende mais pacientes, podendo até mesmo registrar essa informação no Google.

O Método mediaSalario calcula a média salarial de todos os veterinários.

O método mediaDisponibilidade calcula qual a média de disponibilidade de todos os veterinários, para saber qual dia tem mais veterinário disponível.

O atributo QtdClientes na classe Cliente, guarda o valor do total de cliente que a clínica tem em seu banco de dados.

O método obterQtdClientes na classe Cliente calcula a quantidade de clientes que possui no banco de dados.

22 – Para atributos:

private static TipoObjeto NomeVariavel;

para métodos:

public static void NomeDoMetodo(TipoObjeto Argumento1, TipoObjeto ArguentoN) {

implementação...

}

23 – A vantagem dessa dependência é aumento no desempenho, a desvantagem é queda no encapsulamento e aumento no acoplamento.

24 – public class Cliente {

private Animal animal;

...

}

25 – A vantagem é aumento no encapsulamento e queda no acoplamento, a desvantagem é o baixo desempenho.

26 – public class ControleCliente {

public void adicionaCliente(Cliente c) {

...

}

public void consultaCliente(Cliente c) {

...

}

}

27 –

28 – public class ControleExame {

public void marcarExame() {

Animal a = new Animal();

Veterinario v = new Veterinario();

...

}

}