Exercício projeto bancov05. Exercício para ser feito em classe.

1) Abra o eclipse e crie um projeto chamado bancov05 a partir do projeto bancov04. Mantenha as classes Cliente, Conta e TestaEncapsulamento.
2) Faça as seguintes modificações na classe TestaAgregacao2 (a partir da linha 10) e em seguida renomeie para TestaConstrutor (Botão direito do mouse → refactor → rename).

```
2  public class TestaConstrutor {
3      public static void main(String[] args) {
4          Conta umaConta = new Conta();
5          umaConta.getTitular().setNome("Rafael");
6          umaConta.setNumero(1);
7          umaConta.deposita(500);
8          umaConta.saca(1001); // vai impedir
9          umaConta.deposita(-100); //vai impedir
10
11         System.out.println("Nome do titular: "+umaConta.getTitular().getNome());
12         System.out.println("Saldo: "+umaConta.getSaldo());
13         System.out.println("Numero: "+umaConta.getNumero());
14     }
15 }
```

3) Apenas para facilitar nossa programação, crie um método exibeDados na classe Cliente e outro na classe Conta.

Cliente

```
32     public void exibeDados() {
33         System.out.println("Nome: "+this.nome);
34         System.out.println("Cpf: "+this.cpf);
35         System.out.println("Email: "+this.email);
36     }
```

Conta

```
54     public void exibeDados() {
55         System.out.println("Numero: "+this.numero);
56         System.out.println("Saldor: R$"+this.saldo);
57         System.out.println("Dados do Titular: ");
58         this.titular.exibeDados();
59     }
```

Em seguida, modifique a classe TestaConstrutor e verifique se tudo continua funcionando.

```
2  public class TestaConstrutor {
3      public static void main(String[] args) {
4          Conta umaConta = new Conta();
5          umaConta.getTitular().setNome("Rafael");
6          umaConta.setNumero(1);
7          umaConta.deposita(500);
8          umaConta.saca(1001); // vai impedir
9          umaConta.deposita(-100); //vai impedir
10
11         umaConta.exibeDados();
12     }
13 }
```

4) Na classe Conta, retire a inicialização de atributos e forneça um construtor onde esses atributos são inicializados por padrão.

```
2  public class Conta {
3      // Atributos
4      private Cliente titular;
5      private double saldo;
6      private int numero;
7      //construtor
8      public Conta() {
9          System.out.println("Criando/instanciando um objeto do tipo Conta...");
10         this.saldo=500;
11         this.titular=new Cliente();
12     }
```

5) Ainda assim, nossa conta está nascendo num estado inconsistente. Crie outro construtor que receba o número da conta como argumento.

```java
public class Conta {
    // Atributos
    private Cliente titular;
    private double saldo;
    private int numero;
    //construtor
    public Conta() {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
    }
    public Conta(int numero) {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
        this.numero=numero;
    }
```

Perceba que agora temos dois construtores. Um que exige um valor para o atributo número e outro que não exige o fornecimento de nenhuma informação.
Do jeito que está, o fornecimento do número da conta é opcional.

Sobrecarga → Quando temos dois ou mais construtores com o mesmo nome e assinaturas diferentes, damos a isso o nome de sobrecarga. Nosso construtor da classe Conta está sendo sobrecarregado.
O mesmo se aplica a métodos.
Métodos com nome igual e assinaturas diferentes são métodos sobrecarregados. Peça ao seu professor para mostrar um exemplo.

Veja como ficaria a classe TestaConstrutor. Teste esse código!

```java
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        umaConta.setNumero(1);
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        Conta outraConta = new Conta();

        umaConta.exibeDados();
        outraConta.exibeDados();
    }
}
```

6) Acontece que não queremos que o fornecimento do número da conta seja opcional. Queremos que seja obrigatório. Nesse caso, só pode haver um construtor que exige essa informação. Comente o construtor sem argumentos e teste a classe TestaConstrutor novamente obviamente comentando as linhas em que se invoca o construtor que foi comentado.

Conta

```java
public class Conta {
    // Atributos
    private Cliente titular;
    private double saldo;
    private int numero;
    //construtor
    /*public Conta() {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
    }*/
    public Conta(int numero) {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
        this.numero=numero;
    }
```

TestaConstrutor

```java
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        umaConta.setNumero(1);
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        //Conta outraConta = new Conta();

        umaConta.exibeDados();
        //outraConta.exibeDados();
    }
}
```

7) NOVA REGRA! Uma vez que seu número foi fornecido na criação de uma conta, o valor desse atributo não pode mais ser modificado. Esse é o princípio da imutabilidade. Portanto, comente o método setNumero de sua classe Conta e transfira a regra que havia nesse método para o construtor.

```java
public class Conta {
    // Atributos
    private Cliente titular; //get (IMUTÁVEL)
    private double saldo; // get e operações bancárias
    private int numero; // get (IMUTÁVEL)
    //construtor
    /*public Conta() {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
    }*/
    public Conta(int numero) {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
        if(numero<=0) {
            System.out.println("Numero invalido para uma conta. A aplicacao sera encerrada.");
            System.exit(0);
        }
        this.numero=numero;
    }
    //métodos acessores
    /*public void setNumero(int numero) {
        if(numero<=0)
            return;//Early return
        //atributo numero = valor do argumento número
        this.numero=numero;
    }*/
```

Perceba que no construtor modificamos um pouco a regra. Agora, ao receber um número inválido, encerramos a aplicação utilizando o método exit da classe System. Essa não é a melhor forma de resolver o problema, mas é o melhor que podemos fazer com os conhecimentos que temos até o momento.

Veja como ficará a classe TestaConstrutor. Não se esqueça de tentar criar uma conta com número inválido e ver o que acontece.

```
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        //umaConta.setNumero(1); //Não existe mais
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        //Conta outraConta = new Conta();

        umaConta.exibeDados();
        //outraConta.exibeDados();
    }
}
```

8) Na classe Cliente, vamos criar dois construtores. Um construtor que recebe o nome do cliente como argumento e outro construtor que não recebe argumentos. Em seguida, altere a classe TestaConstrutor para testar esses novos construtores.

Cliente

```
public class Cliente {
    String nome;
    String cpf;
    String email;
    //Construtor

    public Cliente(String nome) {
        this.setNome(nome);//Aproveitando as regras do setNome
    }
    public Cliente() {

    }
```

TestaConstrutor

```
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        //umaConta.setNumero(1); //Não existe mais
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        //Conta outraConta = new Conta();

        umaConta.exibeDados();
        //outraConta.exibeDados();

        Cliente umCliente = new Cliente("Renata");
        umCliente.exibeDados();
        Cliente outroCliente = new Cliente();
        outroCliente.exibeDados();
    }
}
```

Perceba que até o momento, o fornecimento do nome é opcional e não obrigatório.

9) Na classe Conta, crie um construtor que receba o número da conta e o nome do titular.

```java
public class Conta {
    // Atributos
    private Cliente titular; //get (IMUTÁVEL)
    private double saldo; // get e operações bancárias
    private int numero; // get (IMUTÁVEL)
    //construtor
    public Conta(int numero) {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente();
        if(numero<=0) {
            System.out.println("Numero invalido para uma conta. A aplicacao sera encerrada.");
            System.exit(0);
        }
        this.numero=numero;
    }

    public Conta(int numero, String nomeDoTitular) {
        System.out.println("Criando/instanciando um objeto do tipo Conta...");
        this.saldo=500;
        this.titular=new Cliente(nomeDoTitular);
        if(numero<=0) {
            System.out.println("Numero invalido para uma conta. A aplicacao sera encerrada.");
            System.exit(0);
        }
        this.numero=numero;
    }
```

10) Use TestaConstrutor para testar essa nova opção

```java
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        //umaConta.setNumero(1); //Não existe mais
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        umaConta.exibeDados();
        System.out.println("###################################");
        Conta outraConta = new Conta(26,"Maria");
        outraConta.saca(300);
        outraConta.exibeDados();

        Cliente umCliente = new Cliente("Renata");
        umCliente.exibeDados();
        Cliente outroCliente = new Cliente();
        outroCliente.exibeDados();
    }
}
```

11) Vamos testar outras opções. Vamos criar, na classe Cliente, um construtor que recebe o nome e o cpf do titular.

```java
public class Cliente {
    String nome;
    String cpf;
    String email;
    //Construtor

    public Cliente(String nome) {
        this.setNome(nome);//Aproveitando as regras do setNome
    }
    public Cliente(String nome, String cpf) {
        this(nome);//Invocando o outro construtor que já tem as regras
        this.setCpf(cpf);//Reaproveitando a regra do cpf
    }
```

12) Em TestaConstrutor, teste esse novo construtor de Cliente.

```java
public class TestaConstrutor {
    public static void main(String[] args) {
        Conta umaConta = new Conta(25);
        umaConta.getTitular().setNome("Rafael");
        //umaConta.setNumero(1); //Não existe mais
        umaConta.deposita(500);
        umaConta.saca(1001); // vai impedir
        umaConta.deposita(-100); //vai impedir

        umaConta.exibeDados();
        System.out.println("################################");
        Conta outraConta = new Conta(26,"Maria");
        outraConta.saca(300);
        outraConta.exibeDados();

        Cliente umCliente = new Cliente("Renata");
        umCliente.exibeDados();
        Cliente outroCliente = new Cliente("Pedro","12345678920");
        outroCliente.exibeDados();
    }
}
```

13) Desafio: Na classe Cliente, crie um construtor que receba os 3 argumentos. Na classe conta, crie um construtor que receba o número da conta, o nome, o cpf e o e-mail do titular.
14) Desafio parte 2: Na classe conta, crie um construtor que receba o número e o titular da conta.