Universidade Federal de Uberlândia

Faculdade de Computação
GGI030 – Programação Orientada a Objetos
2o. Semestre de 2018
Prof. Renato Pimentel

Laboratório 05 – Classes abstratas, polimorfismo

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.

```
1
2  class ContaCorrente {
3     private float saldo;
4     private int estado;  // 1=conta ativa 2=conta inativa
5     private int numConta;
6     private int senha;
7
8     public ContaCorrente( float val, int num, int pwd ) {
9        numConta = num;
10       senha = pwd;
11       saldo = val;
12       estado = 1; // conta ativa
13    }
14
15    public boolean debitaValor( float val, int pwd ) {
16       if ( pwd != senha )
17          return ( false ); //senha deve ser válida
18       if ( estado != 1 )
19          return ( false ); //conta deve ser ativa
20       if ( val <= 0 )
21          return ( false ); //val > 0
22       if ( val > saldo )
23          return ( false );
24
25       saldo -= val;
26       if ( saldo == 0 )
27          estado = 2; // torna conta inativa
28       return (true);
29    }
30
31    public void debitaValor( float val ) {
32       saldo -= val;
33    }
```

```
34
35      public float getSaldo( int pwd ) {
36          if ( senha == pwd )
37              return saldo;
38          else
39              return -1; // indicando que houve problema na senha
40      }
41
42      public void creditaValor( int pwd, float val ) {
43          if ( senha == pwd )
44              saldo += val;
45      }
46
47      protected int getEstado( int pwd ) {
48          if ( senha == pwd )
49              return estado;
50          else
51              return -1;
52      }
53
54      protected void setEstado( int pwd, int e ) {
55          if ( senha == pwd )
56              estado = e;
57      }
58
59      protected boolean isSenha( int pwd ) {
60          if ( senha == pwd )
61              return true;
62          else
63              return false;
64      }
65  }
```

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 nem ser reativada novamente. Essa regra está implementada no método `debitaValor()` da classe `ContaCorrente`.

2. Modifique o código para trabalhar com classes abstratas. No caso, `ContaCorrente` deverá ser uma classe abstrata. Implemente as classes `ContaEspecial` e `ContaComum`, derivadas de `ContaCorrente`, de acordo com a especificação:

   (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 `UsaBanco`, 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, 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 da classe `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**