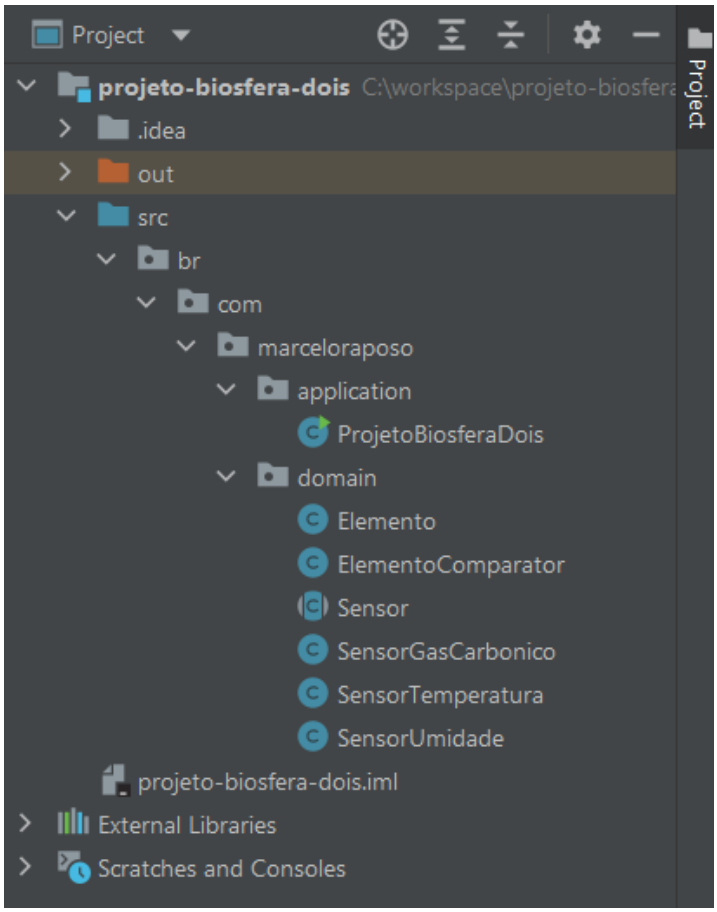
Projeto Biosfera 2

Nome: Marcelo Sodré Raposo Júnior
Matrícula: 01503627
Curso: Superior de tecnologia em Análise e Desenvolvimento de Sistemas

Foi pedido para que fosse criado um programa, na linguagem de computação Java, onde sua função é registrar as informações de 3 (três) tipos de sensores, eles medem e registram a temperatura, umidade e a quantidade de gás carbônico em um ambiente controlado.

Segue a implementação do *Projeto Biosfera 2* com os conceitos aprendidos durante o curso.

Foi utilizado o ambiente de desenvolvimento integrado IntelliJ IDEA e a estrutura de pacotes do projeto ficou da seguinte maneira:



Pode-se perceber que foram implementadas as classes Elemento, ElementoComparador, Sensor, SensorGasCarbonico, SensorTemperatura, SensorUmidade e ProjetoBiosferaDois que possui o método main() para rodar o programa.

A classe abstrata Sensor foi implementada para que as classes SensorGasCarbonico, SensorTemperatura e SensorUmidade estendessem e usassem seus métodos, observemos eles e a codificação da classe:

```java
package br.com.marceloraposo.domain;

import java.util.Random;
public abstract class Sensor {

    private final Random valores = new Random();

    public abstract void inicializar();

    public abstract void ordenarDados();

    protected static void geradorHoras(String[][] dadosMedicao){
        int hora = 0;

        for (int i = 0; i < 24; i++) {
            for (int j = 0; j < 2; j++) {
                if (i < 10) {
                    if (j % 2 == 0) {
                        dadosMedicao[hora][1] = "0" + i + ":00";
                    } else {
                        dadosMedicao[hora][1] = "0" + i + ":30";
                    }
                } else {
                    if (j % 2 == 0) {
                        dadosMedicao[hora][1] = i + ":00";
                    } else {
                        dadosMedicao[hora][1] = i + ":30";
                    }
                }
                hora++;
            }
        }
    }

    protected static void message(String[][] dadosMedicao) {
        String message = "";
        int count = -1;

        while (count != 47) {
            count++;
            message = String.format("%s - %s ", dadosMedicao[count][0], dadosMedicao[count][1]);
            System.out.println(message);
        }
    }

    public Random getValores() {
        return valores;
    }
}
```

Na classe abstrata, temos os métodos abstratos inicializar() e ordenarDados(), que têm que ser implementados pelas outras classes que estenderem ela. Também temos o método geradorHoras() que recebe como parâmetro uma array multidimensional (matriz) para gerar as horas em que os sensores precisam fazer as medições (logicamente podemos perceber que este método gera horários fictícios para exemplo no programa). Em seguida temos o método message(), que recebe como parâmetro uma matriz para gerar uma mensagem com os valores. Tem como único atributo, valores, que tem como tipo o Random, para gerar valores aleatórios, o atributo tem como modificador de

visibilidade private, para que apenas a própria classe manipule-o, ele fica sendo acessado pelo método getValor() que retorna o valor do atributo, esse conceito é chamado de encapsulamento.

Em seguida, temos a implementação da classe SensorGasCarbonico, que estende a classe Sensor, vejamos:

```
package br.com.marceloraposo.domain;

import java.util.*;
public class SensorGasCarbonico extends Sensor {

    private String[][] medicaoGasCarbonico;
    private List<Elemento> data;

    public SensorGasCarbonico() {
        medicaoGasCarbonico = new String[48][2];
        data = new ArrayList<>();
    }

    @Override
    public void inicializar() {
        for (int i = 0; i < 48; i++) {
            medicaoGasCarbonico[i][0] = "" + getValores().nextInt(900, 1200);
        }
        Sensor.geradorHoras(medicaoGasCarbonico);

        for (int i = 0; i < 48; i++) {
            data.add(new Elemento(Integer.parseInt(medicaoGasCarbonico[i][0]), medicaoGasCarbonico[i][1]));
        }
    }

    public void imprimir() {
        System.out.println("MEDIÇÃO GÁS CARBÔNICO EM PPM (Parte por milhão)");
        System.out.println("_____");
        message(medicaoGasCarbonico);
        System.out.println("_____");
    }

    @Override
    public void ordenarDados() {
        data.sort(new ElementoComparator());

        System.out.println("ORDENAÇÃO - (MEDIÇÃO GÁS CARBÔNICO)");
        System.out.println(data);
    }
}
```

Quando uma classe utiliza a palavra reservada extends, isso quer dizer que ela está se valendo de um conceito de POO que se chama Herança, dessa forma, a superclasse (Sensor) fica vinculada a subclasse (SensorGasCarbonico) por uma hierarquia e por conseguinte é obrigado a implementar todos os seus métodos abstratos, como já foi dito acima.

É interessante notarmos que a classe SensorGasCarbonico tem dois atributos, são eles: medicaoGasCarbonico, é a matriz onde guardará todos os dados de medição, horas e data, tem como tipo uma estrutura de dados chamada List, que guardará valores do tipo Elemento. O método construtor da classe, assim que é chamado, quando instancia-se um

objeto, inicializa os atributos da classe para serem utilizados. Temos como métodos: inicializar(), que chama o método da classe Sensor getvalores() para preencher a primeira coluna da matriz medicaoGasCarbonico, e chama o método geradorHoras() para preencher a segunda coluna com os valores das horas. Tem-se também o método ordenarDados(), que usa o método sort() para ordenar os valores que foram atribuídos a Lista de Elementos, que por sinal, é o mesmo que foram atribuídos à matriz medicaoGasCarbonico. Foi optado por usar esta estrutura de dados, porque é possível receber dois tipos de valores diferentes e assim ordenar as medições de acordo com as horas. Por último, temos o método imprimir(), que, como o próprio nome se refere, imprime os valores na saída do console.

A implementação das classes SensorTemperatura e SensorUmidade, seguem a mesma lógica da classe SensorGasCarbonico. Vamos a suas implementações:

```
package br.com.marceloraposo.domain;

import java.util.ArrayList;
import java.util.List;
public class SensorTemperatura extends Sensor {

    private String[][] medicaoTemperatura;
    private List<Elemento> data;

    public SensorTemperatura(){
        medicaoTemperatura = new String[48][2];
        data = new ArrayList<>();
    }

    @Override
    public void inicializar() {
        for (int i = 0; i < 48; i++) {
            medicaoTemperatura[i][0] = "" + getValores().nextInt(20, 26);
        }
        Sensor.geradorHoras(medicaoTemperatura);

        for (int i = 0; i < 48; i++) {
            data.add(new Elemento(Integer.parseInt(medicaoTemperatura[i][0]), medicaoTemperatura[i][1]));
        }
    }

    public void imprimir() {
        System.out.println("MEDIÇÃO: TEMPERATURA °C (GRAUS CELCIUS)");
        System.out.println("_____");
        message(medicaoTemperatura);
        System.out.println("_____");
    }

    @Override
    public void ordenarDados() {
        data.sort(new ElementoComparator());

        System.out.println("_____");
        System.out.println("ORDENAÇÃO - (MEDIÇÃO TEMPERATURA)");
        System.out.println(data);
        System.out.println("_____");
    }
}
```

```java
package br.com.marceloraposo.domain;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class SensorUmidade extends Sensor {

    private String[][] medicaoUmidade;
    private List<Elemento> data;

    public SensorUmidade(){
        medicaoUmidade = new String[48][2];
        data = new ArrayList<>();
    }

    @Override
    public void inicializar() {
        for (int i = 0; i < 48; i++) {
            medicaoUmidade[i][0] = "" + getValores().nextInt(40, 70);
        }
        Sensor.geradorHoras(medicaoUmidade);

        for (int i = 0; i < 48; i++) {
            data.add(new Elemento(Integer.parseInt(medicaoUmidade[i][0]), medicaoUmidade[i][1]));
        }
    }

    public void imprimir() {
        System.out.println("MEDIÇÃO: UMIDADE RELATIVA DO AR EM % (Porcentagem)");
        System.out.println("_____");
        message(medicaoUmidade);
        System.out.println("_____");
    }

    @Override
    public void ordenarDados() {
        data.sort(new ElementoComparator());
        Collections.reverse(data);

        System.out.println("_____");
        System.out.println("ORDENAÇÃO - (MEDIÇÃO UMIDADE)");
        System.out.println(data);
        System.out.println("_____");
    }
}
```

As classes Elemento e ElementoComparador foram implementadas para poder, como já foi dito, usar o método sort(), que recebe como parâmetro uma classe que implementa a interface Comparator.

Segue a estrutura das classes:

```java
package br.com.marceloraposo.domain;

public class Elemento implements Comparable<Elemento>{

    private int medicao;
    private String hora;
```

```
    public Elemento(int medicao, String hora) {
        this.medicao = medicao;
        this.hora = hora;
    }

    public int getMedicao() {
        return medicao;
    }

    public void setMedicao(int medicao) {
        this.medicao = medicao;
    }

    public String getHora() {
        return hora;
    }

    public void setHora(String hora) {
        this.hora = hora;
    }

    @Override
    public int compareTo(Elemento o) {
        return 0;
    }

    @Override
    public String toString() {
        return "{" +
                "Medição = '" + medicao + '\'' +
                ", Hora = " + hora +
                '}' + "\n";
    }
}
```

```
package br.com.marceloraposo.domain;

import java.util.Comparator;
public class ElementoComparator implements Comparator<Elemento> {

    @Override
    public int compare(Elemento o1, Elemento o2) {
        return Integer.compare(o1.getMedicao(), o2.getMedicao());
    }
}
```

E por fim temos a classe ProjetoBiosferaDois que implementa o método main() e assim consegue executar o programa.

O programa interage com o usuário pedindo para que seja digitada uma opção válida, caso não seja, o programa captura a exceção, identifica e retorna o erro, dessa forma, mantém o programa em andamento até que o usuário encerre-o digitando a opção de saída.

Temos o programa:

```java
package br.com.marceloraposo.application;

import br.com.marceloraposo.domain.SensorGasCarbonico;
import br.com.marceloraposo.domain.SensorTemperatura;
import br.com.marceloraposo.domain.SensorUmidade;

import java.util.InputMismatchException;
import java.util.Scanner;

public class ProjetoBiosferaDois {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        SensorGasCarbonico sensorGasCarbonico = new SensorGasCarbonico();
        sensorGasCarbonico.inicializar();

        SensorUmidade sensorUmidade = new SensorUmidade();
        sensorUmidade.inicializar();

        SensorTemperatura sensorTemperatura = new SensorTemperatura();
        sensorTemperatura.inicializar();

        System.out.println();
        System.out.println("                                    BEM-VINDO AO SISTEMA DE MONITORAMENTO DO
PROJETO BIOSFERA 2");
        System.out.println();

        int opcao = 0;
        while (true) {

            try {
                System.out.println("Qual dos Sensores você deseja visualizar os dados: ");
                System.out.println("[ 1 ] Sensor de TEMPERATURA");
                System.out.println("[ 2 ] Sensor de UMIDADE");
                System.out.println("[ 3 ] Sensor de GÁS CARBÔNICO");
                System.out.println("[ 0 ] Sair do Sistema");
                System.out.print("Digite sua Opção: ");
                opcao = scanner.nextInt();
                System.out.println();

                if (opcao == 0) {
                    break;

                } else if (opcao == 1) {
                    sensorTemperatura.imprimir();
                    System.out.println();
                    System.out.println("Deseja Ordenar os Dados?");
                    System.out.println("[ 1 ] SIM");
                    System.out.println("[ 2 ] NÃO");
                    opcao = scanner.nextInt();
                    System.out.println();
                    if (opcao == 1) {
                        sensorTemperatura.ordenarDados();
                        System.out.println();
                    } else if (opcao == 2) {
                        continue;
                    } else {
                        System.out.println("Opção Inválida!!!");
```

```
                }

            } else if (opcao == 2) {
                sensorUmidade.imprimir();
                System.out.println();
                System.out.println("Deseja Ordenar os Dados?");
                System.out.println("[ 1 ] SIM");
                System.out.println("[ 2 ] NÃO");
                opcao = scanner.nextInt();
                System.out.println();
                if (opcao == 1) {
                    sensorUmidade.ordenarDados();
                    System.out.println();
                } else if (opcao == 2) {
                    continue;
                } else {
                    System.out.println("Opção Inválida!!!");
                }

            } else if (opcao == 3) {
                sensorGasCarbonico.imprimir();
                System.out.println();
                System.out.println("Deseja Ordenar os Dados?");
                System.out.println("[ 1 ] SIM");
                System.out.println("[ 2 ] NÃO");
                opcao = scanner.nextInt();
                System.out.println();
                if (opcao == 1) {
                    sensorGasCarbonico.ordenarDados();
                    System.out.println();
                }  else if (opcao == 2) {
                    continue;
                } else {
                    System.out.println("Opção Inválida!!!");
                }
            } else {
                System.out.println("Opção inválida!");
                System.out.println("Tente novamente.");
            }
        } catch (InputMismatchException exception) {
            scanner.nextLine();
            System.err.println("Digite uma opção válida!!!");
        }
    }
  }
}
```

Referências:

Fábio da Silva Souza: Programação Orientada a Objetos (Ser Educacional).