CESAR SCHOOL
CIÊNCIAS DA COMPUTAÇÃO 3P - POO 2ª PROVA

NOME:_____

INSTRUÇÕES SOBRE RESPOSTAS:

- **QUESTÕES 1 e 2 – Usar caneta.** Indicar a linha por referência cruzada. Colocar um sequencial crescente em cada linha com erro no código e, na resposta, colocar o sequencial, em seguida o problema encontrado e finalmente a descrição da causa do problema, conforme trabalhado em sala de aula.
- **QUESTÃO 3 – Usar caneta.** Indicar o que é impresso no console EM ORDEM dos Sysouts executados.
- **QUESTÃO 4** – **Usar caneta.** Novo código: escrever o código todo. Código alterado: marcar a linha a ser alterada no QUADRO com uma referência cruzada e REESCREVER TODA A LINHA alterada. Código excluído e código alterado em uma mesma classe: reescrever a classe toda.
- **QUESTÃO 5 – Usar lápis ou caneta.**

ATIVIDADE CONTINUADA (4,0 PONTOS): Entrega das fases 4 e 5 (opcional) até 28/11/2022.

QUESTÃO 1 (1,5 PONTOS): Dado o código abaixo, quais linhas possuem erros de compilação ou de execução? Descreva as causas dos erros apontados.

```
public class Alimento {}
public class Lactinio extends Alimento {}
public class Fruta extends Alimento {}
public class Leite extends Lactinio {}
public class Banana extends Fruta {}
public class Uva extends Fruta {}
public class CasosConversao {
	public void processarConversoes1() {
		Fruta m1 = new Uva();
		Lactinio c = new Leite();
		Alimento ma = new Alimento();
		c = ma;
		Leite n1 = m1;
		Banana mc = new Banana();
		m1 = mc;
	}
	public void processarConversoes2() {
		Alimento ma = new Banana();
		Lactinio c = new Leite();
		Fruta m1 = new Uva();
		Fruta f = (Fruta)c;
		Fruta mt = (Banana)ma;
		Banana mc = (Fruta)ma;
		c = (Leite)ma;
	}
}
```

QUESTÃO 2 (0,5 PONTO): Dado o código abaixo, quais linhas possuem erros de compilação? Descreva as causas dos erros apontados.

```
public abstract class Alimento {
	public abstract void origem();
	public abstract void calorias();
}
public abstract class Lactinio extends Alimento {}
```

```
public abstract class Fruta extends Alimento {
	public void calorias() {
		System.out.println("NAO DISPONIVEL");
	}
}
public class Banana extends Fruta {
	public final void origem() {
		System.out.println("AREA EQUATORIAL");
	}
}
public final class Uva extends Fruta {
	public void origem() {
		System.out.println("AREA NAO EQUATORIAL");
	}
	public void calorias() { }
}
public class Leite extends Lactinio {
	public void origem() {
		System.out.println("MAMIFEROS");
	}
}
public abstract final class Verdura extends Alimento {
	public void temSemente();
}
public class BananaNanica extends Banana {
	public void origem() {
		System.out.println("BRASIL");
	}
}
public class UvaPassa extends Uva {}
```

QUESTÃO 3 (1,5 PONTOS): Dado o código abaixo, indique as saídas no console quando executamos o programa em questão.

```
public class Alimento {
	public void origem() {
		System.out.println("UNIVERSO");
	}
	public void calorias() {
		System.out.println("NAO DISPONIVEL");
	}
}
public class Fruta extends Alimento {
	public void origem() {
		super.origem();
		System.out.println("TERRA");
	}
}
public class Lactinio extends Alimento {}
public class Banana extends Fruta {
	public void origem() {
		super.origem();
		System.out.println("AREA EQUATORIAL");
	}
}
public class Uva extends Fruta {}
public class Leite extends Lactinio {
	public void calorias() {
		super.calorias();
		System.out.println("200");
	}
}
```

```java
public class ProgramaAlimento {
	public static void main(String[] args) {
		Alimento m1 = new Fruta();
		m1.origem();
		m1.calorias();
		Alimento m2 = new Leite();
		m2.origem();
		m2.calorias();
		Alimento m3 = new Lactinio();
		m3.origem();
		m3.calorias();
		Banana m4 = new Banana();
		m4.origem();
		m4.calorias();
		Fruta m5 = new Uva();
		m5.origem();
		m5.calorias();
	}
}
```

QUESTÃO 4 (1,0 PONTO): O código abaixo apresenta duas hierarquias já implementadas e um código duplicado. Usando os recursos de polimorfismo avançado, evolua a solução em questão, SEM QUEBRAR AS HIERARQUIAS atuais, de forma que o código duplicado seja único, servindo às duas classes que originalmente lá aparecem.

```java
public class Ativo {
	double valor;
}
public class Moeda extends Ativo {}
public class MoedaTradicional extends Moeda {}
public class Acao extends Ativo {}
public class AcaoPreferencial extends Acao {}
public class Criptomoeda extends Moeda {
	boolean semVencimento;
	double adicional;
	public boolean isSemVencimento() {
		return semVencimento;
	}
	public double retornarImposto() {
		if (!isSemVencimento()) {
			return valor * 0.05;
		} else {
			return (adicional + valor) * 0.035;
		}
	}
}
public class AcaoOrdinaria extends Acao {
	double aliquotaBase;
	public double retornarImposto() {
		return (1 + aliquotaBase/100) * valor;
	}
}
public class AvaliadorExpectativa {
	public boolean excedeExpectativa(AcaoOrdinaria acao, int prazo, double taxa) {
		double imposto = acao.retornarImposto();
		if (prazo < 180) {
			return imposto < taxa;
		} else {
			return imposto < (taxa * 0.9);
		}
	}
```

```
	public boolean excedeExpectativa(Criptomoeda cripMoeda, int prazo, double taxa) {
		double imposto = cripMoeda.retornarImposto();
		if (prazo < 180) {
			return imposto < taxa;
		} else {
			return imposto < (taxa * 0.9);
		}
	}
}
```

QUESTÃO 5 (1,5 PONTOS): Escreva uma classe em JAVA **`CalculadoraDistanciaCoordenadas`** cujo objetivo é calcular a distância entre duas coordenadas cartesianas, representados por tuplas (x, y – no plano) ou (x, y, z – no espaço). Premissas:

- A classe **`CalculadoraDistanciaCoordenadas`** deve ter dois métodos sobrecarregados para calcular as e retorna as distâncias entre duas coordenadas, sendo um método que calcule a distância entre duas coordenadas NO PLANO, e outro método que calcule a distância entre duas coordenadas NO ESPAÇO.
- Os parâmetros de entrada dos métodos que representam coordenadas no plano e no espaço devem ser dos tipos dados no quadro abaixo.
- Deve haver reuso de um método por outro.

```
public class CoordenadaPlano {
	private double x;
	private double y;
	public CoordenadaPlano(double x, double y) {
		this.x = x;
		this.y = y;
	}
	public double getX() {
		return x;
	}
	public double getY() {
		return y;
	}
}
public class CoordenadaEspaco extends CoordenadaPlano {
	private double z;
	public CoordenadaEspaco(double x, double y, double z) {
		super(x, y);
		this.z = z;
	}
	public double getZ() {
		return z;
	}
}
```

No PLANO – Dadas duas coordenadas A(x_A, y_A) e B(x_B, y_B), a distância entre elas é:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

No ESPAÇO – Dadas duas coordenadas A(x_A, y_A, z_A) e B(x_B, y_B, z_B), a distância entre elas é:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$$