Programação Orientada a Objetos

Profa. Mirela Teixeira Cazzolato

Monitor: Erik Oda Coulter

Lista de exercícios 4: Aula 06 – Herança e polimorfismo

1. Marque as seguintes afirmações como verdadeiras ou falsas. Justifique (apenas) as alternativas falsas.

a) Construtores das superclasses não são herdadas por subclasses

( ) Justificativa: _____.

b) Um relacionamento parte-de é implementado via herança

( ) Justificativa: _____.

c) Uma classe Carro tem um relacionamento é-um com as classes Volante e Freios

( ) Justificativa: _____.

d) Quando uma subclasse redefine um método da superclasse (usando a mesma assinatura), ela é dita por sobrecarregar o método da superclasse.

( ) Justificativa: _____.

2. Discuta de que maneira a herança promove a reutilização de software, economiza tempo durante o desenvolvimento de um programa e ajuda a evitar erros.

3. Elabore um aplicativo Java de simulação de controle de contas bancárias. Todas as contas devem ter o nome do cliente, número da conta, saldo e métodos para sacar e depositar o dinheiro.

- Especialize a classe para que ela possa representar diferentes tipos de contas: conta poupança, conta-corrente e conta investimento.
- Conta poupança possui um dia de rendimento fixo (para todos os clientes) e um método de calcular e atribuir novo saldo, dado o rendimento (%), invocado pelo administrador do sistema. Saques não podem deixar poupança negativa.
- Conta-corrente possui um limite que pode ser sacado depois do saldo estar zerado.
- Conta investimento só pode permitir saque após 30 dias do último depósito, e possui rendimento diário (%). O admin pode atualizar o saldo da conta passando o percentual de rendimento diário e o número de dias equivalente. Saques não podem deixar a conta investimento negativa.

Crie um aplicativo para testar a classe com um menu que receba o nome do usuário. Caso usuário (cliente) já exista, abra outro menu que o permita sacar, depositar, verificar dia de rendimento (caso conta seja do tipo poupança) ou atualizar o saldo da conta com o rendimento correspondente. Caso usuário não exista, abra opção de criar um dos três tipos de conta. Caso o usuário entrada seja "admin", liste todos os correntistas com seus respectivos tipos de conta.

4. Descreva e discuta como é feita a chamada de construtores em subclasses. O código abaixo irá funcionar? Justifique sua resposta.

```
public class TesteConstrutorPai{
   public TesteConstrutorPai(String nomePai){
        System.out.println("Construtor pai falho: "+ nomePai);
   }
}

public class TesteConstrutorFilho extends TesteConstrutorPai
{
   public TesteConstrutorFilho (){
        System.out.println("Construtor filho");
}

   public static void main(String args[]){
        TesteConstrutorFilho t = new TesteConstrutorFilho();
   }
}
```