Exercícios propostos POO (herança e interfaces)

O seguinte conjunto de exercícios propostos resulta de uma recolha de exercícios dados em vários anos no curso de POO nos cursos do DCTI, ISCTE-IUL e criados ou adaptados pelos docentes da disciplina. Dado que o ênfase do programa varia ligeiramente consoante o trabalho final e com as alterações sofridas pela linguagem Java algumas das questões adaptam-se melhor que outras ao ano corrente, no entanto, consideramos que todas podem ser úteis para exercitar a programação.

---

1. Desenvolva uma classe que permita representar números complexos. Um complexo tem uma parte real e uma parte imaginária que são números decimais. Assim, a classe Complexo deve permitir as seguintes operações:

- Construir um complexo dados os valores da parte real e da parte imaginária;
- Construir um complexo dado outro complexo;
- Construir um complexo com módulo igual a zero;
- Saber a parte real;
- Saber a parte imaginária;
- Saber o módulo;
- Saber o ângulo;
- Devolver um complexo igual ao seu conjugado;
- Mostrar um complexo no ecrã;
- Somar um complexo, dado como argumento, à instância implicita;
- Construir uma String a partir de um complexo, método public String toString(). Para um complexo com parte real 1.0 e parte imaginária 2.0 o output deverá ser a String: "1.0 + 2.0i".

Comece por fazer um programa para testar todas as rotinas da classe.

---

2. Desenvolva uma classe que permita representar uma pessoa, de acordo com as seguintes indicações:

- A classe deve denominar-se Contacto;
- Deve incluir apenas um construtor, onde são passados o nome e o telefone da pessoa em questão;
- Deve disponibilizar, através de dois inspectores, a consulta do nome e telefone.
- Deve disponibilizar dois modificadores:
    - public void modificaTelefone(final int telefone), que muda o telefone da pessoa;
    - public void modificaNome(final String nome), que modifica o nome da pessoa.
- Deve redefinir o método equals e implementar o interface Comparable, sendo comparação por ordem alfabética de nome
- Inclua a possibilidade de pessoas com o mesmo nome serem ordenadas pelo número de telefone
- Inclua a possibilidade de ter vários telefones associados à mesma pessoa.

Comece por fazer um programa para testar todas as rotinas da classe.

---

3. Desenvolva um programa que permita converter Escudos em Euros. O seu programa deverá receber um valor em escudos **como argumento de entrada** e apresentar o correspondente valor em Euros na consola, devidamente formatado. Para esse efeito consulte as seguintes classes da API do Java.:

DecimalFormat, DecimalFormatSymbols, NumberFormat

Comece por fazer um programa para testar todas as rotinas da classe.

---

4. a) Defina completamente a classe Aluno que representa um aluno numa disciplina. Esta classe deve guardar o número (um inteiro), o nome (uma cadeia de caracteres) e a nota do aluno (um inteiro), e deve ainda permitir as seguintes operações:

- Construir um aluno, dados o número, o nome e a nota do aluno. O número do aluno deve ser positivo, o nome não pode ser null, nem vazio, e a nota também deve ser um inteiro positivo;
- Saber o número;
- Saber o nome;
- Saber a nota;
- Alterar o número, dado um novo número de aluno. O novo número de aluno tem de ser positivo;
- Alterar o nome, dado um novo nome. O novo nome não pode ser null, nem vazio;
- Alterar a nota, dada uma nova nota. A nova nota tem de ser um inteiro positivo;
- Mostrar o aluno.

4.b) Acrescente uma variável à classe Aluno indicadora do número de instâncias que são criadas da classe. Defina um método que permita consultar o número de instâncias criadas.

4.c) Explique a particularidade da variável que definiu na alínea (b), por oposição às variáveis definidas na alínea (a).

4.d) Efectue as alterações necessárias ao código no módulo físico Aluno.java, bem como à estrutura de directórios onde o módulo físico é guardado, de forma a ficar incluído num pacote denominado poo.alunos.

5.a) Complete a definição da classe Pauta, criando ou alterando em conformidade a estrutura de directórios que utiliza para guardar o respectivo módulo físico (o ficheiro Pauta.java).

```
package poo.alunos.classificacoes;

import poo.alunos.Aluno;

public class Pauta {
   public voidinsere(final Aluno novo_aluno) {
          // a completar...
   }
   public void removeAlunoComNome(final String nome) {
          // a completar...
   }
   public boolean estáVazia() {
          // a completar...
   }
   public void mostra() {
          // a completar...
   }
   public int capacidade() {
```

```
        assert cumpreInvariante();
        return alunos.length;
    }
    private static final int capacidade_inicial = 20;
    private Aluno[] alunos = new Aluno[capacidade_inicial];
    private int número_de_alunos = 0;
    private boolean cumpreInvariante() {
        // a completar...
    }
}
```

Comece por fazer um programa para testar todas as rotinas da classe.

6.a) Considere as novas classes Tuna, definida no módulo Tuna (ficheiro Tuna.java) e Pauta, definida no módulo Pauta (ficheiro Pauta.java, atenção: não grave por cima do ficheiro Pauta.java da alínea anterior). Realize as alterações necessárias (incluindo no ficheiro TesteDePautaMusical.java), sem mudar o nome de nenhuma das classes, para que o código seguinte funcione.

```
public class TesteDePautaMusical {
    public static void main(String[] argumentos) {
        Pauta pauta = new Pauta();
        Tuna tuna = new Tuna();
        Pauta frere_jacques = new Pauta();
        frere_jacques.insere("dó");
        frere_jacques.insere("ré");
        frere_jacques.insere("mi");
        frere_jacques.insere("dó");
        frere_jacques.insere("dó");
        frere_jacques.insere("dó");
        frere_jacques.insere("ré");
        frere_jacques.insere("mi");
        frere_jacques.insere("fá");
        frere_jacques.insere("sol");
        Aluno zacarias = new Aluno(1, "Zacarias", 1);
        tuna.insere(zacarias);
        tuna.insere(frere_jacques);
        tuna.mostra();
        pauta.insere(zacarias);
        pauta.mostra();
    }
}
```

6.b) Explique o que é o *default package* em JAVA.

6.c) Complete os ficheiros Tuna.java e Pauta.java com comentários de documentação adequados.

---

7. Construa uma classe Menu de modo a que, quando executado o método mostra(), apareça uma janela com botões que permita escolher uma das opções com que o menu foi criado, ou a opção Sair. Para mostrar o menu numa janela pesquise na biblioteca padrão JOptionPane o método showOptionDialog(...). Os argumentos

parentComponent e icon deverão ter o valor null, neste caso. Os argumentos do tipo Object serão, neste caso, String.

8.a) Programe uma classe que permita instanciar objetos do tipo Pessoa. Cada pessoa tem um nome e uma idade. Providencie pelo menos um constructor, dois inspectores, a sobreposição do método toString() e um método para testar a igualdade entre duas pessoas (duas pessoas são iguais se têm o mesmo nome e idade).

8.b) Construa uma classe Funcionario como uma extensão da classe Pessoa. Um funcionário é uma pessoa com um vencimento base e um vencimento líquido (aquele que é efectivamente pago). Os constructores da classe deverão receber o valor do vencimento base, mas não o do vencimento líquido. Considere ainda que poderão existir vários tipos de funcionários (e.g. assalariados, sub-contratados), de cujo o tipo e o cálculo do vencimento líquido depende. Desta forma, considere o seguinte interface

Assalariado:

```
interface Assalariado {
   double IMPOSTO = 0.3;
   double calculaVencimentoLíquido();
}
```

Implemente as classes Administrador e TrabalhadorÀComissão. O vencimento líquido de um administrador é fixo, com deduções do seu vencimento base a uma taxa igual aAssalariado.IMPOSTO.

O vencimento líquido de um trabalhador por comissão na área de vendas é igual ao vencimento base mais um bónus por cada unidade de produto vendido, com posterior desconto de impostos a uma taxa igual a Assalariado.IMPOSTO. Os constructores destas classes deverão inicializar os atributos relevantes para a sua implementação.

8.c) Construa a título experimental uma lista de funcionários com vários administradores e trabalhadores por comissão. Percorra os elementos da lista com um iterador e implemente o cálculo dos vencimentos líquidos dos funcionários, mostrando-os na consola.

8.d) Desenvolva um menu para adicionar e apagar funcionários numa lista. Considere pelo menos quatro opções:

- Adicionar administrador
- Adicionar trabalhador por comissão
- Remover funcionário
- Listar funcionários e vencimentos

8.e) Defina a classe Name de modo a que o seguinte programa funcione correctamente ordenando os nomes por ordem crescente da sua chave (o número inteiro) e mostrando o resultado da forma indicada.

```
public static void main(String [] argv) {
   Scanner teclado = new Scanner(System.in);
   // Criação de um HashMap
   LinkedList<Pessoa> list = new LinkedList<Pessoa>();
   for(int i = 0; i < 5 ; i++) {
          String nome = teclado.nextLine();
          int bi = teclado.nextInt();
```

```
            list.add(new Pessoa(nome, bi));
        }
        System.out.println("Ordem Arbitrária");
        for (Pessoa pessoa: list) {
            System.out.println(pessoa);
        }
        System.out.println("__________");
        Collections.sort(list);
        System.out.println("Ordem Alfabética");
        for (Pessoa pessoa: list) {
            System.out.println(pessoa);
        }
    }
```

Exemplo do resultado pretendido:

```
Ordem Arbitrária
Manuel João
Maria José
Arlindo Pereira
Xavier Zingaro
Carlos Rui
__________
Ordem Alfabética
Arlindo Pereira
Carlos Rui
Manuel João
Maria José
Xavier Zingaro
```

---

9. Considere que está inteira e correctamente definida a classe Data, cuja parte pública tem a seguinte estrutura:

```
public class Data implements Comparable<Data> {
    /**
    Construtor que cria uma nova data, inicializada com o dia, mês
    e ano correntes (dia de hoje)
    */
    public Data(){...}
    /**
    Construtor que cria a data dia/mês/ano
    */
    public Data(int dia, int mês, int ano) {...}
    /**
    Devolve a Data n dias após a data guardada na instância
    implícita
    */
    public Data soma(int n) {...}
    /**
    …
    */
    public int compareTo(Data d){...}
    public boolean equals(Object d){...}
```

```
    public String toString() {...}
    public int getDia() {...}
    public int getMês() {...}
    public int getAno() {...}
    public boolean cumpreInvariante() {...}
}
```

No projecto em que está envolvido (criação de software para um supermercado), foi incumbido de criar uma parte da aplicação que gere as encomendas que os clientes fazem *online* (leia toda a alínea antes de começar a responder).

Considere que na análise foi prevista a existência de uma classe abstracta Produto que representa alguma informação que é relevante para esta tarefa, referente a um lote de produtos de supermercado. A classe Produto deve ter os seguintes atributos: código (inteiro positivo), nome e data de validade (do tipo Data, definido atrás). Nesta aplicação dois (lotes de) produtos são iguais se tiverem o mesmo código e a mesma data de validade.

9. a) Defina a classe Produto. Nesta fase defina apenas os atributos, um único construtor por classe que inicialize todos os seus atributos, o teste da condição invariante (cumpreInvariante), que deve verificar a validade de todos os atributos, os inspectores do código e data de validade e o método equals, que considera iguais dois produtos que tenham o mesmo código e data de validade. Sempre que dá entrada um lote de produtos de um tipo é criado um novo objeto de uma das subclasses de Produto (ver figura) e posto na lista de produtos disponíveis (há apenas uma lista de produtos disponíveis na aplicação). Dois produtos na lista de disponíveis podem ter o mesmo código, mas nunca podem ter, simultaneamente, o mesmo código e a mesma validade que outro produto que já se encontre na lista de disponíveis. Logo que é verificado que já não há mais produtos deste lote no supermercado, este é retirado da lista de disponíveis.

9. b) Crie o método boolean existeNaListaDeDisponíveis(Produto produto), da classe GestãoEncomendas. Assuma que está definida nesta classe a seguinte lista de produtos (que contém os produtos disponíveis):

```
private LinkedList<Produto> disponíveis = new LinkedList<Produto>();
```

9. c) Crie o método void insereProduto(Produto produto) ..., da classe GestãoEncomendas, que insere o produto pedido na lista de produtos disponíveis, caso não exista. Se o produto já existir deve lançar uma excepção ProdutoExistente (defina também a excepção). Deverá também ser criada uma classe designada Encomenda, representativa de um conjunto de produtos a entregar, a um determinado cliente (indicado pelo seu código de cliente), numa data de entrega específica. A encomenda pode também ser considerada fechada ou não (essa informação deve ser guardada). Ao proceder a uma encomenda o cliente cria uma nova encomenda, inicialmente sem produtos e sem data de entrega e não fechada, apenas com o seu nº de cliente (inteiro positivo), e vai seleccionando códigos de produtos, que devem ser procurados na lista de disponíveis e, caso existam, adicionados à encomenda. No final o cliente irá dizer qual a data de entrega e só depois esta poderá ser marcada como fechada.

9. d) Defina a classe Encomenda. Nesta fase defina apenas, os respectivos atributos, um único construtor que inicialize os seus atributos (o construtor recebe apenas informação sobre o código do cliente), o teste da condição invariante (cumpreInvariante) da classe, que deve verificar a validade de todos os atributos, o inspector do código do cliente.

Quando o cliente decide qual a data de entrega, deve ser verificada a lista de produtos dessa encomenda e garantido que a data de validade de todos é superior em mais de N dias à data de entrega. O número N é dado

pelo método abstracto nDiasValidadeMinima(). Caso a data de validade não seja superior à data de entrega em mais de N dias, deve procurar na lista de disponíveis se há um produto com o mesmo código que cumpra o requisito, caso exista deve substituir o produto, caso contrário deve retirar o produto que não cumpre o requisito e sinalizar a encomenda como incompleta (i.e. não fechada). Caso todos os produtos cumpram o requisito e não tenham sido retirados produtos, a encomenda deve ser fechada.

9. e) Crie o método void setDataEntrega(GestaoDeEncomendas gest_encomendas, Data data_entrega), da classe Encomenda que faz o processo descrito acima de verificação e fecho de uma encomenda. Crie e defina os métodos auxiliares que achar apropriados.

9. f) Crie a classe ProdutoFresco que contém um atributo extra, o número mínimo de dias entre a entrega e o fim da data de validade, definindo o seu construtor e a concretização do método abstracto da classe base.

---

10. Considere uma classe denominada Artigo, representativa de um artigo de uma revista. Um artigo caracteriza-se por um texto (o texto do artigo, que pode ser representada por uma classe do tipo String), um título, um autor, uma classificação e um estado. A classificação de um artigo é um valor inteiro compreendido entre 0 e 20 e que indica o mérito do texto escrito, atribuído pelo director editorial da revista.

O estado do artigo pode ser de “aceite”, “em análise” ou “recusado”. O estado do artigo é necessariamente aceite quando a classificação é superior a 15.

Considere ainda a classe Revista caracterizada pelos seguintes atributos: o título da revista, uma lista de artigos submetidos e uma lista de artigos definitivamente aceites.

10. a) Defina a classe abstracta Artigo e a classe Revista. Nesta fase defina apenas, para cada classe, os respectivos atributos, um único construtor por classe que inicialize os seus atributos e dois inspectores simples, um para o nome do autor do artigo e outro para o título da revista. O construtor do artigo inicializa sempre o estado com o valor "em análise" a menos que a classificação seja superior a 15. A revista começa sempre sem quaisquer artigos. Defina ainda as estruturas do tipo enumerado que considerar necessárias. Defina uma função que teste a condição invariante da classe e use-a no construtor.

10. b) Defina na classe Revista um método de classe (estático) que, dada uma lista de artigos, devolve uma lista ordenada, por ordem alfabética, e sem repetições de todos os nomes de autores presentes na lista de artigos dada.

10. c) Defina na classe Revista o método escreveAceites(...), que escreve na consola todos os artigos da lista de artigos definitivamente aceites ordenados por ordem alfabética de autor.

10. d) Defina nas classes Artigo e Revista os métodos para ler e escrever estes dados num ficheiro com etiquetas (XML).

---

11. Considere uma classe denominada Militante que representa um militante de um partido. Esta classe deve incluir os seguintes atributos: o número de militante, o nome do militante, o ano de inscrição no partido e a cidade

de recenseamento do militante.

Considere ainda uma classe Partido, que pretende representar um partido político. Um partido político é basicamente constituído pela sua designação e um conjunto de militantes. Considere que o partido não pode ter em nenhum momento menos que 10 militantes ao longo da sua existência. Considere ainda que o conjunto de militantes fundadores do partido é fornecido aquando da criação do partido, e assume-se que têm números de militantes compreendidos entre 1 e o número de militantes fundadores.

11. a) Defina a classe Partido e Militante. Nesta fase defina apenas os respectivos atributos, os constructores, o inspector e o modificador do número de militante na classe Militante.

11. b) Defina o método retiraMilitante na classe Partido onde dado um militante retira-o da lista se nela existir. Acrescente o(s) método(s) necessário(s) da classe Militante para a realização desta operação.

11. c) Suponha que o primeiro militante a inscrever-se no partido tem atribuído um número imediatamente a seguir ao número do último fundador, sendo os números dos outros militantes atribuídos sequencialmente por ordem crescente de inscrição. Defina o método inscreveMilitante na classe Partido.

11. d) Defina na classe Partido o método adequaNumeroMilitantes(). Uma vez que ao fim de vários anos o número de militante atribuído às novas inscrições pode ser mais alto que o número efectivo de militantes, este método atribui novos números aos militantes, de acordo com a sua sequência na lista, com excepção dos militantes com número inferior ou igual a 10, que mantém sempre os seus números originais de inscrição no partido.

---

12. Considere a classe Eleito, representativa de um militante eleito num órgão directivo do partido. A classe Eleito, regista um dado número de votos associado a um dado militante. Considere ainda a classe Orgao, representativa de um órgão directivo de um partido. Um partido tem três tipos de órgãos directivos:

As comissões temáticas, cada uma dedicada a um dado tema político e constituída por um conjunto de eleitos, incluindo um presidente e um vice-presidente. Os presidente e vice-presidente são os eleitos mais votados para esse órgão. Podem existir diversas comissões temáticas num partido, cada um dedicada a um determinado tema.

A direcção executiva do partido, constituída por um conjunto de eleitos e um conjunto de funcionários do partido que são militantes e dão apoio de secretariado à direcção. A direcção tem um presidente e um vice-presidente, que são os eleitos mais votados para esse órgão.

A assembleia geral, constituída por um conjunto de eleitos para a mesa da assembleia, um conjunto de militantes representativos dos delegados eleitos para a assembleia e os presidente e vice-presidente que são membros da mesa da assembleia.

12. a) Desenhe o diagrama UML de classes deste problema, incluindo no seu diagrama as classes do problema anterior. Não precisa definir os atributos nem os métodos das classes no seu diagrama UML, apenas o nome da classe e a sua relação com as outras classes representadas.

12. b) Defina as classes utilizadas no seu diagrama UML. Nesta fase defina para cada classe apenas os seus atributos e um construtor que com os argumentos necessários para inicializar todos os atributos da classe.

12. c) Considere o método calculaOrçamento que se pode aplicar a qualquer órgão directivo do partido. O orçamento de um órgão é a soma de uma constante definida para cada tipo de órgão do partido com o número de militantes (eleitos ou não eleitos) vezes uma constante de classe definida na classe Partido, e denominada ORÇAMENTO_POR_MILITANTE. Defina a operação nas classes que entender necessário, definindo para isso eventuais métodos adicionais nas classes que entender necessárias para implementar esta operação.

12. d) Defina o método orcamentoTotal na classe Partido, que calcula o orçamento total dos órgãos do partido.

---

13. Considere a seguinte definição das classes Aluno e Turma de uma escola secundária:

```
public class Aluno {
   protected String nome;
   protected int numero;
   public Aluno(final String nome, final int numero) {
         this.nome=new String(nome);
         this.numero=numero;
   }
}

class Turma {
   LinkedList<Aluno> alunos;
   public Turma(final LinkedList<Aluno> alunos) {
         this.alunos=alunos;
   }
}
```

13. a) Implemente a cópia profunda da classe Turma, realizando as alterações que forem necessárias nessa classe.

13. b) Suponha a existência de dois tipos de aluno, os alunos internos e externos. Um aluno interno é um aluno com um número de um quarto onde dorme. Um aluno externo é um aluno que apenas frequenta a escola em três dias da semana. Defina as classes representativas dos alunos interno e externo, incluindo os seus atributos e um constructor.

---

14. Considere uma classe designada AgenteCooperante que representa um agente cooperante em serviço num país estrangeiro. Essa classe deve incluir os seguintes atributos: o nome do agente, um conjunto não vazio de línguas que domina, um conjunto não vazio de países onde pode prestar serviços, o país onde está presentemente a prestar serviço e o salário base que aufere. Considere ainda a classe País, representativa de um país estrangeiro. Essa classe deve representar o nome do país, um conjunto de línguas faladas pelos seus cidadãos e um conjunto de agentes cooperantes. Considere ainda que o número de agentes cooperantes de um país não pode ser em nenhum momento maior que 10.

14. a) Defina as classes AgenteCooperante e País. Nesta fase defina apenas, para cada classe, os respectivos atributos, um único constructor por classe que inicialize os seus atributos e o teste da condição invariante de classe.

14. b) Defina na classe País o método custoIndividualBase, onde dado um agente cooperante, devolve o salário base desse agente cooperante se existir nesse país e zero se não existir. Defina os métodos adicionais que considerar necessários em qualquer das classes para a implementação desta função

14. c) Defina na classe País um método que retorna uma lista com todos os seus agentes capazes de falar todas as línguas desse país (cujo resultado pode ser eventualmente uma lista vazia). Defina os métodos adicionais que considerar necessários em qualquer das classes para a implementação desta função.

---

15. Considere a existência de três tipos de agentes cooperantes, o agente cooperante secreto, o agente cooperante da ONU e o agente cooperante em serviço parcial. Os agentes da ONU e em regime parcial têm uma morada de residência e um tempo pré-determinado de serviço previsto no país, medido em número de dias. Além disso, os salários mensais finais auferidos pelos agentes são calculados pela operação salárioMensal da seguinte forma, de acordo com o tipo de agente em causa:

- o agente cooperante secreto, cujo salário mensal é igual ao salário base acrescido de uma componente dada pela seguinte fórmula: $s*n/10$, onde s é o salário base do agente e n é o número de agentes cooperantes no seu país.
- o agente cooperante da ONU, cujo salário mensal é igual ao salário base acrescido de uma componente igual a 20% do salário base;
- o agente cooperante em regime parcial, cujo salário mensal é igual ao salário base mais uma quantia fixa dada pela seguinte fórmula: $10000/n*30$, onde n é o número pré-determinado de dias de serviço previsto no país.

15. a) Defina as classes adequadas para este problema, incluindo para cada classe proposta um constructor para a inicialização dos seus atributos, e alterando se necessário a classe AgenteCooperante.

15. b) Defina uma função na classe País designada custoMensal(), que calcula o custo mensal de cooperação com esse país, dada pela soma dos salários mensais dos agentes em cooperação nesse país.

---

16. Considere agora uma capacidade específica, definida pela possibilidade dos agentes auferirem de forma independente uma quantia extra mensal, através da prestação de serviços a países terceiros distintos do país que representam.

Para o caso do agente cooperante secreto, essa quantia é dada pelo número de serviços prestados vezes 20% do seu salário base, e para o caso dos agentes da ONU e cooperantes em regime parcial é dada pelo número de serviços prestados vezes 10% do seu salário base, com subtração posterior de 20% de impostos.

16. a) Modifique as classes anteriores de forma a integrar esta nova capacidade nos agentes, definindo para isso uma operação quantiaExtra, onde o número de serviços prestados é dado, e que devolve a quantia extra mensal auferida de um agente.

16. b) Defina uma função na classe País designada valoresPrevistosTotais, onde dado um número previsto de serviços prestados, calcula o valor total mensal previsto envolvido nas remunerações dos agentes, igual à soma dos salários mensais com as quantias extras mensais dos agentes.

---

17. Considere o seguinte código, em que temos um conjunto de valores. Cada valor guarda um número inteiro. Temos também um conjunto de variáveis. Uma variável pode ser um valor ou uma referência para um valor. Uma referência guarda o valor indirectamente, apontando para o item correspondente na lista de valores:

```
public static void main(String[] args) {
   Vector<Valor> valores;
   Vector<Variável> variáveis;
   valores.add(new Valor(10)); // 1a)
   valores.add(new Valor(20));
   valores.add(new Valor(30));
   // adicionando às variáveis uma referência para a
   // posição 0 que contém o valor 10
   variáveis.add(new Referência(valores.get(0))); // 1b)
   variáveis.add(new Referência(valores.get(1)));
   // adicionando às variáveis o valor 3
   variáveis.add(new Valor(3));
   variáveis.add(new Referência(valores.get(2)));
   variáveis.add(new Valor(5));
   int sum = 0;
   for (Variável : variáveis) { // 2)
      sum += op.getValor();
   }
   System.out.println("A soma de todas as variáveis é: " + sum);
}
```

17. a) Indique quais as modificações necessárias para corrigir os erros assinalados no código por "// 1a)", "// 1b)" e "// 2)", que dão origem às seguintes mensagens (não pode alterar o código existente, apenas adicionar código):

1a) The local variable valores may not have been initialized
1b) The local variable variáveis may not have been initialiazed
2) Syntax error on token Variável, identifier expected after this token

17. b) Sabendo que Variável é uma classe abstracta, defina as classes Variável, Valor e Referência de modo a que o código acima funcione correctamente.

17. c) Que alterações deveria fazer para que fosse possível invocar um método endereço() para qualquer variável, que, caso a variável fosse uma Referência devolveria o seu endereço, i.e. o índice na lista de valores, e caso fosse um valor lançaria a excepção VariávelNãoReferenciável (derivada de RunTimeException).

17. d) Escreva o código necessário para adicionar duas variáveis (quer sejam valores ou referências) e criar uma referência onde o resultado seja colocado.

---

18. Dada a classe:

```
public class Valor {
	private int valor;
	public Valor(int valor) {
		this.valor = valor;
```

```
        }
}
```

Qual seria o resultado da seguinte operação?

new Valor(3).equals(new Valor(3));

Discuta como poderia alterar esse resultado e se faria sentido fazê-lo.

---

19. Cada desenho (Desenho) é composto por um número arbitrário de gráficos (Gráfico). Os gráficos podem ser elementares (e.g., Círculo e Triângulo) ou compostos. Neste último caso, o gráfico na realidade consiste num conjunto de outros gráficos, que por sua vez tanto podem ser elementares como compostos.

Notas:

- Um gráfico não pode constar senão uma vez em cada GráficoComposto e em cada Desenho.
- Um gráfico não pode constar mais do que uma vez num dado Desenho, *mesmo que esteja em gráficos compostos diferentes*.
- Deve ser possível adicionar gráficos a um gráfico composto ou a um desenho.
- Da mesma forma, deve ser possível remover gráficos de um gráfico composto ou de um desenho.
- Deve ser possível passar conjuntos de gráficos pré-existentes ao construtor de GráficoComposto. Deve ser possível criar gráficos compostos e desenhos vazios, i.e., inicialmente sem qualquer gráfico.
- Defina as classes de forma a suportarem a estrutura indicada acima. Não precisa de declarar e implementar senão os métodos que permitem manipular essa estrutura, i.e., construtores e, no caso dos desenhos e gráficos compostos, métodos para adicionar e remover gráfico. Defina todos os atributos correspondentes à estrutura*. Justifique detalhadamente todas as suas opções.*
- Coloque as justificações na forma de comentários.
- Verifique todas a pré-condições. Indique as condições invariantes de instância e verifique-as

---

20. Considere a classe Humano. Esta classe representa seres humanos e as relações biológicas entre eles. Implemente a classe de modo a que seja possível executar o seguinte código:

```
import static java.lang.System.out;
public final class TesteDeHumano {
   public static void main(final String[] argumentos) {
      Humano joão = new Humano("João", Sexo.MASCULINO, null, null);
      Humano maria = new Humano("Maria", Sexo.FEMININO, null, null);
      Humano josé = new Humano("José", Sexo.MASCULINO, null, null);
      Humano ana = new Humano("Ana", Sexo.FEMININO, null, null);
      Humano rui = new Humano("Rui", Sexo.MASCULINO, joão, maria);
      Humano helena = new Humano("Helena", Sexo.FEMININO, josé, ana);
      Humano pedro = new Humano("Pedro", Sexo.MASCULINO, rui, helena);
      out.println("Árvore do Pedro:");
      pedro.mostraÁrvore();
      out.println();
      out.println("Árvore do Rui:");
```

```
        rui.mostraÁrvore();
    }
}
```

A execução deste código deverá produzir
Árvore do Pedro:
Pedro
    Ascendentes:
    Pai: Rui
    Pai: João
    Mãe: Maria
    Mãe: Helena
    Pai: José
    Mãe: Ana
    Descendentes:
Árvore do Rui:
Rui
    Ascendentes:
    Pai: João
    Mãe: Maria
    Descendentes:
    Filho: Pedro

Notas:

- Admita que o tipo enumerado Sexo está já definido.
- Um pai não pode ser senão do sexo masculino.
- Uma mãe não pode ser senão do sexo feminino.
- Um humano pode não ter qualquer dos pais. Isso significa simplesmente que esse pai (ou mãe) é desconhecido.
- Deve verificar as pré-condições de cada método desenvolvido.

---

21. Crie um programa que joga xadrês escolhendo aleatoriamente peças que se podem mover e fazendo um movimento aleatório dentro dos possíveis para essa peça.
