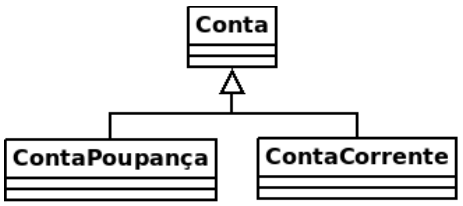
Prova 3 – PC2 / LPC2 – Eng. da Computação – Prof.

Aluno(a): _____ – Valor: 25 pontos – 12/02/2021

OBSERVAÇÕES:
– NÃO usar acentuação ou cedilha;
– Os arquivos .java devem estar dentro de um package com o nome do aluno;
– Para CADA questão é necessário fazer um PROGRAMA DE TESTE;
– Funções auxiliares devem ter seu código implementado;
– Não é permitido o uso de `java.util.List` e classes que o implementam;
– Enviar pelo SIGAA a pasta ***package*** compactada (usar ZIP). O arquivo compactado DEVE ter o nome do aluno.

QUESTÃO 1 (5 pontos)

Crie uma hierarquia de herança que um banco possa utilizar para representar dois tipos de conta: poupança e conta-corrente.



A classe `Conta` possui os atributos: número (autogerado) e saldo, também tem os métodos para retornar o saldo, para depositar e sacar. Não deve existir objeto conta genérica.

A classe `ContaCorrente` deve incluir atributos que representem o limite e a taxa (por ex. 0,1%) cobrada por transação de débito. Redefina o método de saque para descontar o valor da taxa a cada transação bem-sucedida.

A classe `Poupança` deve ter um atributo relacionado a taxa de rendimento mensal. A classe só deve permitir a criação de objetos da classe Poupança cujos valores dos atributos sejam válidos. Deve-se criar também um método para atualização do rendimento mensal da conta.

QUESTÃO 2 (6 pontos)

```
public class Questao2 {
	public static void main(String args[]) {
		Acervo acervo = new Acervo();
		// código, título, editora, nº paginas, edição
		Livro livro = new Livro(1,"Guerra dos Tronos","LeYa", 592, 4);
		int ret = acervo.add(livro);
		if (ret == 0)
			System.out.println("Livro adicionado com sucesso");
		else if (ret == 1)
			System.out.println("Falha ao cadastrar: título já existe");
		else if (ret == 2)
			System.out.println("Não foi possível inserir pois o vetor está cheio");
		else
			System.out.println("Falha desconhecida");
	}
}
```

Seja o trecho de programa acima, reimplemente o método `add` da classe `Acervo` de forma a este lançar uma exceção diferente para cada uma das condições acima. Rescreva o código da classe `Questao2` com o devido tratamento da exceção.

OBS: O método add deve ser implementado de tal forma que o tratamento de exceções seja obrigatório na classe que o use.

QUESTÃO 3 (6 pontos)
Suponha que você esteja desenvolvendo uma aplicação web para envio de cartões virtuais. Para tanto, crie uma classe abstrata chamada `CartaoWeb`, que representa os tipos de cartões da aplicação e conterá os atributos `remetente` e `destinatário`. Derive de `CartaoWeb` as classes `DiaDosNamorados` e `Aniversario`, que representam os cartões reais que existem na aplicação. Cada uma dessas classes deve conter um método construtor que recebe os nomes do remetente e do destinatário do cartão. Também deve implementar o método `mostrarMensagem()`, que mostra uma mensagem para a data comemorativa do cartão.

Por exemplo, esta poderia ser uma mensagem de um cartão de dia dos namorados:

> *"Querida Maria,*
> *Feliz Dia dos Namorados!*
> *Espero que esse tenha sido o único cartão do dia dos namorados que tenha ganhado nessa data!"*
> *De todo meu coração,*
> *João"*

Em uma classe `Teste` (`main`), crie um array de `CartaoWeb`. Insira de forma alternada, instâncias dos 2 tipos de cartões neste array. Após a inserção use um laço "for" para exibir as mensagens deste cartão chamando o método `mostrarMensagem()`.

OBS: Não usar **`instanceof`**

QUESTÃO 4 (8 pontos)
Existe no Java a interface `Comparable`, que é definida do seguinte modo:

```
public interface Comparable {
      public int compareTo(Object o);
}
```

Esta interface impõe a todas as classes que a implementam, a definição de um método que indica se uma instância da classe é maior, menor ou igual do que outra da mesma classe.

a) Defina uma classe `Funcionario` que implementa a interface `Comparable`. A classe tem os atributos `(id, nome, cpf, salario)` e um método fábrica para inicializá-los. O método exigido pela interface deverá retornar 1, 0, -1, se o salário do objeto funcionário receptor for maior, igual ou menor que o salário do objeto parâmetro, respectivamente. (2)
b) Supondo uma classe `Empresa` que contenha um vetor de funcionários, implemente o método `public Funcionario[] getFuncOrdemSalario() {…}` que retorna os funcionários da empresa em ordem crescente de salário. O método deve usar obrigatoriamente o método `compareTo` da letra (a). (6)

OBS: CUIDADO ao retornar referências de objetos internos. Proponha uma solução.

```
Funcionario f1 = Funcionario.getInstance(1,"Zé", "123.456.789-10",2000);
Funcionario f2 = Funcionario.getInstance(2,"Maria", "231.564.897-01",3000);
f1.compareTo(f2); // -1
f2.compareTo(f1); // 1
```

BOA PROVA!