CESAR School – Ciências da Computação
Programação Orientada a Objetos – 3º período

QUESITO 1 (2,5 PONTOS): Implemente uma classe de nome TelefoneMovel, com visibilidade irrestrita, que possua as seguintes características descritas no quadro abaixo.

Atributos: Marca (string), capacidade da bateria em mAH (inteiro) e preço (número real). Todos os atributos devem ter visibilidade apenas na própria classe. Métodos set e get: para marca, visibilidade no pacote e na sub hierarquia; para capacidade da bateria, visibilidade apenas no pacote; para preço, visibilidade irrestrita. A marca deve ser imutável. Construtor: deve inicializar os três atributos e ter visibilidade apenas no pacote. Métodos: implementar um método calcularTempoUso, de visibilidade irrestrita, que receba como parâmetro um valor de potência média de uso (em W) e que retorne o tempo estimado de uso da bateria em horas, considerando a voltagem padrão igual a 12V, de acordo com a fórmula: **`tempoDeUso = (capacidadeDaBateria * VOLTAGEM_PADRAO / 1000) / potenciaMedia`**

REFERÊNCIA NO MATERIAL

POO-IntroducaoJavaEstruturado.pdf, pags. 24-26, 42-57; POO – Classes e objetos.pdf, pags. 04-17, 26-27; POO – Encapsulamento.pdf, pags. 13-17.

EXPLICAÇÃO SOBRE A SOLUÇÃO

Esta questão consiste basicamente em definir uma classe levando-se em conta os seguintes pontos:

- Como declarar uma classe: definir modificador de acesso, palavra reservada “class” e nome.
- Como declarar atributos: definir modificador de acesso, tipo e nome. No caso, os tipos são: String para marca, double ou float para preço e double, float, long ou int para capacidade da bateria.
- Como declarar constantes (no caso, VOLTAGEM_PADRAO): definir modificador de acesso, palavras reservadas static (opcional, mas desejável) final, tipo, nome e atribuição do valor inicial, no caso 12.
- O termo “imutável” para um atributo indica simplesmente que ele não deve ter método “set”.
- Como declarar construtor: definir modificador de acesso, colocar o nome da classe (padrão de definição de construtores em JAVA) e lista de parâmetros, com seus respectivos nomes e tipos. A implementação do construtor que “deve inicializar os três atributos”, conforme enunciado, é a atribuição dos parâmetros aos atributos da classe, usando-se o padrão this. nome_atributo = nome_parametro. Se o nome_parametro é diferente do nome_atributo correspondente, o “this” é opcional. Caso contrário, o “this” é obrigatório.
- Como declarar e implementar métodos simples: definir modificador de acesso, tipo de retorno (no caso double ou int), nome e lista de parâmetros, com seus respectivos tipos (no caso 1 parâmetro do tipo int ou double). A implementação é o cálculo do tempo de uso, implementado exatamente como colocado no enunciado.
- Os modificadores de acesso são definidos de acordo com o que foi pedido no enunciado para cada atributo, método, construtor e a própria classe, segundo a relação entre os termos colocados no enunciado e o modificador de acesso correspondente (ver tabela abaixo).

Termo no enunciado	Palavra reservada
“visibilidade irrestrita”	public
“visibilidade no pacote e na sub hierarquia”	protected
“visibilidade apenas no pacote”	Não se coloca palavra reservada, pois trata-se de visibilidade "default”
“visibilidade apenas na própria classe”	private

CRITÉRIO DE PONTUAÇÃO

Desconto de 0,2 a 0,3 por modificador de acesso errado. Desconto de 0,3 a 0,4 por declaração de construtor ou de método errada. Desconto de 0,2 por declaração de atributo errada. Desconto de 0,3 por implementação de método ou de construtor errada. Desconto de 0,2 por implementação errada de set/get para atributo imutável. Desconto de 0,1 por não declaração de constante para o valor de 12V. Desconto de 0,3 por declaração errada de classe. Desconto de 0,1 por declaração errada de constante. Estes critérios a princípio são aplicados mas podem ser flexibilizados.

RESPOSTA

A classe devidamente implementada. A resposta “padrão”, que inclui a forma usual de representação dos nomes dos atributos e parâmetros em construtores e set´s, segue abaixo.

```
public class TelefoneMovel {
	private static final int VOLTAGEM_PADRAO = 12;
	private String marca;
	private int capacidadeBateria;
	private double preco;
	protected String getMarca() {
		return marca;
	}
	int getCapacidadeBateria() {
		return capacidadeBateria;
	}
	void setCapacidadeBateria(int capacidadeBateria) {
		this.capacidadeBateria = capacidadeBateria;
	}
	public double getPreco() {
		return preco;
	}
	public void setPreco(double preco) {
		this.preco = preco;
	}
	TelefoneMovel(String marca, int capacidadeBateria, double preco) {
		this.marca = marca;
		this.capacidadeBateria = capacidadeBateria;
		this.preco = preco;
	}
	public double calcularTempoUso(double potenciaMedia) {
		return (capacidadeBateria * VOLTAGEM_PADRAO / 1000) / potenciaMedia;
	}
}
```

QUESITO 2 (1,5 PONTOS): Dado o código mostrado no QUADRO 3, indique as linhas onde há erros de compilação e descreva os motivos de tais erros.

REFERÊNCIA NO MATERIAL

POO – Encapsulamento.pdf, pags. 04-12, 16-24.

EXPLICAÇÃO SOBRE A SOLUÇÃO

Trata-se de questão para verificar o uso permitido ou proibido (no sentido do código não compilar) de classes e de elementos de classes (atributos, métodos e construtores) em função dos modificadores de acesso a estes associados, segundo as regras que foram estudadas, conforme resumo abaixo:

Modificador	Visibilidade / restrição de uso.
public	Visibilidade irrestrita, a classe ou seu elemento pode ser usada em qualquer contexto.
protected	Visibilidade no mesmo pacote ou na mesma sub hierarquia, o elemento pode ser usado por classes filhas, netas, etc., ou por classes do mesmo pacote.
sem modificador	Visibilidade no mesmo pacote, a classe ou seu elemento pode ser usado por classes do mesmo pacote.
private	Visibilidade na mesma classe, o elemento só pode ser usado na classe onde ele foi definido.
static	Restrição de uso: métodos static não podem usar métodos e atributos não static.
final	Restrição de uso: atributos final não podem ter valores a eles atribuídos após a linha de sua definição. Na prática, são constantes.

A resolução da questão consiste em identificar onde há classes ou elementos utilizados com visibilidades indevidas ou com restrições de uso, com as devidas justificativas colocadas de acordo com a tabela acima.

CRITÉRIO DE PONTUAÇÃO

No código apresentado existem 05 linhas com erro de compilação. Cada linha corretamente apontada e com o erro corretamente justificado pontua 0,3, totalizando 1,5. Pontuações intermediárias por linha são pertinentes, a depender do que esteja escrito na justificativa. Há desconto de 0,3 por linha incorreta apontada. Se não fosse assim, poderia se apontar todas as linhas como incorretas e sempre pontuar potencialmente pelo máximo possível.

RESPOSTA
PROPULSAO_A_VENTO = 3; - atributo é final
ta.checkUp(); - método é private
tipoPropulsao = 3; - atributo é default
checkUp(); - método é private
*altitude = altitude * (1 + fat/100); - método static acessando atributo de instância*

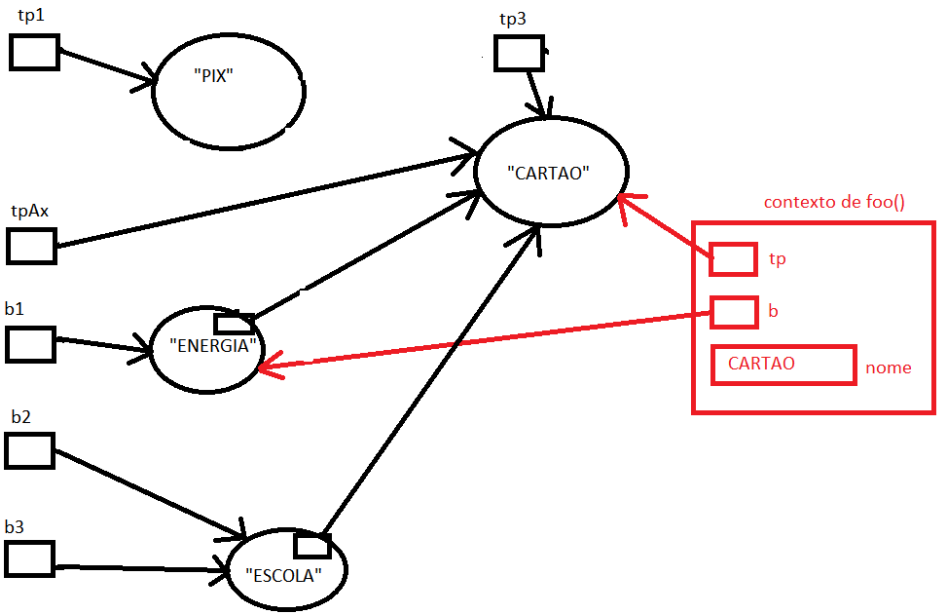
QUESITO 3 (1,5 PONTOS): Dado o código mostrado nos QUADRO 4 e QUADRO 5, desenhe o mapa de memória correspondente à execução do programa existente no QUADRO 5 quando a última linha é executada (estado final da execução), e indique as saídas no console.

REFERÊNCIA NO MATERIAL
POO-IntroducaoJavaEstruturado.pdf, pags. 27-28; POO – Classes e objetos.pdf, pags. 17-25, 28-29.

EXPLICAÇÃO SOBRE A SOLUÇÃO
A questão trata da relação entre variáveis de tipos não primitivos e objetos. Aquelas armazenam ponteiros (referências, endereços de memória) para objetos instanciados através do new. Quando atribuímos um new, o endereço de memória do objeto alocado pelo new é copiado para a variável. Quando atribuímos outra variável, o conteúdo dela (um endereço de memória) é copiado para a para a variável. A ideia é ir desenhando as variáveis declaradas, os objetos criados (e seus conteúdos principais) e as referências, que podem ser setas ou valores arbitrários que representam endereços de memória de objetos instanciados. Na questão, há a chamada a um método com passagem de parâmetros, que deve ser considerada uma atribuição, onde o lado direito é o valor ou variável passada e o lado esquerdo é o parâmetro definido no método. O escopo das variáveis do método pode aparecer na solução, embora no fim do código estas não mais existam. As saídas no console devem ser avaliadas observando-se os conteúdos dos objetos referenciados pelas variáveis que aparecem nos Sysouts.

CRITÉRIO DE PONTUAÇÃO
No código apresentado existem 05 Sysouts. Cada Sysout correto, NA ORDEM CORRESPONDENTE à ORDEM em que aparecem os Sysouts no código, vale 0,2. Se mais de 05 saídas forem indicadas, as saídas de ordem superior a 5 serão ignoradas. Se menos de 05 saídas forem indicadas, serão avaliadas até a quantidade apresentada, com pontuação eventualmente proporcional. O mapa de memória correto vale 0,5, e será pontuado através de análise específica.

RESPOSTA
Saída no console
ESCOLA
PIX
CARTÃO
ENERGIA
CARTÃO
Mapa de memória



QUESITO 4 (0,5 PONTO): Dado o código mostrado no QUADRO 1, indique as linhas onde há erros de compilação e descreva tais erros.

REFERÊNCIA NO MATERIAL
POO-IntroducaoJavaEstruturado.pdf, pags. 24-26, 36-41.

EXPLICAÇÃO SOBRE A SOLUÇÃO
A solução passa por aplicar as regras de conversão implícita de tipos primitivos. A regra geral prega que um tipo "mais largo" não pode ser atribuído a um tipo "mais curto", sob pena do código não compilar. Outra regra diz que um boolean não pode ser atribuído a nenhum tipo numérico nem caractere, e vice-versa. A "largura" dos tipos numéricos é (do menos largo para o mais largo): byte – short – int – long – float – double. Lembrar que constantes em JAVA são tipadas e têm representações próprias (ex. constantes no formato N.N são do tipo double). A justificativa para o erro de compilação é que um tipo A não "cabe" no tipo B ou "A pra B" (não permitido).

CRITÉRIO DE PONTUAÇÃO
Na questão há 05 linhas com erros de compilação. Cada erro identificado e corretamente justificado vale 0.1. Há desconto de 0,1 por linha incorreta apontada. Se não fosse assim, poderia se apontar todas as linhas como incorretas e sempre pontuar potencialmente pelo máximo possível.

RESPOSTA
2 - double pra float
3 - float pra int
4 - double pra float
6 - boolean pra float
8 - long para byte

QUESITO 5 (0,5 PONTO): Dado o código mostrado no QUADRO 2, indique as linhas onde há perda de informação quando o executamos, e descreva de forma QUALITATIVA as perdas observadas. Se foi a parte inteira que ficou inconsistente na variável de destino, se foram as casas decimais que se perderam, ou se foram ambas as situações.

REFERÊNCIA NO MATERIAL
POO-IntroducaoJavaEstruturado.pdf, pags. 24-26, 29-35.

EXPLICAÇÃO SOBRE A SOLUÇÃO
A solução passa por analisar se as conversões explícitas (de tipos mais largos em tipos mais curtos de maneira forçada, através do cast) implicam em perda de informação, que ocorre quando há conversão de tipos decimais COM VALORES DECIMAIS SIGNIFICATIVOS para tipos inteiros. Nestes casos há perda da parte decimal. Também há perdas quando a parte inteira do tipo mais largo está fora da faixa de valores do tipo mais curto. Nestes casos há perda da parte inteira.

CRITÉRIO DE PONTUAÇÃO
Na questão há 05 linhas com problemas de perda de informação. Cada erro identificado e corretamente apontado (parte inteira e/ou parte decimal perdida) vale 0.1. Linhas. Há desconto de 0,1 por linha incorreta apontada. Se não fosse assim, poderia se apontar todas as linhas como incorretas e sempre pontuar potencialmente pelo máximo possível.

RESPOSTA
2 - partes int e dec
5 - parte dec
9 - parte dec
10 - parte dec
11 - parte dec

QUADROS

QUADRO 1

```
1           double d = 2.11F;
2           float x = d;
3           int a = x;
4           x = 2.0;
5           boolean y = false;
6           d = y;
7           long l = a;
8           byte b = 12L;
9           short s = b;
10          a = s;
```

QUADRO 2

```
1           double d = 300000.11;
2           short s = (short)d;
3           byte b = (byte)11.0;
4           float f = (float)d;
5           int x = (int)d;
6           f = 22.99F;
7           int g = 10000;
8           double v = g * g + 0.1;
9           g = (int)v;
10          long a = (long)f;
11          a = (long)v;
```

QUADRO 3

```
package br.com.cesar.school.cc.poo.prova1;
public class TransporteAereo {
	int tipoPropulsao;
	protected void decolar() {
		checkUp();
		System.out.println("Subindo...");
	}
	public void pousar() {
		System.out.println("Pousando...");
	}
	private void checkUp() {
		System.out.println("Checando tipo propulsão");
		System.out.println(tipoPropulsao);
	}
}
```

```java
package br.com.cesar.school.cc.poo.prova1;
public class ModificadoresAcesso {
	private static final int PROPULSAO_A_JATO = 1;
	private static final int PROPULSAO_A_VENTO = 2;
	static void boo() {
		TransporteAereo ta = new TransporteAereo();
		PROPULSAO_A_VENTO = 3;
		ta.tipoPropulsao = PROPULSAO_A_JATO;

	}
	public static void foo(TransporteAereo ta) {
		ta.decolar();
		ta.checkUp();
	}
}
```

```java
package br.com.cesar.school.cc.poo.prova1.interno;
import br.com.cesar.school.cc.poo.prova1.TransporteAereo;
public class Aviao extends TransporteAereo {
	private double altitude;
	protected void aterrar() {
		decolar();
		pousar();
		tipoPropulsao = 3;
		checkUp();
	}
	static void calibrarAltitude(double fat) {
		altitude = altitude * (1 + fat/100);
	}
}
```

QUADRO 4

```java
public class TipoPagamento {
	String nome;
	public TipoPagamento(String nome) {
		this.nome = nome;
	}
}
public class Boleto {
	String descricao;
	private TipoPagamento tipoPagamento;
	TipoPagamento getTipoPagamento() {
		return tipoPagamento;
	}
	void setTipoPagamento(TipoPagamento tipoPagamento) {
		this.tipoPagamento = tipoPagamento;
	}
}
```

QUADRO 5

```
public class ProgramaBoleto {
	public static void main(String[] args) {
		TipoPagamento tp1 = new TipoPagamento("CASH'
		TipoPagamento tp3 = tp1;
		TipoPagamento tpAx = null;
		Boleto b1 = new Boleto();
		b1.descricao = "ENERGIA";
		b1.setTipoPagamento(tp1);
		Boleto b3 = new Boleto();
		Boleto b2 = b3;
		b3.descricao = "ESCOLA";
		b3.setTipoPagamento(tp3);
		tp1.nome = "PIX";
		System.out.println(b2.descricao);
		tpAx = b3.getTipoPagamento();
		System.out.println(tpAx.nome);
		foo(b1, "CARTÃO");
		tp3 = b1.getTipoPagamento();
		b2.setTipoPagamento(tp3);
		tpAx = b2.getTipoPagamento();
		System.out.println(tpAx.nome);
		b2 = b3;
		System.out.println(b1.descricao);
		tpAx = b1.getTipoPagamento();
		System.out.println(tpAx.nome);
	}
	static void foo(Boleto b, String nome) {
		TipoPagamento tp = new TipoPagamento(nome);
		b.setTipoPagamento(tp);
	}
}
```