Universidade Luterana do Brasil **ULBRA – Campus Torres** **Pró-Reitoria de Graduação**		Tipo de atividade: Prova ( ) Trabalho ( ) ..... ( ) Avaliação: G1 ( ) G2 ( ) Substituição de Grau: G1 ( ) G2 ( )
Curso:	Disciplina:	Data:
Turma:	Professor(a):	Valor da Avaliação:
Acadêmico(a): n°:		Nota:

A avaliação parcial 2 será composta de três exercícios, que deverão ser realizados da seguinte forma:
1 – Primeiro exercício será feito com Dojô, aulos remotos contribuem no chat do meet.
2 – Escolher um exercícios entre os exercícios restantes para gravar um vídeo explicando a técnica utilizada.
3 – Todos os exercícios e vídeos devem ser enviados para uma pasta no drive e o link deve ser enviado no tópico da AP2
Exercício 01

Crie duas classes: `Ponto2D` e `Ponto3D`. `Ponto2D` possui como atributos as coordenadas `x` e `y`, enquanto `Ponto3D`, além delas, também possui a coordenada `z`. Utilize a relação de herança para representar estas classes.

A respeito dos construtores, `Ponto2D` deve ter apenas um construtor, que recebe os valores de `x` e `y` como parâmetros (tipo `double`). Já `Ponto3D` também deve ter apenas um construtor, que deve receber `x`, `y` e `z` como parâmetros (também do tipo `double`).

Dica: Se a relação de herança e a declaração dos construtores foram feitas corretamente, você deverá, obrigatoriamente, chamar o construtor da superclasse explicitamente.

Ambas as classes devem sobrescrever o método `toString()`, que é originalmente declarado na classe `Object`. Este método deve retornar uma representação do objeto em forma de `String`, indicando qual o valor de cada coordenada. É importante que `Ponto3D` tire proveito do método `toString()` de `Ponto2D` para mostrar os valores das coordenadas `x` e `y`.

Exercício 02

Crie uma classe `Veiculo` com um atributo `ligado` (privado), que indica se o carro está ligado ou não. Esta classe deve ter também os métodos `ligar()` e `desligar()`, que definem o valor para este atributo, e um método *getter* (`isLigado()`).

Depois crie três subclasses de `Veiculo`: `Automovel`, `Motocicleta` e `Onibus`. Cada classe destas deve sobrescrever os métodos `ligar()` e `desligar()` e deve imprimir mensagens como "*Automóvel ligado*", "*Motocicleta desligada*", etc. Para manter a consistência do modelo, descubra como fazer para que o atributo ligado de `Veiculo` tenha o valor correto quando os métodos são chamados.

Crie uma aplicação que instancia três veículos, um de cada tipo, e chama os métodos `ligar()`, `desligar()` e `isLigado()`. O resultado obtido deve ser consistente com o que o modelo representa. Por exemplo, ao chamar o método `ligar()` de um `Automovel`, é esperado que o método `isLigado()` retorne `true`.

Exercício 03

Crie uma interface `AreaCalculavel` com um método `calcularArea()` e crie classes de figuras geométricas que implementam este método (como quadrado, circunferência e retângulo). Depois crie uma classe com um método `main()` para exercitar as chamadas aos métodos que calculam a área.