Universidade Federal de Uberlândia

Faculdade de Computação
FACOM39502 – Programação Orientada a Objetos
2023/2
Prof. Renato Pimentel

Laboratório 06 – Polimorfismo, classes abstratas

Atividade **individual**.

1. Considere a classe `ContaCorrente`, com a especificação colocada na listagem a seguir. Copie o código, compile a classe e entenda seu funcionamento.

```
public class ContaCorrente {
   private float saldo;
   private int estado;  // 1=conta ativa 2=conta inativa
   private int numConta;
   private int senha;

   public ContaCorrente( float val, int num, int pwd ) {
      numConta = num;
      senha = pwd;
      saldo = val;
      estado = 1; // conta ativa
   }

   public boolean debitaValor( float val, int pwd ) {
      if ( pwd != senha )
         return ( false ); //senha deve ser válida
      if ( estado != 1 )
         return ( false ); //conta deve ser ativa
      if ( val <= 0 )
         return ( false ); //val > 0
      if ( val > saldo )
         return ( false );

      saldo -= val;
      if ( saldo == 0 )
         estado = 2; // torna conta inativa
      return (true);
   }

   public void debitaValor( float val ) {
      saldo -= val;
   }

   public float getSaldo( int pwd ) {
      if ( senha == pwd )
```

```
            return saldo;
        else
            return -1; // indicando que houve problema na senha
    }

    public void creditaValor( int pwd, float val ) {
        if ( senha == pwd )
            saldo += val;
    }

    protected int getEstado( int pwd ) {
        if ( senha == pwd )
            return estado;
        else
            return -1;
    }

    protected void setEstado( int pwd, int e ) {
        if ( senha == pwd )
            estado = e;
    }

    protected boolean isSenha( int pwd ) {
        if ( senha == pwd )
            return true;
        else
            return false;
    }
}
```

Perceba que uma conta-corrente comum deve se tornar inativa se durante a sua movimentação o seu saldo se igualar a zero. Nesse caso, ela não pode receber mais lançamentos e tampouco ser reativada novamente. Essa regra está implementada no método `debitaValor()` da classe `ContaCorrente`.

2. Modifique o código para trabalhar com abstração de classes. No caso, `ContaCorrente` deverá ser a classe abstrata (não se esqueça de modificar os encapsulamentos de seus atributos para protegidos). Implemente as classes `ContaEspecial` e `ContaComum`, derivadas de `ContaCorrente`, segundo as seguintes especificações:

   (a) Considere que a classe `ContaEspecial` defina um atributo `limite`, do tipo `float`, que determina o limite de crédito de uma conta especial. Uma conta especial com limite igual a ZERO deve ser modificada para uma `ContaComum`. Para a conta modificada deve ser criado um novo objeto para que essa conta seja desta nova categoria. Uma conta especial, que tenha um limite

de crédito maior que zero, pode ter o seu saldo igualado a zero sem que a conta se torne inativa.

(b) Implemente também o construtor da classe, considerando que no momento da criação de uma conta especial o valor do limite é inicializado e a conta comum tem esse valor igual a ZERO.

3. Implemente a classe `Banco` – que será a classe principal da aplicação, contendo o método `main()` – e crie objetos da classe `ContaEspecial`. Faça débitos nas contas e analise o funcionamento do sistema.

4. Em sites de relacionamento, é possível categorizar contatos pessoais em subtipos, tais como: família, amigos e colegas de trabalho. Faça um programa, em Java, contendo:

   (a) A classe-mãe, chamada `Contato`, que deve ser abstrata e com os atributos `apelido`, `nome`, `email` e `aniversario`. Acrescente nesta superclasse o método `public String imprimirBasico()`, que imprime o conteúdo básico dos contatos. A seguir defina um método abstrato `imprimirContato()`, que será então implementado nas subclasses de acordo com suas especificidades. Chame o método `imprimirBasico()` dentro dos métodos de `imprimirContato()` das subclasses.

   (b) A classe `Familia`, subclasse de `Contato`, que possui também o atributo `parentesco`, que descreve o tipo de parentesco desse contato (ex.: pai, irmão, etc.).

   (c) A classe `Amigos`, subclasse da classe `Contato`, que possui também o atributo `grau`, que descreve o grau de amizade desse contato (1 = melhor amigo; 2 = amigo; 3 = conhecido).

   (d) A classe `Trabalho`, subclasse da classe `Contato`, que possui também o atributo `tipo`, que descreve o tipo desse contato no trabalho (ex.: chefe, colega, etc.).

   (e) A classe `FaceFriends`, contendo o método `main()`. Nesta classe defina um vetor de objetos com `Contatos`. Em seguida, implemente um MENU para executar as seguintes operações:

   - Inserir um contato, especificando o subtipo e então requerendo os seus campos.
   - Imprimir todos os contatos.
   - Imprimir somente os familiares.

- Imprimir somente os amigos.
- Imprimir somente os colegas de trabalho.
- Imprimir os MELHORES amigos (`grau == 1`), os IRMÃOS (`parentesco.equals("irmão")`) e os COLEGAS de trabalho (`tipo.equals("colega")`).
- Imprimir os dados de um **único** contato, escolhido pelo índice. Antes de imprimir o contato escolhido, o programa deve também imprimir o tipo de contato ao qual aquele índice se refere (Amigos, Família ou Trabalho).

Fazer:

(a) **Modelagem do problema** (diagrama de classes).

(b) **Código**