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: 10,0	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 **26/10/2022**.

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 criar um protótipo funcional de um sistema, que é responsável por realizar o registro de atendimentos realizadas por clientes/animais no petshop BIXIM. Este protótipo deve receber os dados referentes aos animais (código, nome, endereço, cidade) e serviços (código, descrição e valor), o empregado deve informar na opção de atendimentos, qual um código, o serviço, o animal e a data do atendimento. O sistema deve disponibilizar informações ao usuário como: a) valor total dos atendimentos por animal; b) relatório dos dados cadastrados dos animais, serviços e atendimentos; c) atendimento mais caro por animal; d) atendimento mais barato por animal; e finalmente, e) emitir um "cupom fiscal". Veja na Figura 1, a tela que deverá ser implementada.

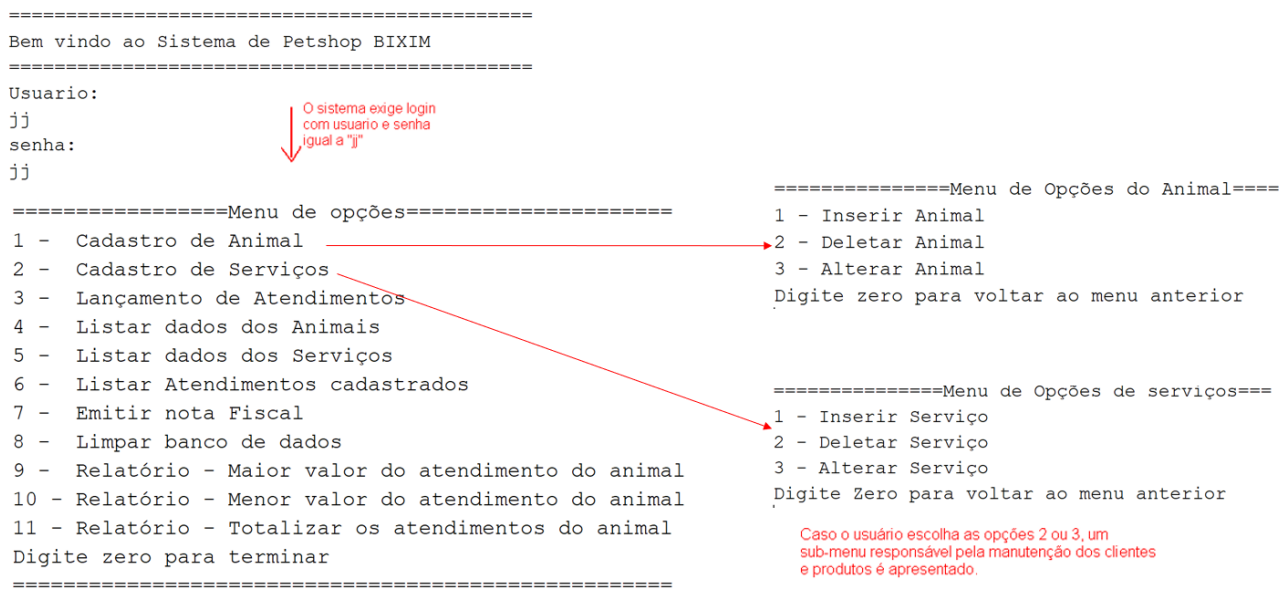

Figura 1 – Tela do sistema a ser desenvolvido.

A criação do sistema será feita seguindo os passos abaixo e deverá seguir o layout e nomes definidos neste documento.

Passo 1) Crie um projeto no Netbeans e salve-o com o nome **PetshopBIXIM**.

O primeiro passo é criar a tela de login. Esta tela deve seguir o cabeçalho da Figura 1:

a) Caso o login informado seja "jj" e a senha "jj", o usuário tem permissão de logar no sistema.
b) Caso o usuário ou a senha estejam incorretos, o sistema deve apresentar uma mensagem e terminar sua execução.

Dica: Para encerrar um programa use o comando: `System.exit(0);`

Em seguida, crie um menu principal e mostre-o para o usuário. O menu deve ter as mesmas opções contidas na Figura 1. Repare que este menu tem as opções de 1 a 11. Por exemplo, se o usuário digitar a opção 7, o sistema deve emitir uma nota fiscal, se a opção escolhida foi a quinta, o sistema deve listar os dados dos serviços. Como mostra a Figura 1, para encerrar o programa o número digitado deve ser o Zero.

OBS: caso uma opção diferente das listadas no menu seja digitada, uma mensagem de erro deve ser emitida. Veja Figura 2.

```
10
Opcao Invalida!!
```

Figura 2 – Mensagem de erro exibida ao escolher uma opção inexistente.

Passo 2) Neste passo iremos criar a estrutura do menu, ou seja, ainda não iremos implementar as funcionalidades. Para criar esta estrutura use o comando "**Case"**. Dentro das onze opções do **case** imprima um cabeçalho informando a opção selecionada. Por exemplo, caso a opção escolhida seja a terceira, o sistema deve mostrar a seguinte informação:

```
====================
CADASTRO DO SERVIÇO
====================
```

Figura 3 – Cabeçalho da opção 3.

Lembre-se que este menu deve ser exibido até que a opção escolhida seja a zero (para encerrar o programa). Dica: Use um laço de repetição.

Passo 3) Agora que o menu principal está sendo impresso e a estrutura do programa está implementada vamos codificar a primeira opção. A Figura 4 mostra que deve existir um sub-menu para inserir, alterar ou deletar um animal.

Como existem vários animais devemos armazená-los em um vetor. Crie um vetor do Tipoo animal com tamanho 10. Como o próprio nome indica ele armazenará objetos que tenham um código do animal, o nome, o endereço e a cidade.

```
===============Menu de Opções do Animal===
1 - Inserir Animal
2 - Deletar Animal
3 - Alterar Animal
Digite zero para voltar ao menu anterior
```

Figura 04 – Sub-menu de animais

A primeira opção do menu inseri os dados no vetor (Veja Figura 05). Logo, o programa deve verificar no vetor está nula (null) e inserir um objeto nessa posição. É importante verificar se o limite já foi preenchido (10 animais). Ou seja, você terá que analisar se o ainda tem alguma posição nula. Caso o vetor esteja cheio, cancele a operação e exiba uma mensagem informando ao usuário que não existe espaço suficiente.

```
==================
CADASTRO DE ANIMAL
==================
Insira seu código
100
Nome do ANIMAL
toto
Endereço do animal
rua_do_centro_80
Cidade do animal
Formiga
Dados inseridos com sucesso
```

Figura 05 – Entrada de animais

A segunda opção tem o objetivo de deletar um animal cadastrado. O sistema pede o código a ser deletado e apaga os dados no vetor (colocar "null") na posição correspondente ao objeto. Veja a Figura 06, no final do processo uma mensagem deve ser exibida informando ao usuário que os dados foram apagados

```
================
DELETAR ANIMAL
================
Informe o codigo do animal que será deletado
1
Dados deletados com sucesso
```

Figura 06 – Apagando um animal cadastrado

A terceira opção altera um animal já cadastrado. Para isto, o usuário deve informar o código a ser alterado e digitar o novo nome, endereço e cidade (Figura 07). Seria importante imprimir uma mensagem para o usuário que a alteração foi realizada com sucesso.

```
================
ALTERAR ANIMAL
================
Informe o codigo do animal que será alterado
1
Novo nome do animal:
ludinha
Novo endereço do animal:
rua_nova
Nova cidade do animal:
formiga
Dados alterados com sucesso
```

Figura 07 – Alterando um animal cadastrado

A quarta opção (digitar Zero) retorna ao menu principal.

Passo 4) A opção dois também consiste em 3 sub-opções responsáveis pela gestão dos serviços (Veja Figura 08). Como podem existir vários serviços suas informações são armazenadas em um vetor do tipo Serviço. O objeto tem o código do serviço, o nome e o valor. Também teremos um vetor com tamanho 10., se uma posição do vetor é igual a *null*, então não existe serviço naquela posição.

```
====================
CADASTRO DO SERVIÇO
====================
Insira o código do serviço
200
Descrição do serviço
tosa
Valor do serviço
30
```

Figura 08 – Submenu responsável pela gestão de serviços.

OBS: Os sub-menus tem as mesmas funcionalidades apresentadas no "Passo 3", mas toda a manipulação é feita em cima dos serviços, ao invés, de trabalhar com animais.

Passo 5) Neste passo vamos implementar a opção 3. Esta opção tem como função cadastrar os atendimentos, ou seja, os serviços realizados em algum animal. Nesta opção o usuário não poderá alterar ou deletar os itens lançados. Ele apenas deverá inserir os dados. O sistema terá a capacidade de armazenar 20 atendimentos e em cada atendimento, o usuário deverá informar os seguintes dados: Código do animal, código do serviço e a data do atendimento (nesse caso, trabalhe com o tipo Date). Deve existir também um código único para cada atendimento. Logo, crie um vetor do tipo Atendimento.

OBS: Não precisa de Sub-menus

Passo 6) Neste passo vamos implementar a opção 4. Esta opção tem como função imprimir na tela a listagem de todos os animais cadastrados. Veja na Figura 09 como devem ser mostrados os dados.

```
================================
LISTAGEM DE ANIMAIS CADASTRADOS
================================
Deseja realmente imprimir o relatório? (S/N)
S
Animal: - Código: 100 Nome: toto Endereço: rua_do_centro_80 Cidade: Formiga
Animal: - Código: 1 Nome: ludinha Endereço: rua_nova Cidade: formiga
```

Figura 9 – Impressão da listagem de animais cadastrados

Essa impressão tem dois requisitos:

1) Deve existir uma mensagem perguntando se o usuário deseja realmente imprimir o relatório.
2) Caso não haja nenhum animal cadastrado o sistema deve emitir a seguinte mensagem:

"Não existem animais cadastros no sistema!"

Passo 7) Agora vamos implementar a opção 5. Esta opção tem como função imprimir na tela a listagem de todos os serviços cadastrados. Veja na Figura 10 como devem ser mostrados os dados.

```
================================
LISTAGEM DOS SERVIÇOS CADASTRADOS
================================
Deseja realmente imprimir o relatório? (S/N)
S
Serviço - Código: 200 Descrição: tosa Valor: R$ 30.0
Serviço - Código: 300 Descrição: castracao Valor: R$ 90.0
```

Figura 10 – Impressão da listagem de serviços cadastrados

Essa impressão tem dois requisitos:

1) Deve existir uma mensagem perguntando se o usuário deseja realmente imprimir o relatório.

2) Caso não haja nenhum serviço cadastrado o sistema deve emitir a seguinte mensagem:
"Não existe serviços cadastros no sistema!"

Passo 8) Agora vamos implementar a opção 6. Esta opção tem como função imprimir na tela a listagem de todas os atendimentos lançados. Veja na Figura 11 como devem ser mostrados os dados.

```
Deseja realmente imprimir o relatório? (S/N)
S
Atendimento - Código: 1000 Cliente: 100 Produto: 50 Quantidade: 1.0 Data: 31/08/2022
Atendimento - Código: 1001 Cliente: 100 Produto: 51 Quantidade: 1.0 Data: 01/09/2022
```

Figura 11 – Impressão da listagem de atendimentos cadastrados

Essa impressão tem dois requisitos:

1) Deve existir uma mensagem perguntando se o usuário deseja realmente imprimir o relatório.
2) Caso não haja nenhum atendimento cadastrado o sistema deve emitir a seguinte mensagem:
"Não existem atendimentos lançados no sistema!"

Passo 9) A opção 7 visa simular a impressão de uma nota fiscal. Ou seja, devem ser impressos os dados do animal escolhido, juntamente, com os atendimentos realizados pela empresa e um totalizador somando os valores. O layout que deve ser seguido pode ser visto na Figura 12.

```
============================
NOTA FISCAL
============================
Nome: toto
Endereço: rua_do_centro
Cidade: formiga
============================
  ===ATENDIMENTOS===
============================
Serviço: tosa        Valor: 20.0
Serviço: castracao        Valor: 90.0

============================
Total: R$ 110.0
============================
```

Figura 12

A impressão da Nota fiscal tem três requisitos:

1) Todos os dados do animal devem ser exibidos, senão uma mensagem deve ser exibida informando a obrigatoriedade dos dados e, por fim, a operação deve ser abortada.
2) Deve existir, pelo menos, um atendimento com valor e descrição informados, caso contrário, uma mensagem deve ser exibida informando a obrigatoriedade dos dados e, por fim, a operação deve ser abortada.
3) Somente serviços com descrição e valor preenchidos devem ser impressos e totalizados.

Passo 10) A opção 8 limpa (deleta) todos os dados cadastrados. É importante que uma mensagem de confirmação seja exibida no início do processo. Ao final, outra mensagem informando que os dados foram deletados com sucesso também deve ser exibida. Veja figura abaixo.

```
====================
DELETAR BANCO DE DADOS
====================
Deseja Realmente APAGAR todos os dados cadastrados (S\N):

S
Banco de dados deletado com sucesso
```

Figura 13

Passo 11) A opção 9 informa ao usuário qual o atendimento cadastrado com o maior valor para um determinado animal. A Figura 14 mostra como deve ser impressa a mensagem ao usuário. É importante informar ao usuário através de uma mensagem, caso não exista atendimentos cadastrados.

```
Insira o código do animal:
100
RELATÓRIO - ATENDIMENTO DE MAIOR VALOR: castracao Valor: R$ 90.0
```

Figura 14

Passo 12) A opção 10 informa ao usuário qual o atendimento cadastrado com o menor valor para um determinado animal. A Figura 15 mostra como deve ser impressa a mensagem ao usuário. É importante informar ao usuário através de uma mensagem, caso não exista atendimentos cadastrados.

```
Insira o código do animal:
100
RELATÓRIO - ATENDIMENTO DE MENOR VALOR: tosa Valor: R$ 20.0
```

Figura 15

Passo 13) A opção 11 informa ao usuário o total de atendimentos (cadastrados). A Figura 16 mostra como deve ser impressa a mensagem ao usuário. É importante informar ao usuário através de uma mensagem, caso não exista atendimentos cadastrados.

```
11
Insira o código do animal:
100
RELATÓRIO - O TOTAL DOS ATENDIMENTOS DO ANIMAL É: R$ 90.0
```

Figura 16

Passo14)

Atrelado a tudo que foi dito, deve existir também uma nova opção no menu: "**12 - Relatório - Atendimento entre um período**". A Figura 16 mostra a execução dessa opção:

```
12
Insira a data inicial (dd/mm/aaaa):
01/01/2022
Insira a data final (dd/mm/aaaa):
31/12/2022
RELATÓRIO - ATENDIMENTOS NO PERÍODO:
Atendimento: 100 Suzi - Banho R$ 30.0
Atendimento: 1 Suzi - Banho R$ 30.0
```

Figura 16 – Relatório de atendimentos por data

Atenção, se não tiver atendimentos na data informada, dê uma mensagem informando ao usuário.

Passo15) 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 17 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.

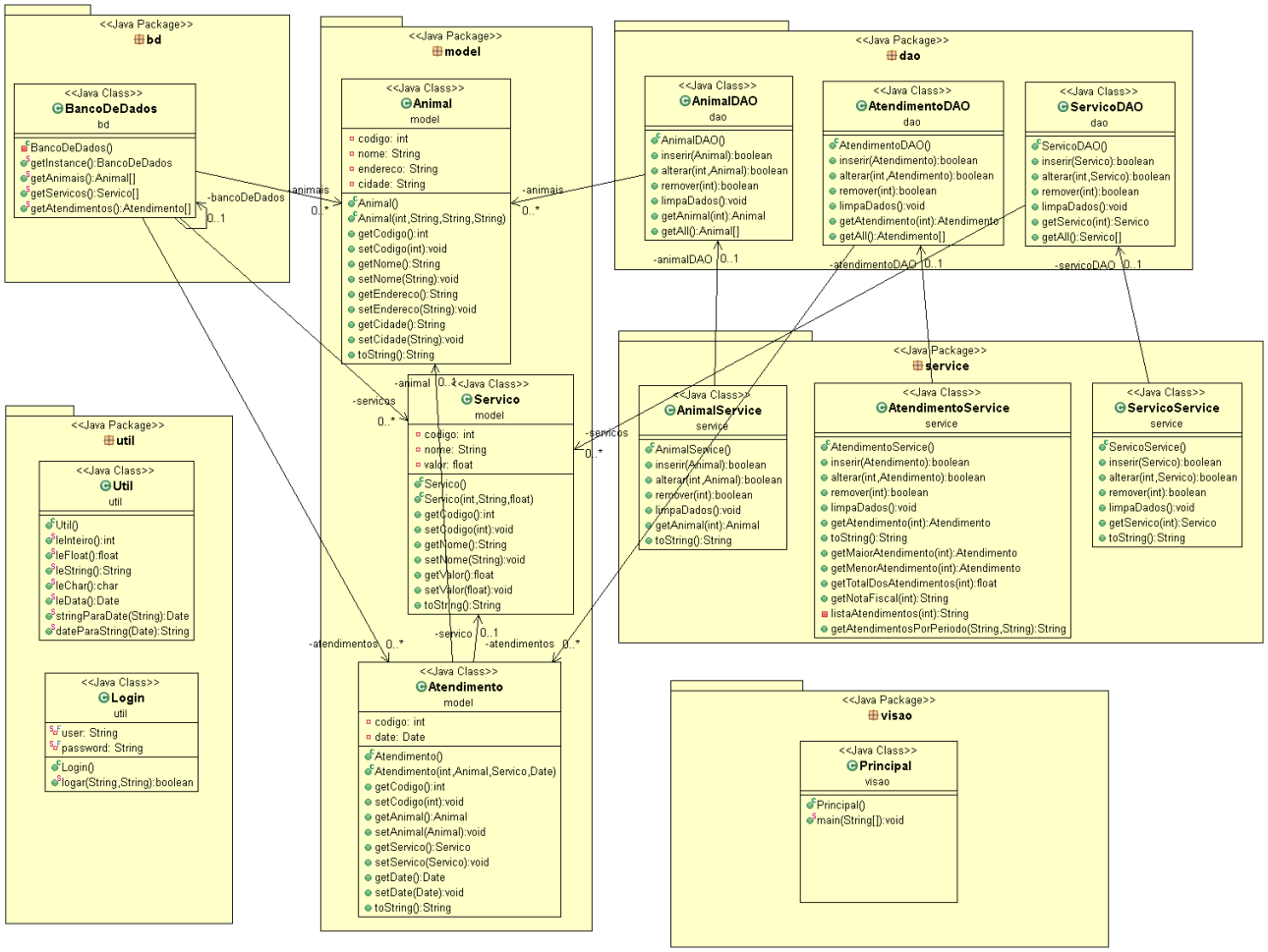

Figura 17 – Diagrama de Classe do projeto

Subpasso 15a) Crie todos os pacotes que o diagrama indica

Subpasso 15b) Crie as três classes do pacote “model”, pois várias outras classes dependem delas.

Subpasso 15c) 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 15d) As classes do pacote “dao” são responsáveis por manipular os vetores, ou seja, seus métodos inserem, alteram, removem dados nos vetores, existem métodos também para retornarem um ou todos os dados dos arrays. O nome “dao” vem do inglês, Data Access Object, ou seja, objeto que acessam os dados, os quais nesse caso, estão 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 15e) 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 15f) 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”.

Passo14) Faça testes e verifique se o desenvolvimento está correto.

Passo15) 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;
    }

    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;
    }


}
```