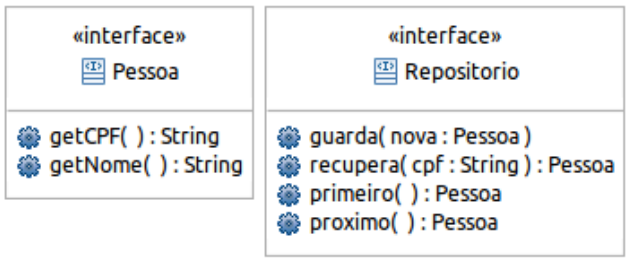
Lista de Exercícios

MC302 – Programação Orientada a Objetos Instituto de Computação Universidade Estadual de Campinas		Polimorfismo André Santanchè 2015

Questão 1

Dada as seguintes interfaces:



Pessoa - representa genericamente uma pessoa	
getCPF	retorna o CPF da pessoa
getNome	retorna o nome da pessoa
tipo	recebe como parâmetro o número da cadeira e retorna 'F' se for uma cadeira para fumantes e 'N' se for para não fumantes

Repositorio - representa genericamente um repositório	
guarda	guarda uma pessoa
recupera	recupera pessoa com o CPF informado
primeiro	se desloca para a primeira pessoa e a retorna
proximo	se desloca para a próxima pessoa e a retorna

Escreva uma classe denominada utilitários que possua os seguintes métodos:

duplica	Recebe como parâmetro dois objetos que implementam a interface Repositorio A e B e copia todas as pessoas do repositório A para o repositório B.
diferenca	Recebe como parâmetro três objetos que implementam a interface Repositorio A, B e C e coloque no repositório C todas as pessoas de A que não estiverem em B.

Questão 2

Um órgão de levantamento meteorológico possui equipamentos para medir a pluviosidade (pluviômetros), onde cada unidade é representada em um programa de computador por um objeto da classe:

```
public class Pluviometro
{
```

```
  protected String tipo;
  public Pluviometro(String tipo)
  {
    :
  }
  public String getTipo()
  {
    :
  }
  public int getPeso()
  {
    :
  }
  public int getCapacidade()
  {
    :
  }
}
```

Construtor	Recebe como parâmetro o tipo de equipamento.
getTipo	Retorna o tipo do pluviômetro.
getPeso	Retorna o peso do pluviômetro em quilos. Este peso é calculado automaticamente pela classe a partir do tipo.
getCapacidade	Retorna a capacidade do pluviômetro em mililitros. Esta capacidade é calculada automaticamente pela classe a partir do tipo.

Os pluviômetros são carregados por caminhões que, no programa de computador, são representados genericamente por objetos da classe `Caminhao` (esta classe não deve ser implementada nesta questão). A classe define os seguintes métodos:

Construtor	Recebe como parâmetro a quantidade de equipamentos que o caminhão irá carregar.
inserePluviometro	Recebe como parâmetro um objeto da classe `Pluviometro` e o coloca dentro do caminhão se a capacidade do mesmo permitir.

Cada objeto da classe representa um caminhão. Esta classe não possui nenhum atributo e seus métodos não possuem implementação, pois serão implementados nas subclasses.

Escreva duas classes herdeiras da classe `Caminhao` que representam dois tipos deste veículo com capacidades diferentes:

Caminhão Alfa

Consegue carregar no máximo 5 toneladas de pluviômetros, independente da quantidade e tipo.

Caminhão Beta

Consegue carregar qualquer quantidade e peso de pluviômetros, no entanto, não é capaz de carregar mais do que um pluviômetro de cada tipo.

Ambas as classes devem sobrescrever o método `inserePluviometro`.

Questão 3

Dada uma classe denominada `Controle`, definida em um sistema de controle de transporte de pluviômetros, que possui dois métodos estáticos:

`leString()`	Solicita ao usuário uma String pelo teclado e retorna através do método (função tipo String).
`leInteito()`	Solicita ao usuário um valor inteiro pelo teclado e retorna através do método (função tipo int).

Escreva uma classe herdeira de `Controle` que acrescente um método estático responsável pela seleção do caminhão mais apto a distribuição de pluviômetros. O caminhão será aquele capaz de conduzir pluviômetros cuja soma de capacidade seja a maior, independente do tipo de cada um deles.

O programa irá solicitar uma lista de caminhões. Para cada caminhão ele pergunta:

- Tipo do caminhão (‘Alfa’ ou ‘Beta’);
- número de pluviômetros a ser transportados;
- lista dos pluviômetros a ser transportados no caminhão (para cada pluviômetro é digitado apenas seu tipo).

A lista encerra quando é digitado ‘Fim’ para o Tipo do caminhão.

Depois de digitada toda a lista o programa imprime na tela (`System.out`) os seguintes dados do caminhão mais apto:

- tipo do caminhão (Alfa ou Beta);
- lista dos pluviômetros transportados no caminhão.

Em ambas as questões, devem ser criados os atributos que se fizerem necessários.

Bloco de Questões A

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 A.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 A.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 A.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.

Bloco de Questões B

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 B.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 B.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 B.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.