CESAR School – Ciências da Computação
Programação Orientada a Objetos – 1ª prova **TURMA_____**

NOME:_____

QUESITO 1 (1,0 PONTO): Dado o código mostrado no QUADRO 1, indique as linhas onde há erros de compilação ou onde há perda de informação quando o executamos. Nesta última hipótese, indique que perda(s) ocorre(m).

QUESITO 2 (1,0 PONTO): Dado o código mostrado no QUADRO 2, indique as linhas onde há erros de compilação e descreva os motivos de tais erros.

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

QUESITO 4 (2,5 PONTOS): Implemente as classes especificadas abaixo, considerando que elas estão no mesmo pacote.

Preâmbulo:
Usar para definir os tipos de atributos, de parâmetros e de retornos de métodos, as seguintes premissas:

- Usar para texto o tipo nativo do JAVA que caracteriza uma cadeia de caracteres.
- Usar para inteiros um tipo primitivo do JAVA, que represente inteiros, com 32 bits.
- Usar para números reais o tipo primitivo do JAVA, que represente números decimais, com 64 bits.

Classe *Motor*, de visibilidade irrestrita.
Atributos: Modelo (texto), potência (número real) e tempo de vida (inteiro). Todos os atributos devem ter visibilidade apenas na própria classe.
Métodos set e get: para modelo, visibilidade apenas no pacote; para potência, visibilidade no pacote e na sub hierarquia; para tempo de vida, visibilidade irrestrita. O modelo é imutável.
Construtores: deve haver dois construtores, sendo um inicializando os três atributos, e o outro inicializando modelo e tempo de vida. Os construtores devem ter qualquer visibilidade que não seja apenas na própria classe. Promover reuso entre construtores, onde o que possui mais parâmetros reusa o que possui menos parâmetros.
Classe *MotorEletrico*, de visibilidade irrestrita, deve ser subclasse de *Motor*.
Atributos: voltagem (número real). Todos os atributos devem ter visibilidade apenas na própria classe.
Métodos set e get: Para voltagem, visibilidade no pacote e na sub hierarquia. A voltagem deve ser imutável.
Construtores: deve haver um único construtor que inicializa todos os atributos da classe. O construtor deve ter qualquer visibilidade que não seja apenas na própria classe.
Método: implementar um método com visibilidade irrestrita *calcularCustoOperacao*, que recebe como parâmetros um array de números decimais representando os valores do preço do KWh dos últimos N meses, onde N é o tamanho do array, e o total de horas de funcionamento do motor. O método deve, com base nestas entradas e em alguns atributos, calcular e retornar o custo de operação, usando a seguinte lógica: Se o array com os valores do preço do KWh for nulo ou não contiver elementos, ou se o total de horas de funcionamento for menor do que 0, retornar -1. Caso contrário, calcular a média dos valores constantes no array. O custo total será média calculada x potência x total de horas de funcionamento, se voltagem for menor do que 380; e será média calculada x potência x total de horas de funcionamento x 1.12 se voltagem for maior ou igual a 380.

QUADRO 1

```
double d0;
byte b0;
float f1 = 198.11F;
double d1 = f1;
int a1 = (byte)f1; (1)
double d2 = 5452.0;
int a2 = (long)d2; (2)
short s1 = (short)d2;
b0 = -30;
b0 = (byte)d2; (3)
long l2 = (int)f1; (4)
d0 = false; (5)
float f2 = (float)d2;
```

QUADRO 2

```
package br.com.cesarschool.poo.ativos;
public class Ativo {
        public static final short NORMAL = 1;
        public static final short ESPECIAL = 2;
        private short tipo;
        float cotacao;
        protected void corrigirCotacao(float valor) {
                cotacao = cotacao + valor;
        }
        void atualizarTipoNormal(short tipo) {
                NORMAL = tipo; (1)
        }
}
```

QUADRO 2 (cont.)

```
package br.com.cesarschool.poo.ativos;
public class ProgramaAtivo {
	public static void main(String[] args) {
		Ativo at = new Ativo();
		at.cotacao = 100.22F;
		at.tipo = Ativo.ESPECIAL; (2)
		at.corrigirCotacao(20.0F);
	}
}
```

QUADRO 2 (cont.)

```
package br.com.cesarschool.poo.ativos.acoes;
import java.util.Date;
import br.com.cesarschool.poo.ativos.Ativo;
public class Acao extends Ativo {
	Date dataCotacao;
	Acao(short tipo, float cotacao) {
		this.tipo = tipo; (3)
		this.cotacao = cotacao; (4)
	}
	void atualizarPreco(float preco) {
		corrigirCotacao(preco);
	}
	static void atualizarDataCotacao() {
		dataCotacao = new Date(); (5)
	}
}
```

QUADRO 3

```
class Bateria {
	double carga;
	void carregar(double valor) {
		carga = carga + valor;
	}
	void descarregar(double valor) {
		carga = carga - valor;
	}
}
```

QUADRO 3 (cont.)

```
class Celular {
	String marca;
	Bateria bateria;
	Celular(String marca) {
		this.marca = marca;
	}
}
```

QUADRO 4

```
public class ProgramaCelular {
	public static void main(String[] args) {
		Bateria b1 = new Bateria();
		b1.carregar(100.0);
		Bateria b2 = b1;
		b2.descarregar(40.0);
		Celular c1 = new Celular("SONY");
		Celular c2 = new Celular("MOTO");
		c1.bateria = b2;
		c2.bateria = new Bateria();
		Celular c3 = c2;
		Celular c4 = c1;
		System.out.println(c3.marca);
		System.out.println(b1.carga);
		Bateria br = alterarBateria(c2, 30.0);
		br.descarregar(10.0);
		br = alterarBateria(c1, 80.0);
		c4.marca = "APPLE";
		System.out.println(c3.bateria.carga);
		br.descarregar(20.0);
		System.out.println(c4.bateria.carga);
		System.out.println(c1.marca);
	}
	static Bateria alterarBateria(Celular c, double carga) {
		Bateria b = new Bateria();
		b.carregar(carga);
		c.bateria = b;
		return b;
	}
}
```

FOLHA DE RESPOSTAS 1/3

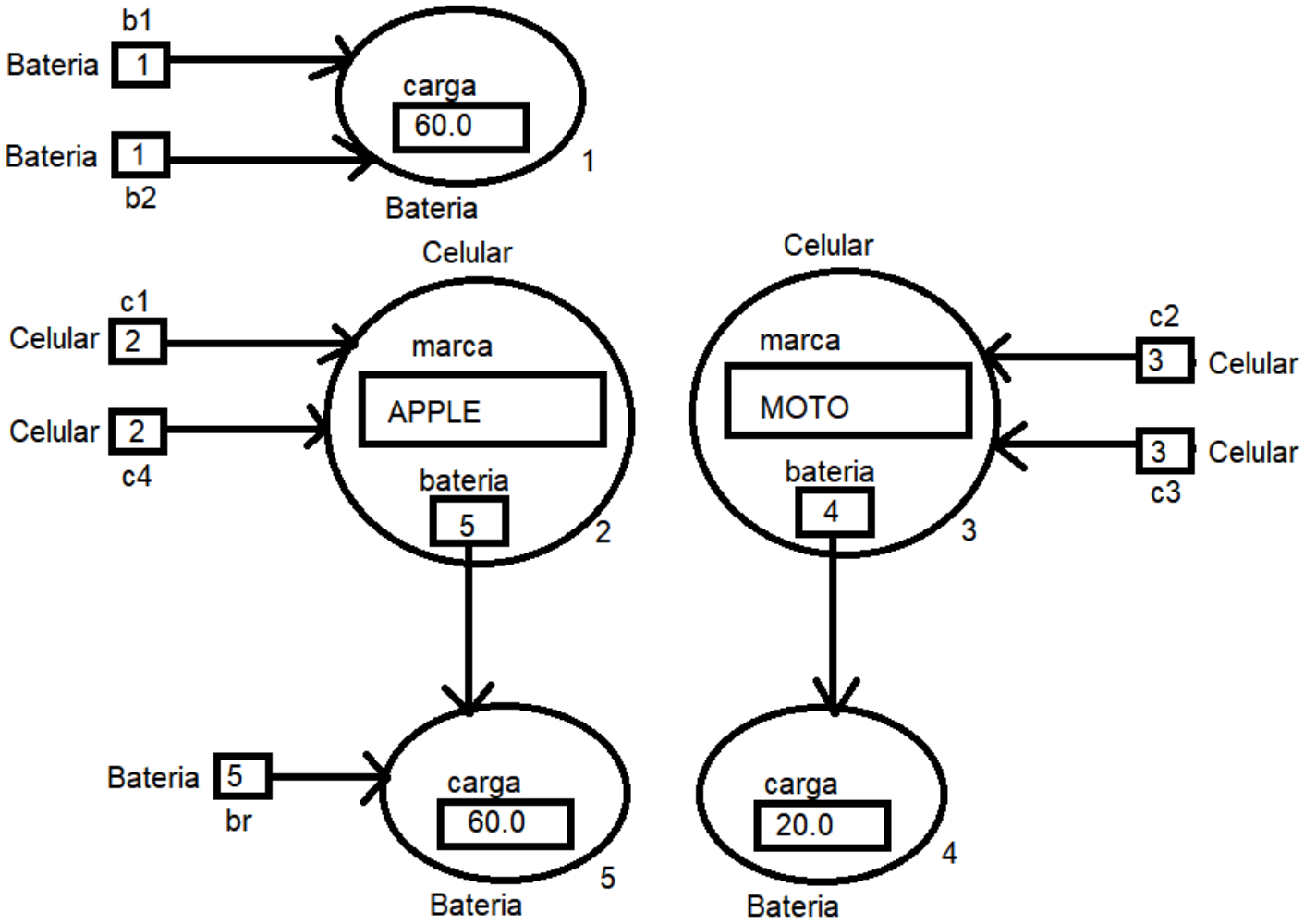
QUESTÃO 1		
Ref. cruzada	**Erro (não compila, ou perda)**	**Que perda(s), se for o caso**
1	HÁ PERDA	PARTES INTEIRA E DECIMAL
2	NÃO COMPILA	
3	HÁ PERDA	PARTE INTEIRA
4	HÁ PERDA	PARTE DECIMAL
5	NÃO COMPILA	

QUESTÃO 2	
Ref. cruzada	**Motivo do erro de compilação**
1	Porque o atributo NORMAL é final, e não pode ser alterado.
2	Porque o atributo tipo é private, e não pode ser acessado fora da classe onde ele foi definido.
3	Porque o atributo tipo é private, e não pode ser acessado fora da classe onde ele foi definido.
4	Porque o atributo cotacao é default e não pode ser acessado fora do pacote onde ele foi definido.
5	Porque o atributo dataCotacao tem escopo de objeto e o método atualizarDataCotacao é static, tem escopo de classe e não tem escopo de objeto.

FOLHA DE RESPOSTAS 2/3

QUESTÃO 3

Desenhe aqui o mapa de memória (com lápis ou caneta)



Escreva aqui as saídas no console na sequência dos System.out.println do código (com caneta)

```
MOTO
60.0
20.0
60.0
APPLE
```

FOLHA DE RESPOSTAS 3/3

QUESTÃO 4

Escreva NESTA PÁGINA a classe Motor, e no VERSO DESTA PÁGINA a classe MotorEletrico.

```
public class Motor {
	private String modelo;
	private double potencia;
	private int tempoVida;
	public Motor(String modelo, double potencia, int tempoVida) {
		this(potencia, tempoVida);
		this.modelo = modelo;
	}
	public Motor(double potencia, int tempoVida) {
		this.potencia = potencia;
		this.tempoVida = tempoVida;
	}
	String getModelo() {
		return modelo;
	}
	protected double getPotencia() {
		return potencia;
	}
	protected void setPotencia(double potencia) {
		this.potencia = potencia;
	}
	public int getTempoVida() {
		return tempoVida;
	}
	public void setTempoVida(int tempoVida) {
		this.tempoVida = tempoVida;
	}
}
```

QUESTÃO 4

Escreva NESTA PÁGINA a classe Motor, e no VERSO DESTA PÁGINA a classe MotorEletrico.

```
public class MotorEletrico extends Motor {
	private double voltagem;
	public MotorEletrico(String modelo, double potencia, int tempoVida,
			double voltagem) {
		super(modelo, potencia, tempoVida);
		this.voltagem = voltagem;
	}
	protected double getVoltagem() {
		return voltagem;
	}
	public double calcularCustoOperacao(double[] precosKwh,
		 int totalHorasFunc) {
		if (precosKwh == null || precosKwh.length == 0
				|| totalHorasFunc < 0) {
			return -1;
		}
		double acc = 0;
		for (int i=0; i<precosKwh.length; i++) {
			acc = acc + precosKwh[i];
		}
		// alternativamente, pode-se usar a versão simplificada
		// do for para iteração com arrays
		//for (double precoKwh : precosKwh) {
		//    acc = acc + precoKwh;
		//}
		double media = acc / precosKwh.length;
		double fator = 1;
		if (voltagem >= 380) {
			fator = 1.12;
		}
		return media * getPotencia() * totalHorasFunc * fator;
	}
}
```