INSTITUTO FEDERAL Minas Gerais Campus Formiga	**Ciência da Computação** Trabalho Prático Criação de um protótipo de controle de petshop chamado BIXIM.

Disciplina: Programação Orientada a Objetos	Data: ___/___/_____	
Professor: Bruno Ferreira	Turma: 4° período	
Aluno	Valor: 28,0	Nota:

Objetivo: Reforçar conceitos orientados a objetos.

Forma de Entrega: O código fonte deverá ser entregue em um arquivo zipado via Google Class. O CODIGO FONTE DEVERÁ SER APRESENTADO INDIVIDUALMENTE PARA O PROFESSOR em sala no dia **01/12/2022**.

OBS: Será observado e avaliado a legibilidade do código sendo fundamental uma boa identação e a utilização de comentários.

O trabalho é em dupla, se for verificada cópia de trabalho os trabalhos serão considerados com nota zero.

Atenção: Leia todo o documento antes de começar a implementação.

O trabalho consiste em melhorar o primeiro protótipo de acordo com novas necessidades do cliente.

Passo 1) O sistema continua com o cadastro de serviços, mas agora deve-se tratar os serviços de forma diferente para Gatos e Cachorros. Um gato paga um adicionou de 15% sobre o valor informado no cadastro de serviço. Já os cachorros pagam um adicionou de 10% (Esse petshop só atende esses dois tipos de animais). Além disso, um cachorro agora tem a propriedade indicando se ele é de pedigree(true ou false). Já o gato tem duas propriedades novas. O número do documento do dono e o tipo do documento (CPF ou CNPJ). Para esse primeiro passo, siga o Diagrama de Classe no final desse documento, modelamos esse problema com os conceitos de classe abstrata, herança e enumeradores.

Veja um exemplo de inserção de um cachorro na Figura 1 e de um gato na Figura 2. As alterações e deleções seguem o mesmo exemplo do Trabalho 01.

```
====================
CADASTRO DE ANIMAL
====================
Insira seu código
1
Nome do animal
Totó
Endereço do animal
centro
Cidade do animal
Formiga
Tipo: 1) Cachorro 2) Gato
1
Tem perdigree? (true ou false)
true
Dados inseridos com sucesso
```

Figura 1 – Fluxo de cadastro de um cachorro.

```
==================
CADASTRO DE ANIMAL
==================
Insira seu código
2
Nome do animal
Xaninho
Endereço do animal
centro
Cidade do animal
divinopolis
Tipo: 1) Cachorro 2) Gato
2
Tipo documento do dono.: 1) CPF 2) CNPJ
1
Número de documento do dono
043.987.987-87
CPF Inválido.
Dados inseridos com sucesso
```

Figura 2 – Fluxo de cadastro de um gato.

Referente ao cadastro, tem-se uma regra nova. O sistema deve validar o CPF e o CNPJ. Para isso crie dois métodos na classe “Util” do pacote “útil”, conforme mostra o diagrama de classe. Deve-se criar também uma classe para representar o erro de documento inválido. Essa classe se chama “DocumentoException”. Mais uma vez, analise essa classe no pacote “útil” do diagrama.

Dica: Depois dessas alterações, verifique se o seu sistema está funcionando corretamente. As outras camadas do sistema não terão grandes alterações, somente as camadas de modelo e de visão.

Passo 2) O cliente não gostou do uso de vetores no primeiro projeto. Ele deseja uma estrutura em memória que seja **dinâmica**, ou seja, vamos usar o pacote Collections do Java. O cliente exigiu que a estrutura de dados **não aceite dados duplicados**, então a interface List não seria muito indicada. Essa é uma alteração que vai impactar diretamente a camada DAO e consequentemente a classe “BancoDeDados” do pacote “bd”. Veja o diagrama de classe para mais detalhes.

Para uma visão geral de como ficou o sistema, imagine que o usuário cadastrou o serviço e o atendimento mostrados na Figura 3.

```
====================
CADASTRO DO SERVIÇO
====================
Insira o código do serviço
1
Descrição do serviço
Banho
Valor do serviço
50
Dados inseridos com sucesso
```

```
==================
CADASTRO DE ATENDIMENTOS
==================
Insira o código do antedimento:
100
Insira o código do animal:
2
Insira o código do servico:
1
Insira a data do antedimento:
03/11/2022
Dados inseridos com sucesso
```

Figura 3 – Cadastro de um serviço e de um atendimento.

A Figura 4 mostra como ficaria o relatório de animais (opção 4 do menu)

```
================================
LISTAGEM DE ANIMAIS CADASTRADOS
================================
Deseja realmente imprimir o relatório? (S/N)
s
Codigo: 1 Nome: Totó Endereco: centro Cidade: Formiga Perdigree: true
Codigo: 2 Nome: Xaninho Endereco: centro Cidade: BH Documento: 65.987.654-98 (Pessoa física)
```

Figura 4 – Listagem de animais cadastrados.

A Figura 5 mostra como ficaria o relatório de serviços (opção 5 do menu)

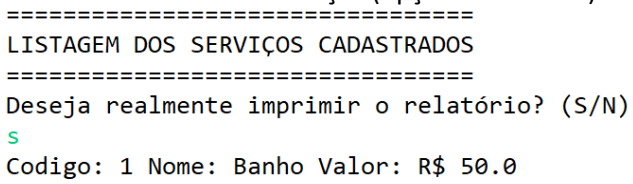

Figura 5 – Listagem de serviços cadastrados.

A Figura 6 mostra como ficaria o relatório de atendimentos (opção 6 do menu)

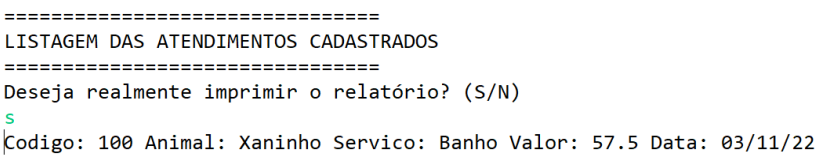

Figura 6 – Listagem de atendimentos cadastrados.

A Figura 7 mostra como ficaria a impressão da nota fiscal (opção 7 do menu). Veja que o serviço impresso foi de R$ 57,50. Esse valor corresponde aos R$ 50,00 do valor base do banho **+** os 15% de acréscimo porque o Xaninho é um Gato. Esse mesmo calculo foi aplicado na Figura 6.

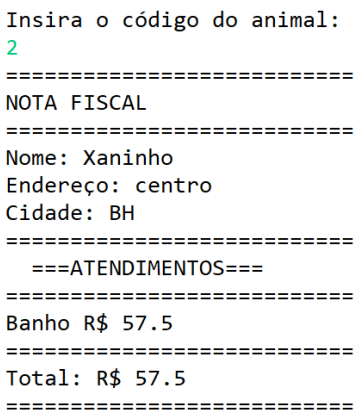

Figura 7 – Impressão da nota fiscal.

A Figura 8 mostra o relatório da opção 9 do menu. Veja que foi considerado o valor com o acréscimo de 15%

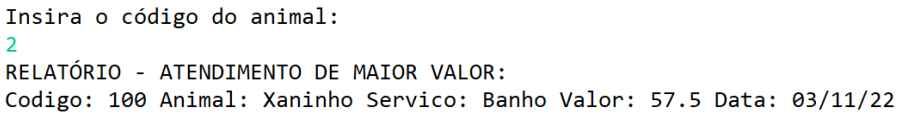

Figura 8 – Relatório com os dados do maior atendimento de um animal.

A Figura 9 mostra o relatório da opção 10 do menu. Veja que também foi considerado o valor com o acréscimo de 15%

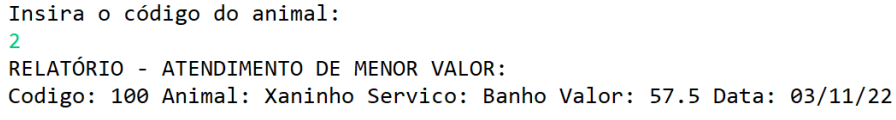

Figura 9 – Relatório com os dados do menor atendimento de um animal.

A Figura 10 mostra o relatório da opção 10 do menu e mais uma vez foi considerado o valor com o acréscimo de 15% para gatos

```
Insira o código do animal:
2
RELATÓRIO - ATENDIMENTO DE MENOR VALOR:
Codigo: 100 Animal: Xaninho Servico: Banho Valor: 57.5 Data: 03/11/22
```

Figura 8 – Relatório com o valor total dos atendimentos de um animal.

Passo 3) Esse trabalho tem como principal função trabalhar de forma Orientada a Objetos. Para te ajudar nessa implementação, o analista de sistemas já desenhou toda a estrutura de classes que o sistema terá. Veja na Figura 11 todos os pacotes e classes com suas propriedades e métodos desse projeto.

A estrutura mostrada no diagrama, segue uma estrutura de classes muito próxima de um padrão profissional, então use essa estrutura para ganhar todos os pontos do trabalho.

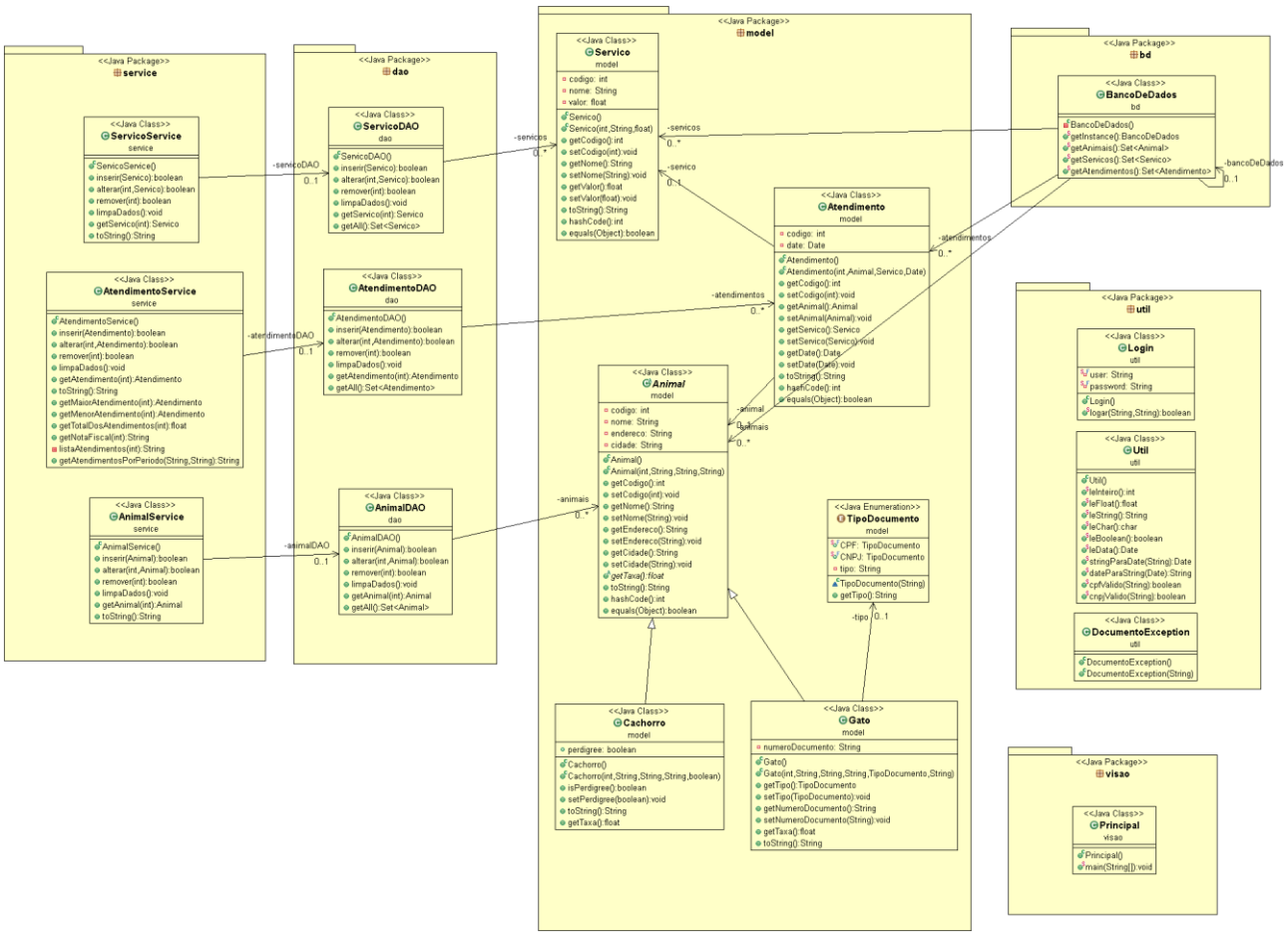

Figura 11 – Diagrama de Classe do projeto

Subpasso 3a) Crie todos os pacotes que o diagrama indica

Subpasso 3b) Crie as três classes do pacote “model”, pois várias outras classes dependem delas.

Subpasso 3c) Agora implemente a classe “BancoDeDados”, ela usa um padrão de projeto chamado Singleton, mas o importante a saber nessa classe é que ela contém os três vetores que armazenaram os dados, esses vetores são privados e estáticos. Os vetores são, respectivamente, do tipo Animal, Serviço e Atendimento (criados no passo 2). Os métodos públicos e estáticos dessa classe retornam esses vetores.

Subpasso 3d) As classes do pacote “dao” são responsáveis por manipular as coleções, ou seja, seus métodos inserem, alteram, removem dados na estrutura escolhida, existem métodos também para retornarem um ou todos os dados da coleção. O nome “dao” vem do inglês, Data Access Object, ou seja,

objeto que acessam os dados, os quais nesse caso, não estarão nos vetores. Esse é mais um padrão de projeto e ele tem uma regra simples, as classes desse pacote não tem regras de negócio, ou seja, eles apenas inserem/alteram/removem/consulta sem seguir qualquer regra ou se preocupam com validações de dados.

Subpasso 3e) Nesse passo, as classes do pacote "service" serão implementadas. Nessa camada de software implementamos todas as regras de negócio e validações do sistema, por exemplo, aqui verificamos se todos os dados foram preenchidos, ou se um cliente ou serviço foi cadastro antes de ser informado em um atendimento. Depois de fazer as devidas validações, essas classes usam as classes do "dao" para inserir os dados nos vetores.

Subpasso 3f) Vamos implementar as classes da camada de visão. É nesse pacote que criamos as classes que irão, de fato, interagir com os usuários, devemos evitar entrar com os dados e imprimir informações nas classes de dos pacotes implementadas nos passos anteriores. Ou seja, faça a leitura de dados e a impressão de informações somente nas classes desse pacote.

Para facilitar, no apêndice A, o professor está disponibilizando a classe "Util" do pacote "util".

Passo 4) Faça testes e verifique se o desenvolvimento está correto.

Passo 5) Gerar o executável (.JAR). Pesquise na internet como fazer isso no NetBeans e um ".bat" para facilitar a execução.

Você deseja ganhar dois pontos extra, então documente todos os métodos das classes e armazene vários logins e suas respectivas senhas em um arquivo do tipo texto ou binário.

Fim

Apêndice A

```
package util;

import java.text.DateFormat;
import java.util.Date;
import java.util.Scanner;


public class Util {

 public static int leInteiro() {
        Scanner entrada;
        int valor = 0;
        boolean erro = true;
        while (erro) {
            try {
                entrada = new Scanner(System.in);
                valor = entrada.nextInt();
                erro = false;
            } catch (Exception e) {
                      System.out.println("Erro ao digitar. Tente novamente.");
                      entrada = null;
            }
        }
        return valor;
    }
```

```java
public static float leFloat() {
    Scanner entrada;
    float valor = 0;
    boolean erro = true;
    while (erro) {
        try {
            entrada = new Scanner(System.in);
            valor = entrada.nextFloat();
            erro = false;
        } catch (Exception e) {
          System.out.println("Erro ao digitar. Tente novamente.");
          entrada = null;
        }
    }
    return valor;
}

public static String leString() {
    Scanner entrada;
    String valor = "";
    boolean erro = true;
    while (erro) {
        try {
            entrada = new Scanner(System.in);
            valor = entrada.nextLine();
            erro = false;
        } catch (Exception e) {
            System.out.println("Erro ao digitar. Tente novamente.");
            entrada = null;
        }
    }
    return valor;
}

public static char leChar() {
    Scanner entrada;
    char valor = 0;
    boolean erro = true;
    while (erro) {
        try {
            entrada = new Scanner(System.in);
            valor = entrada.next().charAt(0);
            erro = false;
        } catch (Exception e) {
            System.out.println("Erro ao digitar. Tente novamente.");
            entrada = null;
        }
    }
    return valor;
}


public static Date leData() {
    Scanner entrada;
    Date valor = null;
    boolean erro = true;

    DateFormat df = DateFormat.getDateInstance();

    while (erro) {
        try {
            entrada = new Scanner(System.in);
```

```
                String dtString = entrada.next();
                valor = df.parse(dtString);
                erro = false;
            } catch (Exception e) {
                System.out.println("Erro ao digitar. Tente novamente.");
                entrada = null;
            }
        }
        return valor;
    }

    public static Date stringParaDate(String dt){

        DateFormat df = DateFormat.getDateInstance();
        Date retorno = null;
        try {
            retorno = df.parse(dt);
        } catch (Exception e) {
            System.out.println("Erro ao digitar a data. Tente novamente.");
            retorno = null;
        }

    return retorno;
    }

    public static String dateParaString(Date dt){
        DateFormat df = DateFormat.getDateInstance();
        String retorno = null;
        try {
            retorno = df.format(dt);
        } catch (Exception e) {
            System.out.println("Erro ao converter a data. Tente novamente.");
            retorno = null;
        }

    return retorno;
    }


}
```