1. Exercícios
2. **Projeto Final**
3. Cada Aluno fará seu projeto final, ou seja o projeto final é **individual**.
4. Descreva a Função do Sistema:
5. **Instruções para confeccionar o projeto.**
6. Crie um novo Projeto Java para o Projeto final.
7. Crie os pacotes :
8. apresentação (console ou telas)\*
9. entidade ( ou controle)
10. Descreva a função do seu programa.
11. Faça a lista de tarefas do programa.
12. Faça o esboço das telas.
13. Faça o diagrama de classes.
14. **O Projeto final deverá ter:**
15. **Itens obrigatórios (para tirar 7.0) ( pelo menos um exemplo)**
16. Correta Implementação de classes com atributos e métodos;
17. Uso de vetor;
18. Uso de estrutura de repetição;
19. Uso dos métodos “get” e “set”;
20. Uso de modificadores de acesso nos atributos e métodos (Todos atributos como private\*);
21. Herança (com implementação de classes);
22. Polimorfismo (pelo menos um exemplo);
23. Manipulação de arquivos;
24. Criação,
25. Exclusão;
26. Salvamento;
27. Edição,
28. Busca,
29. Transferência
30. Interface (pelo menos um exemplo);
31. Tratamento de exceções (ty/catch);
32. Tratamento de erros;
33. Conversões de valores (texto para número e de número para texto\*);
34. Usar a correta forma e escrita de declaração de atributos, métodos e classes;
35. Confeccionar o Diagrama de Classes.
36. Recomendado fazer o sistema com dimensões de 1200 e 800 pixel ou maior
37. **Itens complementares ( para tirar 10.0)**
38. Override (pelo menos um exemplo)
39. Pelo menos uma implementação do pacote Collection
40. Usar pacote Swing (opcional)
41. Confeccionar Diagrama de telas (eventos)
42. **Observação**
43. O sistema poderá ser feito em modo “Tela Preta” ou através de telas gráfica, usando o pacote swing.
44. **Envio do Dever de Casa**
45. O aluno receberá uma relação de exercícios a serem entregues até a próxima aula.
46. Deveres de casa entregues fora do prazo terão sua nota reduzia, em média 1.0 ponto por aula atrasada.
47. O aluno irá ler cada enunciado e realizará a tarefa no eclipse. Depois de realizado a tarefa, copie e cole o código fonte do eclipse e cole abaixo do respectivo enunciado.
48. Use o eclipse com fundo branco e fonte tamanho12. Ao colar mantenha a formatação original, ou seja, mantenha as cores que o eclipse fornece.
49. Use o word ou o openoffice para editar os arquivos de dever de casa. Obs: o word online não funciona para manter a formatação do eclipse.
50. Apos fazer todas as tarefas e colar no arquivo do dever de casa, envie este arquivo no mystet.
51. Obs: O mystet só permite o envio de um arquivo no upload do dever de casa.
52. **Função do seu programa:**

Programa de controle de ativos de TI.

O programa faz interação com o usuário por meio de um menu de opções.

Para cada opção que o usuário escolher vai ser executado uma tarefa.

1. **Lista de tarefas do programa:**
2. **Cadastrar ativo**: o programa solicita informações necessárias para o cadastro de novos ativos, cria um objeto da classe *“Asset”* e salva no arquivo *“CadastroAsset.txt”*.
3. **Localizar ativo**: o programa faz a busca no arquivo *“CadastroAsset.txt”* conforme o dado informado na busca. Caso encontre, exibe as informações do(s) ativo(s).
4. **Saída de ativo**: realiza a saída de um ativo do estoque, sendo feito por meio de modificação do “status”, “usuário”, “departamento” e “data”.
5. **Entrada de ativo**: realiza a entrada de um ativo do estoque, sendo feito por meio de modificação do “status”, “usuário”, “departamento” e “data”.
6. **Listagem geral ativos**: o programa faz a leitura do arquivo *“CadastroAsset.txt”* e retorna as informações de todos os ativos cadastrados no console(tela).
7. **Listagem por parâmetro**: o programa faz a leitura do arquivo *“CadastroAsset.txt”* conforme o dado informado na buscae retorna à informação do ativo cadastrado no console(tela).
8. **Resumo ativos – quantitativo:** o programa lê o arquivo *“CadastroAsset.txt”* e exibe a quantidade de ativos por tipo – Notebook, macbook, desktop etc.
9. **Resumo ativos – contábil:**  o programa lê o arquivo *“CadastroAsset.txt”* e exibe o montante dos ativos por tipo – Notebook, macbook, desktop etc.
10. **Excluir ativos obsoletos**: o programa lê o arquivo *“CadastroAsset.txt,* conforme o dado informado na busca e remove o ativo com status “OBSOLETO” caso encontre.

* **package Entidades;**
  + public class AssetRegister
    - Atributos, Modificadores de acesso, Construtores, Getters e setters, Método
      * public boolean *verificarArquivo ()*
      * public boolean *cadastrarAsset()*
  + public class Asset extends AssetRegister
    - Atributos, Modificadores de acesso, Construtores, Getters e setters, Método
      * public boolean *atribuirAsset()*
      * public boolean *localizarAsset()*
      * public boolean *saidaAsset()*
      * public boolean *entradaAsset()*
      * public boolean *listarAsset()*
      * public boolean *listarAsset()*
      * public boolean *quantitativoAsset()*
      * public boolean *contabilAsset()*
      * public boolean *excluirAsset()*
* **package Apresentacao;**
  + public class Inicio
    - public static void main(String[] args)

**package** apresentacao;

**import** java.io.File;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**import** java.util.Scanner;

**import** entidades.Asset;

**public** **class** Inicio {

/\*

\* Programa de controle de ativos de TI. O programa faz interação com o usuário

\* por meio de um menu de opções. Para cada opção que o usuário escolher vai ser

\* executado uma tarefa.

\*/

**public** **static** **void** main(String[] args) {

// Diretório

File d = **new** File(".");

// Obter caminho absoluto do diretório

String caminhoDiretorio = d.getAbsolutePath();

// Arquivo

File f = **new** File("");

// Obter caminho absoluto do arquivo

String caminho = f.getAbsolutePath() + "\\CadastroAsset.txt";

String op;

// Objeto para ler dados do teclado

Scanner ler = **new** Scanner(System.***in***);

// Menu de opções para o usuário

**do** {

System.***out***.println();

System.***out***.println("╔──────────────────────────────────╗");

System.***out***.println("│ ◄│▒│ MENU │▒│► │");

System.***out***.println("│──────────────────────────────────│");

System.***out***.println("│ [1] CADASTRAR ATIVO │");

System.***out***.println("│ [2] LOCALIZAR ATIVO │");

System.***out***.println("│ [3] SAÍDA DE ATIVO │");

System.***out***.println("│ [4] ENTRADA DE ATIVO │");

System.***out***.println("│ [5] LISTAGEM GERAL ATIVOS │");

System.***out***.println("│ [6] LISTAGEM POR PARÂMETROS │");

System.***out***.println("│ [7] RESUMO ATIVOS - QUANTITATIVO │");

System.***out***.println("│ [8] RESUMO ATIVOS - CONTÁBIL │");

System.***out***.println("│ [9] EXCLUIR ATIVO OBSOLETO │");

System.***out***.println("│ [s] SAIR │");

System.***out***.println("└──────────────────────────────────┘");

System.***out***.println(" │ ");

System.***out***.print(" └■ Opção: ");

// Lê a opção escolhida pelo usuário

op = ler.nextLine();

**switch** (op) {

/\*

\* [1] Cadastrar ativo: o programa solicita informações necessárias para o

\* cadastro de novos ativos, cria um objeto da classe “Asset” e salva no arquivo

\* “CadastroAsset.txt”.

\*/

**case** "1": {

**try** {

// Instanciando a classe Asset

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ SERVICE TAG: ");

String serviceTag = ler.nextLine().toUpperCase().trim();

// Validação SERVICE TAG

**if** (serviceTag.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ SERVICE TAG não informado");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ TIPO:");

System.***out***.println(" └► [1] NOTEBOOK");

System.***out***.println(" └► [2] MACBOOK");

System.***out***.println(" └► [3] IMAC");

System.***out***.println(" └► [4] DESKTOP");

System.***out***.println(" └► [5] AIO");

System.***out***.println(" └► [6] MONITOR");

System.***out***.print(" └■ Opção: ");

String tipo = ler.nextLine().toUpperCase().trim();

// Validação TIPO

**if** (tipo.equals("1")) {

tipo = "NOTEBOOK";

} **else** **if** (tipo.equals("2")) {

tipo = "MACBOOK";

} **else** **if** (tipo.equals("3")) {

tipo = "IMAC";

} **else** **if** (tipo.equals("4")) {

tipo = "DESKTOP";

} **else** **if** (tipo.equals("5")) {

tipo = "AIO";

} **else** **if** (tipo.equals("6")) {

tipo = "MONITOR";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ TIPO inválido!");

**break**;// Caso else retorna o menu, não deixa proseguir o cadastro

}

System.***out***.println(" │ ");

System.***out***.println(" └■ PERFIL:");

System.***out***.println(" └► [1] ADM");

System.***out***.println(" └► [2] DEV");

System.***out***.println(" └► [3] DIR");

System.***out***.print(" └■ Opção: ");

String perfil = ler.nextLine().toUpperCase().trim();

// Validação PERFIL

**if** (perfil.equals("1")) {

perfil = "ADM";

} **else** **if** (perfil.equals("2")) {

perfil = "DEV";

} **else** **if** (perfil.equals("3")) {

perfil = "DIR";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ PERFIL inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ VALOR (apenas números): ");

**double** valor = Double.*parseDouble*(ler.nextLine().toUpperCase().trim());

System.***out***.println(" │ ");

System.***out***.print(" └■ NOTA FISCAL (apenas números): ");

**int** notaFiscal = Integer.*parseInt*(ler.nextLine().toUpperCase().trim());

System.***out***.println(" │ ");

System.***out***.println(" └■ PROPRIETÁRIO:");

System.***out***.println(" └► [1] BIT");

System.***out***.println(" └► [2] AMER");

System.***out***.print(" └■ Opção: ");

String proprietario = ler.nextLine().toUpperCase().trim();

// Validação PROPRIETÁRIO

**if** (proprietario.equals("1")) {

proprietario = "BIT";

} **else** **if** (proprietario.equals("2")) {

proprietario = "AMER";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ PROPRIETÁRIO inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ LOCAL:");

System.***out***.println(" └► [1] SEDE");

System.***out***.println(" └► [2] BIT RJ");

System.***out***.println(" └► [3] BIT SP");

System.***out***.print(" └■ Opção: ");

String local = ler.nextLine().toUpperCase().trim();

// Validação LOCAL

**if** (local.equals("1")) {

local = "SEDE";

} **else** **if** (local.equals("2")) {

local = "BIT RJ";

} **else** **if** (local.equals("3")) {

local = "BIT SP";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ LOCAL inválido!");

System.***out***.println();

**break**;

}

// Data do cadastro apenas

System.***out***.println(" │ ");

System.***out***.print(" └■ DATA CADASTRO: ");

LocalDate dataCadastro = LocalDate.*now*();

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");

System.***out***.println(dataCadastro.format(fmt));

System.***out***.println(" │ ");

System.***out***.println(" └■ STATUS:");

System.***out***.println(" └► [1] ESTOQUE");

System.***out***.print(" └■ Opção: ");

String status = ler.nextLine().toUpperCase().trim();

// Validação STATUS

**if** (status.equals("1")) {

status = "ESTOQUE";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ STATUS inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ SISTEMA OPERACIONAL:");

System.***out***.println(" └► [1] WINDOWNS");

System.***out***.println(" └► [2] LINUX");

System.***out***.println(" └► [3] MAC OS");

System.***out***.println(" └► [4] SEM SO");

System.***out***.print(" └■ Opção: ");

String sistemaOperacional = ler.nextLine().toUpperCase().trim();

// Validação SISTEMA OPERACIONAL

**if** (sistemaOperacional.equals("1")) {

sistemaOperacional = "WINDOWNS";

} **else** **if** (sistemaOperacional.equals("2")) {

sistemaOperacional = "LINUX";

} **else** **if** (sistemaOperacional.equals("3")) {

sistemaOperacional = "MAC OS";

} **else** **if** (sistemaOperacional.equals("4")) {

sistemaOperacional = "SEM SO";

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ SISTEMA OPERACIONAL inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ USUÁRIO (proibido caracteres especias): ");

String usuario = ler.nextLine().toUpperCase().trim();

// Validação USUÁRIO

**if** (usuario.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO não informado");

**break**;

} **else** **if** (!usuario.matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO contém caracteres inválidos!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ DEPARTAMENTO (proibido caracteres especias): ");

String departamento = ler.nextLine().toUpperCase().trim();

// Validação DEPARTAMENTO

**if** (departamento.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ DEPARTAMENTO não informado");

**break**;

} **else** **if** (!departamento.matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ DEPARTAMENTO contém caracteres inválidos!");

**break**;

}

// Data movimento

System.***out***.println(" │ ");

System.***out***.print(" └■ DATA MOVIMENTO:");

LocalDate dataMov = LocalDate.*now*();

System.***out***.println(dataMov.format(fmt));

// Chamando o método que faz o cadastro completo do Asset

a.atribuirUsuario(caminhoDiretorio, caminho, serviceTag, tipo, perfil, valor, notaFiscal,

proprietario, local, dataCadastro, status, sistemaOperacional, usuario, departamento,

dataMov);

} **catch** (NumberFormatException e) {// Informa que tentou converter uma string para um formato numérico

System.***err***.println(" └■ Erro: " + e.getMessage());

}

**break**;

}

/\*

\* [2] Localizar ativo: o programa faz a busca no arquivo “CadastroAsset.txt”

\* conforme o dado informado na busca. Caso encontre, exibe as informações do(s)

\* ativo(s).

\*/

**case** "2": {

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ LOCALIZAR:");

String localizar = ler.nextLine().toUpperCase().trim();

// Chamando o método que localiza

a.localizar(caminhoDiretorio, caminho, localizar);

**break**;

}

/\*

\* [3] Saída de ativo: realiza a saída de um ativo do estoque, sendo feito por

\* meio de modificação do “status”, “usuário”, “departamento” e “dataMov”.

\*/

**case** "3": {

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ LOCALIZAR:");

String stSaida = ler.nextLine().toUpperCase().trim();

// Validação Service Tag

**if** (stSaida.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ SERVICE TAG não informado");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ STATUS:");

System.***out***.println(" └► [1] EM USO");

System.***out***.println(" └► [2] SUPORTE TÉCNICO");

System.***out***.print(" └■ Opção: ");

a.setStatus(ler.nextLine().toUpperCase().trim());

// Validação STATUS utilizando o get e set

**if** (a.getStatus().equals("1")) {

a.setStatus("EM USO");

} **else** **if** (a.getStatus().equals("2")) {

a.setStatus("SUPORTE TÉCNICO");

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ STATUS inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ SISTEMA OPERACIONAL:");

System.***out***.println(" └► [1] WINDOWNS");

System.***out***.println(" └► [2] LINUX");

System.***out***.println(" └► [3] MAC OS");

System.***out***.println(" └► [4] SEM SO");

System.***out***.print(" └■ Opção: ");

a.setSistemaOperacional(ler.nextLine().toUpperCase().trim());

// Validação SISTEMA OPERACIONAL utilizando o get e set

**if** (a.getSistemaOperacional().equals("1")) {

a.setSistemaOperacional("WINDOWNS");

} **else** **if** (a.getSistemaOperacional().equals("2")) {

a.setSistemaOperacional("LINUX");

} **else** **if** (a.getSistemaOperacional().equals("3")) {

a.setSistemaOperacional("MAC OS");

} **else** **if** (a.getSistemaOperacional().equals("4")) {

a.setSistemaOperacional("SEM SO");

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ SISTEMA OPERACIONAL inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ USUÁRIO: ");

a.setUsuario(ler.nextLine().toUpperCase().trim());

// Validação USUÁRIO

**if** (a.getUsuario().isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO não informado");

**break**;

} **else** **if** (!a.getUsuario().matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO contém caracteres inválidos!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ DEPARTAMENTO: ");

a.setDepartamento(ler.nextLine().toUpperCase().trim());

// Validação DEPARTAMENTO

**if** (a.getDepartamento().isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO não informado");

**break**;

} **else** **if** (!a.getDepartamento().matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO contém caracteres inválidos!");

**break**;

}

// Modificando a data de movimento para a atual

System.***out***.println(" │ ");

System.***out***.print(" └■ DATA MOVIMENTO:");

LocalDate dataMov = LocalDate.*now*();

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");

System.***out***.println(dataMov.format(fmt));

// Chamando o método que faz a saída de um Asset

a.saidaAsset(caminhoDiretorio, caminho, stSaida);

**break**;

}

/\*

\* [4] Entrada de ativo: realiza a entrada de um ativo do estoque, sendo feito

\* por meio de modificação do “status”, “usuário”, “departamento” e “dataMov”.

\*/

**case** "4": {

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ LOCALIZAR:");

String stEntrada = ler.nextLine().toUpperCase().trim();

**if** (stEntrada.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ Service Tag não informado");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ STATUS:");

System.***out***.println(" └► [1] ESTOQUE");

System.***out***.println(" └► [2] DEFEITO");

System.***out***.println(" └► [3] OBSOLETO");

System.***out***.print(" └■ Opção: ");

a.setStatus(ler.nextLine().toUpperCase().trim());

**if** (a.getStatus().equals("1")) {

a.setStatus("ESTOQUE");

} **else** **if** (a.getStatus().equals("2")) {

a.setStatus("DEFEITO");

} **else** **if** (a.getStatus().equals("3")) {

a.setStatus("OBSOLETO");

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ STATUS inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.println(" └■ SISTEMA OPERACIONAL:");

System.***out***.println(" └► [1] WINDOWNS");

System.***out***.println(" └► [2] LINUX");

System.***out***.println(" └► [3] MAC OS");

System.***out***.println(" └► [4] SEM SO");

System.***out***.print(" └■ Opção: ");

a.setSistemaOperacional(ler.nextLine().toUpperCase().trim());

**if** (a.getSistemaOperacional().equals("1")) {

a.setSistemaOperacional("WINDOWNS");

} **else** **if** (a.getSistemaOperacional().equals("2")) {

a.setSistemaOperacional("LINUX");

} **else** **if** (a.getSistemaOperacional().equals("3")) {

a.setSistemaOperacional("MAC OS");

} **else** **if** (a.getSistemaOperacional().equals("4")) {

a.setSistemaOperacional("SEM SO");

} **else** {

System.***out***.println(" │");

System.***out***.println(" └█ SISTEMA OPERACIONAL inválido!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ USUÁRIO: ");

a.setUsuario(ler.nextLine().toUpperCase().trim());

**if** (a.getUsuario().isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO não informado");

**break**;

} **else** **if** (!a.getUsuario().matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO contém caracteres inválidos!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ DEPARTAMENTO: ");

a.setDepartamento(ler.nextLine().toUpperCase().trim());

**if** (a.getDepartamento().isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO não informado");

**break**;

} **else** **if** (!a.getDepartamento().matches("[a-zA-Z\s]+")) {

System.***out***.println(" │");

System.***out***.println(" └█ USUÁRIO contém caracteres inválidos!");

**break**;

}

System.***out***.println(" │ ");

System.***out***.print(" └■ DATA MOVIMENTO:");

LocalDate dataMov = LocalDate.*now*();

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");

System.***out***.println(dataMov.format(fmt));

// Chamando o método que faz a entrada de um Asset

a.entradaAsset(caminhoDiretorio, caminho, stEntrada);

**break**;

}

/\*

\* [5] Listagem geral ativos: o programa faz a leitura do arquivo

\* “CadastroAsset.txt” e retorna as informações de todos os ativos cadastrados.

\*/

**case** "5": {

Asset a = **new** Asset();

a.listarAsset(caminhoDiretorio, caminho);

**break**;

}

/\*

\* [6] Listagem por parâmetro: o programa faz a leitura do arquivo

\* “CadastroAsset.txt” conforme o dado informado na busca e retorna à informação

\* do ativo cadastrado no console(tela).

\*/

**case** "6": {

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ LOCALIZAR: ");

String localizar = ler.nextLine().toUpperCase().trim();

a.listarAsset(caminhoDiretorio, caminho, localizar);

**break**;

}

/\*

\* [7] Resumo ativos – quantitativo: o programa lê o arquivo “CadastroAsset.txt”

\* e exibe a quantidade de ativos por tipo – Notebook, macbook, desktop etc.

\*/

**case** "7": {

Asset a = **new** Asset();

a.quantitativoAsset(caminhoDiretorio, caminho);

**break**;

}

/\*

\* [8] Resumo ativos – contábil: o programa lê o arquivo “CadastroAsset.txt” e

\* exibe o montante dos ativos por tipo – Notebook, macbook, desktop etc.

\*/

**case** "8": {

Asset a = **new** Asset();

a.contabilAsset(caminhoDiretorio, caminho);

**break**;

}

/\*

\* [9] Excluir ativos obsoletos: o programa lê o arquivo “CadastroAsset.txt,

\* conforme o dado informado na busca e remove o ativo com status “OBSOLETO”

\* caso encontre.

\*/

**case** "9": {

Asset a = **new** Asset();

System.***out***.println(" │ ");

System.***out***.print(" └■ LOCALIZAR: ");

String excluirSt = ler.nextLine().toUpperCase().trim();

a.excluirAsset(caminhoDiretorio, caminho, excluirSt);

**break**;

}

/\*

\* [s] Sair do programa

\*/

**case** "s":

**case** "S": {

System.***out***.println(" │ ");

System.***out***.println(" └█ SAIR... ");

System.***out***.println(" │ ");

System.***out***.println(" │╔─────────────────────────────────────────────╗");

System.***out***.println(" └│ ■│▒│ FIM DO PROGRAMA │▒│■ │");

System.***out***.println(" └─────────────────────────────────────────────┘");

**break**;

}

**default**:

**break**;

}

// Repete o menu enquanto o usuário não escolher sair

} **while** (!op.equalsIgnoreCase("s"));

ler.close();// Fechamento classe Scanner

}

}

**package** entidades;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**public** **class** AssetRegister {

// Declaração de atributos

**private** String serviceTag;

**private** String tipo;

**private** String perfil;

**private** **double** valor;

**private** **int** notaFiscal;

**private** String proprietario;

**private** String local;

**private** LocalDate dataCadastro;

// Construtor vazio

**public** AssetRegister() {

**super**();

}

// Construtor com parâmetros

**public** AssetRegister(String serviceTag, String tipo, String perfil, **double** valor, **int** notaFiscal,

String proprietario, String local, LocalDate dataCadastro) {

**super**();

**this**.serviceTag = serviceTag;

**this**.tipo = tipo;

**this**.perfil = perfil;

**this**.valor = valor;

**this**.notaFiscal = notaFiscal;

**this**.proprietario = proprietario;

**this**.local = local;

**this**.dataCadastro = dataCadastro;

}

// Getters e Setters

**public** String getLocal() {

**return** local;

}

**public** **void** setLocal(String local) {

**this**.local = local;

}

**public** String getServiceTag() {

**return** serviceTag;

}

**public** String getTipo() {

**return** tipo;

}

**public** String getPerfil() {

**return** perfil;

}

**public** **double** getValor() {

**return** valor;

}

**public** **int** getNotaFiscal() {

**return** notaFiscal;

}

**public** String getProprietario() {

**return** proprietario;

}

**public** LocalDate getDataCadastro() {

**return** dataCadastro;

}

/\*

\* Verifica se os caminhos existem, caso não, cria o diretório e o arquivo

\*/

**public** **boolean** verificarArquivo(String caminhoDiretorio, String caminho) {

File d = **new** File(caminhoDiretorio);

**if** (d.exists()) {

**try** {

File f = **new** File(caminho);

**if** (f.createNewFile()) {

}

} **catch** (Exception e) {

System.***err***.println("Erro ao criar o arquivo. Erro: " + e);

}

} **else** {

**if** (d.mkdirs()) {

**try** {

File f = **new** File(caminho);

**if** (f.createNewFile()) {

}

} **catch** (Exception e) {

System.***err***.println("Erro ao criar o arquivo. Erro: " + e);

}

} **else** {

System.***err***.println("Erro ao criar os diretórios");

}

}

**return** **false**;

}

/\*

\* Cadastrar um novo ativo e enviar as informações para o arquivo

\* "CadastroAsset.txt".

\*/

**public** **boolean** cadastrarAsset(String caminhoDiretorio, String caminho, String serviceTag, String tipo,

String perfil, **double** valor, **int** notaFiscal, String proprietario, String local, LocalDate dataCadastro) {

**try** {// Inicio bloco try-catch

// Chamando método para verificar se diretório ou arquivo existem

verificarArquivo(caminhoDiretorio, caminho);

// Abrindo o arquivo em modo leitura

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

// Abrindo o arquivo em modo escrita acrescentando informação

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(caminho, **true**));

**boolean** cadastrado = **false**;

// Laço de repetição para saber se o vet[0] já existe no arquivo

**while** (br.ready()) {// Enquanto tiver linha para ler, retorna true

String linha = br.readLine();

String vet[] = linha.split("#");

**if** (vet.length >= 2 && vet[0].equalsIgnoreCase(serviceTag)) {

cadastrado = **true**;

}

}

br.close(); // Fechando o arquivo

// Caso true, não cadastrar

**if** (cadastrado) {

System.***out***.println(" │");

System.***out***.println(" └█ Service Tag já cadastrado!");

// Caso false, cadastra novo ativo

} **else** {

dataCadastro = LocalDate.*now*();// Data atual

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");// Formatação data

// Escrevendo no arquivo

bw.write(serviceTag + "#" + tipo + "#" + perfil + "#" + valor + "#" + notaFiscal + "#" + proprietario

+ "#" + local + "#" + dataCadastro.format(fmt));

bw.close();

}

} **catch** (

Exception e) {

e.printStackTrace();// Imprime todos os erros encontrado

}

**return** **false**;

}

}

**package** entidades;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**import** java.util.ArrayList;

// Herança

**public** **class** Asset **extends** AssetRegister {

// Declaração de atributos

**private** String status;

**private** String sistemaOperacional;

**private** String usuario;

**private** String departamento;

**private** LocalDate dataMov;

// Construtor vazio

**public** Asset() {

**super**();

}

// Construtor com parâmetros

**public** Asset(String serviceTag, String tipo, String modelo, String perfil, **double** valor, **int** notaFiscal,

String proprietario, String local, LocalDate dataCadastro, String status, String sistemaOperacional,

String usuario, String departamento, LocalDate dataMov) {

// Chamando o construtor da classe pai para inicializar atributos

**super**(serviceTag, tipo, perfil, valor, notaFiscal, proprietario, local, dataCadastro);

**this**.status = status;

**this**.sistemaOperacional = sistemaOperacional;

**this**.usuario = usuario;

**this**.departamento = departamento;

**this**.dataMov = dataMov;

}

// Getters e Setters

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** String getSistemaOperacional() {

**return** sistemaOperacional;

}

**public** **void** setSistemaOperacional(String sistemaOperacional) {

**this**.sistemaOperacional = sistemaOperacional;

}

**public** String getUsuario() {

**return** usuario;

}

**public** **void** setUsuario(String usuario) {

**this**.usuario = usuario;

}

**public** String getDepartamento() {

**return** departamento;

}

**public** **void** setDepartamento(String departamento) {

**this**.departamento = departamento;

}

**public** LocalDate getDataMov() {

**return** dataMov;

}

**public** **void** setDataMov(LocalDate dataMov) {

**this**.dataMov = dataMov;

}

/\*

\* Atribuir um usuário a um asset e cadastrar o asset e enviar as informações

\* para o arquivo "CadastroAsset.txt".

\*/

**public** **boolean** atribuirUsuario(String caminhoDiretorio, String caminho, String serviceTag, String tipo,

String perfil, **double** valor, **int** notaFiscal, String proprietario, String local, LocalDate dataCadastro,

String status, String sistemaOperacional, String usuario, String departamento, LocalDate dataMov) {

**try** {// Inicio bloco try-catch

// Chamando método para verificar se diretório ou arquivo existem

verificarArquivo(caminhoDiretorio, caminho);

// Abrindo o arquivo em modo leitura

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

// Abrindo o arquivo em modo escrita acrescentando informação

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(caminho, **true**));

**boolean** cadastrado = **false**;

// Laço de repetição para saber se o vet[0] já existe no arquivo

**while** (br.ready()) {// Enquanto tiver linha para ler, retorna true

String linha = br.readLine();

String vet[] = linha.split("#");

**if** (vet.length >= 2 && vet[0].equalsIgnoreCase(serviceTag)) {

cadastrado = **true**;

}

}

br.close(); // Fechando o arquivo

// Caso true, não cadastrar

**if** (cadastrado) {

System.***out***.println(" │");

System.***out***.println(" └█ Service Tag já cadastrado!");

// Caso false, cadastra novo ativo

} **else** {

// Override, modificando o método pai "cadastrarAsset"

**super**.cadastrarAsset(caminhoDiretorio, caminho, serviceTag, tipo, perfil, valor, notaFiscal,

proprietario, local, dataCadastro);

dataCadastro = LocalDate.*now*();// Data atual

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");// Formatação data

// Escrevendo no arquivo

bw.write("#" + status + "#" + sistemaOperacional + "#" + usuario + "#" + departamento + "#"

+ dataMov.format(fmt));

bw.newLine();// Criar nova linha (pular linha)

System.***out***.println(" │");

System.***out***.println(" └█ Asset cadastrado com sucesso!");

bw.close();

}

} **catch** (Exception e) {

e.printStackTrace();// Imprime todos os erros encontrado

}

**return** **false**;

}

/\*

\* Realizar uma busca (localizar)

\*/

**public** **boolean** localizar(String caminhoDiretorio, String caminho, String localizar) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

**if** (localizar.isEmpty()) {// Retorna um boolean informando se está vazia ou não

System.***out***.println(" │");

System.***out***.println(" └█ Campo em branco!");

} **else** {

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

**boolean** naoLocalizado = **true**;

System.***out***.println("\n\t\t\t\t\t\t\t\t\t◄│▒│ LISTAGEM DE ATIVOS │▒│► ");

System.***out***.println(

"╔─►─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╗");

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

"SERVICE TAG", "TIPO", "PERFIL", "VALOR", "NOTA FISCAL", "PROPRIETÁRIO", "LOCAL",

"DATA CADASTRO", "STATUS", "SO", "USUÁRIO", "DEPARTAMENTO", "DATA MOVIMENTO");

System.***out***.println();

System.***out***.println(

"╚─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────◄─╝");

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

**for** (String indice : vet) {

**if** (indice.contains(localizar)) {

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

vet[0], vet[1], vet[2], vet[3], vet[4], vet[5], vet[6], vet[7], vet[8], vet[9],

vet[10], vet[11], vet[12]);

System.***out***.println();

naoLocalizado = **false**;

**break**;

}

}

}

System.***out***.println(

"╚───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╝");

**if** (naoLocalizado) {

System.***out***.println(" │");

System.***out***.println(" └█ Não encontramos resultado para sua pesquisa. <" + localizar + ">");

}

br.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Realiza a saída de um ativo do inventário

\*/

**public** **boolean** saidaAsset(String caminhoDiretorio, String caminho, String stSaida) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

**if** (stSaida.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ Campo em branco!");

} **else** {

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

// Escrevendo no arquivo temporário

BufferedWriter bw = **new** BufferedWriter(

**new** FileWriter(caminho.replace("\\CadastroAsset.txt", "\\tmp.txt")));

dataMov = LocalDate.*now*();

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");// Tratando a linha, fazendo a separação das Strings pelo caracter #

// Editando linha

**if** (vet[0].equalsIgnoreCase(stSaida)) {

bw.write(vet[0] + "#" + vet[1] + "#" + vet[2] + "#" + vet[3] + "#" + vet[4] + "#" + vet[5] + "#"

+ vet[6] + "#" + vet[7] + "#" + status + "#" + sistemaOperacional + "#" + usuario + "#"

+ departamento + "#" + dataMov.format(fmt));

bw.newLine();

} **else** {

bw.write(linha);

bw.newLine();

}

}

System.***out***.println(" │");

System.***out***.println(" └█ Saída realizada com sucesso!");

br.close();

bw.close();

}

// Transferindo informações do arquivo temporário para "CadastroAsset.txt"

BufferedWriter bw2 = **new** BufferedWriter(**new** FileWriter(caminho));

BufferedReader br2 = **new** BufferedReader(

**new** FileReader(caminho.replace("\\CadastroAsset.txt", "\\tmp.txt")));

**while** (br2.ready()) {

String linha = br2.readLine();

bw2.write(linha);

bw2.newLine();

}

br2.close();

bw2.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Realiza a entrada de um ativo do inventário

\*/

**public** **boolean** entradaAsset(String caminhoDiretorio, String caminho, String stEntrada) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

**if** (stEntrada.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ Campo em branco!");

} **else** {

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

BufferedWriter bw = **new** BufferedWriter(

**new** FileWriter(caminho.replace("\\CadastroAsset.txt", "\\tmp.txt")));

dataMov = LocalDate.*now*();

DateTimeFormatter fmt = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

**if** (vet[0].equalsIgnoreCase(stEntrada)) {

bw.write(vet[0] + "#" + vet[1] + "#" + vet[2] + "#" + vet[3] + "#" + vet[4] + "#" + vet[5] + "#"

+ vet[6] + "#" + vet[7] + "#" + status + "#" + sistemaOperacional + "#" + usuario + "#"

+ departamento + "#" + dataMov.format(fmt));

bw.newLine();

} **else** {

bw.write(linha);

bw.newLine();

}

}

System.***out***.println(" │");

System.***out***.println(" └█ Entrada realizada com sucesso!");

br.close();

bw.close();

}

BufferedWriter bw2 = **new** BufferedWriter(**new** FileWriter(caminho));

BufferedReader br2 = **new** BufferedReader(

**new** FileReader(caminho.replace("\\CadastroAsset.txt", "\\tmp.txt")));

**while** (br2.ready()) {

String linha = br2.readLine();

bw2.write(linha);

bw2.newLine();

}

br2.close();

bw2.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Listar todos os ativos no console(tela)

\*/

**public** **boolean** listarAsset(String caminhoDiretorio, String caminho) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

System.***out***.println("\n\t\t\t\t\t\t\t\t\t◄│▒│ LISTAGEM DE ATIVOS │▒│► ");

System.***out***.println(

"╔─►─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╗");

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

"SERVICE TAG", "TIPO", "PERFIL", "VALOR", "NOTA FISCAL", "PROPRIETÁRIO", "LOCAL", "DATA CADASTRO",

"STATUS", "SO", "USUÁRIO", "DEPARTAMENTO", "DATA MOVIMENTO");

System.***out***.println();

System.***out***.println(

"╚─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────◄─╝");

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

vet[0], vet[1], vet[2], vet[3], vet[4], vet[5], vet[6], vet[7], vet[8], vet[9], vet[10],

vet[11], vet[12]);

System.***out***.println();

}

br.close();

System.***out***.println(

"╚───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╝");

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Listar os ativos por parâmetro no console(tela)

\*/

**public** **boolean** listarAsset(String caminhoDiretorio, String caminho, String localizar) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

**if** (localizar.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ Campo em branco!");

} **else** {

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

**boolean** naoLocalizado = **true**;

System.***out***.println("\n\t\t\t\t\t\t\t\t\t◄│▒│ LISTAGEM DE ATIVOS │▒│► ");

System.***out***.println(

"╔─►─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╗");

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

"SERVICE TAG", "TIPO", "PERFIL", "VALOR", "NOTA FISCAL", "PROPRIETÁRIO", "LOCAL",

"DATA CADASTRO", "STATUS", "SO", "USUÁRIO", "DEPARTAMENTO", "DATA MOVIMENTO");

System.***out***.println();

System.***out***.println(

"╚─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────◄─╝");

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

**for** (String indice : vet) {

**if** (indice.contains(localizar)) {

System.***out***.format(

"%12.12s %8.8s %8.8s %8.8s %12.12s %12.12s %6.6s %14.14s %8.8s %10.10s %20.20s %15.15s %15.15s",

vet[0], vet[1], vet[2], vet[3], vet[4], vet[5], vet[6], vet[7], vet[8], vet[9],

vet[10], vet[11], vet[12]);

System.***out***.println();

naoLocalizado = **false**;

**break**;

}

}

}

System.***out***.println(

"╚───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╝");

**if** (naoLocalizado) {

System.***out***.println(" │");

System.***out***.println(" └█ Não encontramos resultado para sua pesquisa. <" + localizar + ">");

}

br.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Exibir a quantidade de ativos por item

\*/

**public** **boolean** quantitativoAsset(String caminhoDiretorio, String caminho) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

ArrayList<String> listaAsset = **new** ArrayList<String>();// Declarando e instanciando uma ArrayList

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

listaAsset.add(vet[1]);// Adicionando um item na ArrayList

}

br.close();

// Contador

**int** notebook = 0;

**int** macbook = 0;

**int** imac = 0;

**int** desktop = 0;

**int** aio = 0;

**int** monitor = 0;

// Percorrendo a lista

**for** (**int** i = 0; i < listaAsset.size(); i++) {

String asset = listaAsset.get(i);

**switch** (asset) {

**case** "NOTEBOOK":

notebook++;

**break**;

**case** "MACBOOK":

macbook++;

**break**;

**case** "IMAC":

imac++;

**break**;

**case** "DESKTOP":

desktop++;

**break**;

**case** "AIO":

aio++;

**break**;

**case** "MONITOR":

monitor++;

**break**;

**default**:

**break**;

}

}

System.***out***.println("╔────────────────────────╗");

System.***out***.println("│ │▒│ ESTOQUE ATIVOS │▒│ │");

System.***out***.println("└────────────────────────┘");

System.***out***.printf("%7s %10s", "ASSET ", " VOL.\n");

System.***out***.println(" ──────────────────────── ");

System.***out***.printf("%10s %4s", " Notebook: ", notebook + "\n");

System.***out***.printf("%10s %4s", " Macbook: ", macbook + "\n");

System.***out***.printf("%10s %4s", " Macbook: ", imac + "\n");

System.***out***.printf("%10s %4s", " Desktop: ", desktop + "\n");

System.***out***.printf("%10s %4s", " Macbook: ", aio + "\n");

System.***out***.printf("%10s %4s", " Monitor: ", monitor + "\n");

System.***out***.println("┌────────────────────────┐");

System.***out***.printf("%10s %4s", " Total: ", listaAsset.size() + "\n");

System.***out***.println("└────────────────────────┘");

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Exibir soma de valores de ativos por item

\*/

**public** **boolean** contabilAsset(String caminhoDiretorio, String caminho) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

ArrayList<String> listaAsset = **new** ArrayList<String>();// ArrayList para o item

ArrayList<Double> listaValores = **new** ArrayList<Double>();// ArrayList para o valor do item

**while** (br.ready()) {

String linha = br.readLine();

String vet[] = linha.split("#");

listaAsset.add(vet[1]);

listaValores.add(Double.*parseDouble*(vet[3]));

}

br.close();

**double** notebook = 0;

**double** macbook = 0;

**double** imac = 0;

**double** desktop = 0;

**double** aio = 0;

**double** monitor = 0;

**double** total = 0;

**for** (**int** i = 0; i < listaAsset.size(); i++) {

String asset = listaAsset.get(i);

**double** valores = listaValores.get(i);

**switch** (asset) {

**case** "NOTEBOOK":

notebook += valores;

**break**;

**case** "MACBOOK":

macbook += valores;

**break**;

**case** "IMAC":

imac += valores;

**break**;

**case** "DESKTOP":

desktop += valores;

**break**;

**case** "AIO":

aio += valores;

**break**;

**case** "MONITOR":

monitor += valores;

**break**;

**default**:

**break**;

}

total += valores;

}

System.***out***.println("╔────────────────────────╗");

System.***out***.println("│ │▒│ ESTOQUE ATIVOS │▒│ │");

System.***out***.println("└────────────────────────┘");

System.***out***.printf("%7s %10s", "ASSET ", " VALOR\n");

System.***out***.println(" ──────────────────────── ");

System.***out***.printf("%10s %4s", " Notebook: R$ ", notebook + "\n");

System.***out***.printf("%10s %4s", " Macbook: R$ ", macbook + "\n");

System.***out***.printf("%10s %4s", " Macbook: R$ ", imac + "\n");

System.***out***.printf("%10s %4s", " Desktop: R$ ", desktop + "\n");

System.***out***.printf("%10s %4s", " Macbook: R$ ", aio + "\n");

System.***out***.printf("%10s %4s", " Monitor: R$ ", monitor + "\n");

System.***out***.println("┌────────────────────────┐");

System.***out***.printf("%10s %4s", " Total: R$ ", total + "\n");

System.***out***.println("└────────────────────────┘");

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

/\*

\* Excluir ativo com status OBSOLETO conforme busca do service tag

\*/

**public** **boolean** excluirAsset(String caminhoDiretorio, String caminho, String excluirSt) {

**try** {

verificarArquivo(caminhoDiretorio, caminho);

BufferedReader br = **new** BufferedReader(**new** FileReader(caminho));

BufferedWriter bw = **new** BufferedWriter(

**new** FileWriter(caminho.replace("\\CadastroAsset.txt", "\\Excluir.txt")));

**boolean** naoLocalizado = **true**;

**if** (excluirSt.isEmpty()) {

System.***out***.println(" │");

System.***out***.println(" └█ Asset não localizado!");

} **else** {

**while** (br.ready()) {

String linha;

linha = br.readLine();

String vet[] = linha.split("#");

**if** (vet[0].equalsIgnoreCase(excluirSt) && vet[8].equalsIgnoreCase("OBSOLETO")) {

naoLocalizado = **false**;

System.***out***.println(" │");

System.***out***.println(" └█ Asset localizado e excluído!");

**continue**;

} **else** {

bw.write(linha);

bw.newLine();

}

}

br.close();

bw.close();

BufferedWriter bw2 = **new** BufferedWriter(**new** FileWriter(caminho));

BufferedReader br2 = **new** BufferedReader(

**new** FileReader(caminho.replace("\\CadastroAsset.txt", "\\Excluir.txt")));

**while** (br2.ready()) {

String linha = br2.readLine();

bw2.write(linha);

bw2.newLine();

}

br2.close();

bw2.close();

}

**if** (naoLocalizado) {

System.***out***.println(" │");

System.***out***.println(" └█ Não encontramos resultado para sua pesquisa. <" + excluirSt + ">");

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **false**;

}

}