public class Insumo {

private String codigo;

private String descricao;

private String deposito;

public Insumo(String codigo, String descricao, String deposito) {

this.codigo = codigo;

this.descricao = descricao;

this.deposito = deposito;

}

public String getCodigo() {

return codigo;

}

public String getDescricao() {

return descricao;

}

public String getDeposito(){

return deposito;

}

}

import java.time.LocalDate;

public class Movimentacao {

private LocalDate data;

private int quantidade;

public Movimentacao(LocalDate data, int quantidade) {

this.data = data;

this.quantidade = quantidade;

}

public LocalDate getData() {

return data;

}

public int getQuantidade() {

return quantidade;

}

}

import javax.swing.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.HashMap;

public class Movimentar {

//ARRYS PARA TER UM HISTORICO

private ArrayList<Insumo> insumos;

private HashMap<String, Integer> quantidades;

private HashMap<String, ArrayList<Movimentacao>> historicoMovimentacoes;

public Movimentar() {

insumos = new ArrayList<>();

quantidades = new HashMap<>();

historicoMovimentacoes = new HashMap<>();

}

// IRÁ MOVIMENTAR

public void movimentar(String codigo, String tipo, int quantidade) {

try {

for (Insumo insumo : insumos) {

if (insumo.getCodigo().equals(codigo)) {

// E, PARA A ENTRADA

if (tipo.equals("E")) {

quantidades.put(codigo, quantidades.getOrDefault(codigo, 0) + quantidade);

registrarMovimentacao(codigo, quantidade);

JOptionPane.showMessageDialog(null, "Entrada de " + quantidade

+ " unidades de " + insumo.getDescricao() + " no depósito " + insumo.getDeposito());

// S, PARA A SAIDA

} else if (tipo.equals("S")) {

int quantidadeAtual = quantidades.getOrDefault(codigo, 0);

if (quantidadeAtual >= quantidade) {

quantidades.put(codigo, quantidadeAtual - quantidade);

registrarMovimentacao(codigo, -quantidade);

JOptionPane.showMessageDialog(null, "Saída de " + quantidade

+ " unidades de " + insumo.getDescricao() + " no depósito " + insumo.getDeposito());

} else {

JOptionPane.showMessageDialog(null, "Estoque insuficiente para "

+ insumo.getDescricao());

}

}

return;

}

}

JOptionPane.showMessageDialog(null, "Insumo com código " + codigo + " não encontrado.");

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erro ao movimentar insumo: " + e.getMessage());

}

}

// REGISTRA AS MOVIMENTAÇÕES, COM DATAS E HORAS

private void registrarMovimentacao(String codigo, int quantidade){

try {

Movimentacao movimentacao = new Movimentacao(LocalDate.now(), quantidade);

historicoMovimentacoes.putIfAbsent(codigo, new ArrayList<>());

historicoMovimentacoes.get(codigo).add(movimentacao);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erro ao registrar movimentação: " + e.getMessage());

}

}

// PEGA A QUANTIDADE DE QUALQUER INSUMO

public int getQuantidade(String codigo) {

return quantidades.getOrDefault(codigo, 0);

}

// ADICIONA INSUMOS, AO ARRAY

public void adicionarInsumo(Insumo insumo) {

try {

for (Insumo i : insumos) {

if (i.getCodigo().equals(insumo.getCodigo())) {

JOptionPane.showMessageDialog(null, "Insumo com código " + insumo.getCodigo() + " já existe.");

return;

}

}

insumos.add(insumo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erro ao adicionar insumo: " + e.getMessage());

}

}

//QUANTIDADE DE ESTOQUE DE MATERIA PRIMA

public int quantidadeTotalMateriaPrima() {

int total = 0;

try {

for (int quantidade : quantidades.values()) {

total += quantidade;

}

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erro ao calcular quantidade total: " + e.getMessage());

}

return total;

}

// HISTORICO DE MOVIMENTAÇÃO DE CADA INSUMO

public ArrayList<Movimentacao> getHistoricoMovimentacoes(String codigo) {

try {

return historicoMovimentacoes.getOrDefault(codigo, new ArrayList<>());

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erro ao obter histórico de movimentações: " + e.getMessage());

return new ArrayList<>();

}

}

}