Trabalho 2 POO

Nome: William Cardoso Barbosa

1 -

As classes abastra são classes que não permitem a criação de instâncias. Elas são criadas apenas para serem modelos de outras classes que são derivadas. As classes deverivadas, normalmente, sobrescrevem os métodos das classes abstratas. Essa ferramenta permite maior nivel de abstração do código e redução do código.

2 -

```
// Classe com problemas


abstract class RespostasProvaConcurso {
  private String concuro = "ProgramadorTJ";
  private int qtQuestoes = 10;
  private char respostas[];
  abstract float calculaNotaPessoa ();
  public boolean[] verificaAcertos(char questoes[]) {
    boolean acertos[];
    for (int i=1; i < this.qtQuestoes; i++) {
      acertos[i] = this.isAcerto(this.getRespostaQuestao(i), questoes[i]);
    }
    return acertos;
  }
  public char getRespostaQuestao(int numeroQuestao) {
    return respostas[numeroQuestao ];
  }
  private boolean isAcerto(char a, char b) {
    return a == b;
  }
}
```

```
// apontamento de erros


abstract class RespostasProvaConcurso {
  private String concuro = "ProgramadorTJ"; // -> o nome da variável está errado
  private int qtQuestoes = 10; // -> não há uma forma da classes concreta acessar essa variável
  private char respostas[]; // -> não há uma forma da classes concreta acessar essa variável

  //não possui construtor
  abstract float calculaNotaPessoa ();
  public boolean[] verificaAcertos(char questoes[]) {
    boolean acertos[]; // -> não é possível instanciar um array de boolean
    for (int i=1; i < this.qtQuestoes; i++) {
      acertos[i] = this.isAcerto(this.getRespostaQuestao(i), questoes[i]); // não é possível acessar um array de boolean
    }
    return acertos;
  }
  public char getRespostaQuestao(int numeroQuestao) {
    return respostas[numeroQuestao ];
  }
```

```
  private boolean isAcerto(char a, char b) {
    return a == b;
  }
}
```

```
// otimizações e correções


import java.util.ArrayList;

abstract class RespostasProvaConcurso {
  private String concurso = "ProgramadorTJ";
  private int qtQuestoes = 10;
  private char respostas[];

  abstract float calculaNotaPessoa();

  public RespostasProvaConcurso(int qtQuestoes, char respostas[]) {
    this.setQtQuestoes(qtQuestoes);
    this.setRespostas(respostas);
  }

  public ArrayList<Boolean> verificaAcertos(char questoes[]) {
    ArrayList<Boolean> acertos = new ArrayList<Boolean>();
    for (int i = 0; i < this.qtQuestoes; i++) {
      acertos.add(this.isAcerto(this.getRespostaQuestao(i), questoes[i]));
    }
    return acertos;
  }

  public char getRespostaQuestao(int numeroQuestao) {
    return this.respostas[numeroQuestao];
  }

  private boolean isAcerto(char a, char b) {
    return a == b;
  }

  public String getConcurso() {
    return this.concurso;
  }

  public int getQtQuestoes() {
    return this.qtQuestoes;
  }

  public char[] getRespostas() {
    return this.respostas;
  }

  public void setRespostas(char[] respostas) {
    this.respostas = respostas;
  }

  public void setConcurso(String concurso) {
    this.concurso = concurso;
  }

  public void setQtQuestoes(int qtQuestoes) {
    this.qtQuestoes = qtQuestoes;
  }
}
```

6 -

A Interface pode ser entendida como um contrato entre a classe e seu exterior. Quando uma classe se compromete a implementar uma interface ela deve construir seus métodos e atributos conforme estabelecidos no contrato. A grande diferença de uma classe abstrata para uma interface seria a implementação de códigos, a interface apenas permite a criação de uma assinatura de um método ou propriedade, já a classe abstrata permite a inserção de blocos de códigos e possíveis sobrescritas no futuro.

7 -

```
public interface AlgoErradoNaoEstaCerto {
private float salarioMensal = 5.000f;
public AlgoErradoNaoEstaCerto () {
}
public AlgoErradoNaoEstaCerto(float salario) {
this.salarioMensal = salario;
}
void calculaDescontoSalario ();
void calcularImpostoDeRenda ();
public void umMetodo(int umValor) {
System.out.println("UmaMensagem");
}
public float getSalarioMensal () {
return this.salarioMensal;
}
public void setSalarioMensal(float salario) {
this.salarioMensal = salario;
}
default mostraMsg
}
```

```
// apontamento de erros
public interface AlgoErradoNaoEstaCerto {
  private float salarioMensal = 5.000f;
  public AlgoErradoNaoEstaCerto () {} // interfaces não podem ter construtores
  public AlgoErradoNaoEstaCerto(float salario) {
  this.salarioMensal = salario;
  }
  void calculaDescontoSalario ();
  void calcularImpostoDeRenda ();
  public void umMetodo(int umValor) { // um método sem ser default ou static não pode ter corpo
  System.out.println("UmaMensagem");
  }
  public float getSalarioMensal () { // um método sem ser default ou static não pode ter corpo
  return this.salarioMensal;
  }
  public void setSalarioMensal(float salario) {
  this.salarioMensal = salario;
  }
  default mostraMsg // erro de sintax
}
```

8 - A firmação é falsa, uma classe abstrata pode conter métodos não abstratos, que possuem corpos e também pode possui atributos não estáticos, inclusive com getters e setters.

9-

```
public interface Banco {
  public static final String NOME_BANCO = "Banco do Brasil";
  public static final String PRIVILEGIO = "admin";
  public double saldo = 0.0;
  public void depositar(double valor);
  public void sacar(double valor);
  public double saldo();
  public void transferir(Banco destino, double valor);
  public default void imprimirExtrato() {
    System.out.println("Saldo: " + saldo());
  }
  public static void imprimirNomeBanco() {
    System.out.println(NOME_BANCO);
  }
  public static void imprimirPrivilegio() {
    System.out.println(PRIVILEGIO);
  }

}
```