5.8 EXERCÍCIOS: ENCAPSULAMENTO, CONSTRUTORES E STATIC

1. Adicione o modificador de visibilidade (`private` , se necessário) para cada atributo e método da classe `Conta` . Tente criar uma `Conta` no `main` e modificar ou ler um de seus atributos privados. O que acontece?

2. Crie apenas os getters e setters necessários da sua classe `Conta` . Pense sempre se é preciso criar cada um deles. Por exemplo:

```
class Conta {
    private String titular;

    // ...

    public String getTitular() {
        return this.titular;
    }

    public void setTitular(String titular) {
        this.titular = titular;
    }
}
```

   Não copie e cole! Aproveite para praticar sintaxe. Logo passaremos a usar o Eclipse e aí sim teremos procedimentos mais simples para este tipo de tarefa.

   Repare que o método `calculaRendimento` parece também um getter. Aliás, seria comum alguém nomeá-lo de `getRendimento` . Getters não precisam apenas retornar atributos. Eles podem trabalhar com esses dados.

3. Modifique suas classes que acessam e modificam atributos de uma `Conta` para utilizar os getters e setters recém criados.

Por exemplo, onde você encontra:

```
c.titular = "Batman";
System.out.println(c.titular);
```

passa para:

```
c.setTitular("Batman");
System.out.println(c.getTitular());
```

4. Faça com que sua classe `Conta` possa receber, opcionalmente, o nome do titular da `Conta` durante a criação do objeto. Utilize construtores para obter esse resultado.

   Dica: utilize um construtor sem argumentos também, para o caso de a pessoa não querer passar o titular da `Conta` .

   Seria algo como:

```
class Conta {
    public Conta() {
        // construtor sem argumentos
    }

    public Conta(String titular) {
        // construtor que recebe o titular
    }
}
```

   Por que você precisa do construtor sem argumentos para que a passagem do nome seja opcional?

5. (opcional) Adicione um atributo na classe `Conta` de tipo `int` que se chama identificador. Esse identificador deve ter um valor único para cada instância do tipo `Conta` . A primeira `Conta` instanciada tem identificador 1, a segunda 2, e assim por diante. Você deve utilizar os recursos aprendidos aqui para resolver esse problema.

   Crie um getter para o identificador. Devemos ter um setter?

6. (opcional) Como garantir que datas como 31/2/2012 não sejam aceitas pela sua classe `Data` ?

7. (opcional) Suponha que temos a classe `PessoaFisica` que tem um CPF como atributo. Como garantir que pessoa física alguma tenha CPF invalido, nem seja criada `PessoaFisica` sem cpf inicial? (Suponha que já existe um algoritmo de validação de cpf: basta passar o cpf por um método `valida(String x)...`)

5.9 DESAFIOS

1. Porque esse código não compila?

```
class Teste {
    int x = 37;
    public static void main(String [] args) {
        System.out.println(x);
```

```
        }
    }
```

2. Imagine que tenha uma classe `FabricaDeCarro` e quero garantir que só existe um objeto desse tipo em toda a memória. Não existe uma palavra chave especial para isto em Java, então teremos de fazer nossa classe de tal maneira que ela respeite essa nossa necessidade. Como fazer isso? (pesquise: singleton design pattern)