Lista de Exercícios

MC302 – Programação Orientada a Objetos Instituto de Computação Universidade Estadual de Campinas		Polimorfismo (2) André Santanchè 2011

Primeiro Bloco de Questões

Dadas as seguintes interfaces:

Representa um animal

```
public interface Animal {
    public String getNomeEspecie();
    public String getNomeAnimal();
}
```

getNomeEspecie	Retorna o nome da espécie do animal.
getNomeAnimal	Retorna o nome do animal.

Representa um conjunto de rotinas utilitárias

```
public interface Ferramentas {
    public Animal[] filtraEspecie(Animal[] completo, String especieFiltrar);
    public String[] classificaEspecies(Animal[] completo);
}
```

filtraEspecie	Recebe como parâmetro um vetor contendo animais, que podem ser de várias espécies diferentes, e retorna um vetor que contém apenas os animais cuja espécie é especificada no parâmetro "`especieFiltrar`". Se não houver nenhum animal da espécie especificada, retorna um vetor com zero posições.
classificaEspecies	Recebe como parâmetro um vetor contendo animais e retorna um vetor de Strings contendo o nome de todas as espécies que foram encontradas no vetor recebido como parâmetro. Cada nome de espécie só aparece uma vez no vetor de saída.

Questão 1

Escreva um método que receba dois parâmetros:

- um vetor A de objetos que implementam a interface `Animal` representando diversos animais
- um objeto que implementa a interface Ferramentas

O método deve contabilizar o número de animais disponíveis em cada uma das espécies e retornar os resultados como um vetor de objetos da classe `Resultado` (apresentada abaixo). Cada objeto conterá uma espécie e o número de animais da mesma contabilizados. Devem ser consideradas apenas as espécies cujos animais estão presentes no vetor.

```java
public class Resultado {
    private String nomeEspecie; // nome da especie
    private int quantidade; // quantidade de animais da especie
    public Resultado(String nomeEspecie, int quantidade) {
      this.nomeEspecie = nomeEspecie;
      this.quantidade = quantidade;
    }
    public String getNomeEspecie() {
      return nomeEspecie;
    }
    public int getQuantidade() {
      return quantidade;
    }
}
```

Nesta questão basta implementar o método, não é necessária a especificação da classe.

Questão 2

Dada a classe `ItemOrcamento` que representa um item de um orçamento:

```java
public class ItemOrcamento {
    private String historico; // historico do item
    private float valor; // valor do item
    public ItemOrcamento(String historico, float valor) {
        this.historico = historico;
        this.valor = valor;
    }
    public String getHistorico() {
      return historico;
    }
    public float getValor() {
      return valor;
    }
}
```

Escreva uma classe herdeira de `ItemOrcamento` denominada `ItemOrcamentoComplexo` que mantenha um vetor com subitens de orçamento que podem ser da classe `ItemOrcamento` ou da classe `ItemOrcamentoComplexo`. A classe implementa os seguintes métodos:

Construtor	Além dos parâmetros da superclasse, recebe como parâmetro o vetor com os subitens de orçamento.
getValor	Sobrescreve o método da superclasse, retornando a soma de valores de todos os subitens de orçamento.
encontraItem	Recebe como parâmetro o histórico de um subitem (String) e retorna o objeto correspondente ao subitem que possui este histórico, se existir. Se não existir retorna null.

Questão 3

Um jardim zoológico definiu a seguinte interface que estende a interface Animal:

```
public interface AnimalOrcamento extends Animal {
    public ItemOrcamentoComplexo orcamentoGastosAnimal();
}
```

O método `orcamentoGastosAnimal` retorna o orçamento para gastos de um animal no zoológico.

O zoológico deseja saber quais de seus animais têm a "vacina W" prevista no seu orçamento.

Escreva um método que receba como parâmetro um vetor de objetos que implementam a interface `AnimalOrcamento` representando todos os animais do zoológico e seus respectivos orçamentos.

O método deve retornar um outro vetor de objetos que implementam a interface `AnimalOrcamento` apenas com aqueles animais que possuem um subitem com histórico "vacina W" prevista no seu orçamento.

Nesta questão basta implementar o método, não é necessária a especificação da classe.

Segundo Bloco de Questões

Um banco possui um sistema onde é definida a seguinte classe que representa um correntista e o saldo de sua conta bancária:

```
public class Correntista {
    private String cpfCliente;  // cpf do correntista
    private float saldo;  // saldo da conta
    public Correntista(String cpfCliente, float saldo) {
      this.cpfCliente = cpfCliente;
      this.saldo = saldo;
    }
    public String getCPFCliente() {
      return cpfCliente;
    }
    public float getSaldo() {
      return saldo;
    }
    public void setSaldo(float saldo) {
      this.saldo = saldo;
    }
}
```

Além disto, o sistema define as seguintes interfaces:

Representa um movimento (débito ou crédito) na conta de um correntista	
`public interface MovimentoConta { public String getCPFCorrentista(); public float getValorMovimento(); }`	
getCPFCorrentista	Retorna o CPF do correntista em cuja conta o movimento será aplicado.
getValorMovimento	Retorna o valor do movimento (positivo se for crédito e negativo se for débito).

Representa uma rotina utilitária	
`public interface OperacoesBanco { public Correntista encontraCorrentista(Correntista todosCorrentistas[], String cpfProcurado); }`	
encontraCorrentista	Procura no vetor `todosCorrentistas` o correntista cujo CPF é igual ao informado no parâmetro `cpfProcurado`. Se o encontrar, retorna seu respectivo objeto através do método, caso contrário retorna null.

Questão 4

Escreva um método que receba três parâmetros:

- um vetor C de objetos da classe `Correntista` representando os correntistas de um banco;
- um vetor M de objetos que implementam a interface `MovimentoConta` representando o movimento de diversos correntistas em um banco;
- um objeto que implementa a interface `OperacoesBanco`.

O método deve atualizar o saldo dos correntistas do vetor C com os movimentos que estão no vetor M. Observe que cada movimento é referente a apenas um dos correntistas.

Nesta questão basta implementar o método, não é necessária a especificação da classe.

Questão 5

Dada as classes a seguir:

Representa o total de despesas de um mês

```
public class DespesaMes {
    private int mes;  // mes da despesa
    private float valor;  // valor da despesa
    public DespesaMes(int mes, float valor) {
      this.mes = mes;
      this.valor = valor;
    }
    public int getMes() {
      return mes;
    }
    public float getValor() {
      return valor;
    }
}
```

Representa o total de despesas de um dia

```
public class DespesaDia extends DespesaMes {
    private int dia;  // dia da despesa
    public DespesaDia(int dia, int mes, float valor) {
      super(mes, valor);
      this.dia = dia;
    }
    public int getDia() {
      return dia;
    }
}
```

Escreva uma classe que representa todas as despesas de um indivíduo. Ela mantém um vetor onde podem ser registradas tanto despesas de um dia (`DespesaDia`), quanto despesas de um mês (`DepesaMes`). A classe implementa os seguintes métodos:

Construtor	Recebe como parâmetro o CPF e um vetor com as despesas de um indivíduo e as guarda.
getCPF	Retorna o CPF do indivíduo.
totalizaMes	Recebe um parâmetro com um mês (int) e retorna um objeto da classe `DespesaMes` onde estará registrada a soma de todas as despesas que o indivíduo fez naquele mês.

Questão 6

Dada a seguinte classe que representa os dados de um correntista, mais as despesas previstas para o mesmo.

```
public class CorrentistaDespesa extends Correntista {
    private DespesasIndividuo despesasPrevistas;  // despesas previstas
    public CorrentistaDespesa(String cpfCliente, float saldo,
                              DespesasIndividuo despesas) {
        super(cpfCliente, saldo);
        this.despesasPrevistas = despesas;
    }
    public DespesasIndividuo getDespesasPrevistas() {
        return despesasPrevistas;
    }
}
```

Escreva um método que receba como parâmetro um vetor de objetos da classe `CorrentistaDespesa`. Este método deve retornar outro vetor da classe `CorrentistaDespesa` apenas com aqueles correntistas cujas despesas previstas para março não sejam maiores que o saldo da conta.

Nesta questão basta implementar o método, não é necessária a especificação da classe.