**Lista de Exercícios**

**Programação II**

Construa um aplicativo flutter para entrada e apresentação de dados para atender cada um dos programas abaixo. Utilize programação orientada a a objetos.

1. Crie uma classe para cada figura geométrica abaixo, cada classe deve calcular a área e o perímetro da figura geométrica:
	1. Retangulo
		1. Atributos:
			1. base, altura
		2. Métodos:
			1. calcularPerimetro(): double;
			2. calcularArea(): double
			3. calcularDiagonal(): double
			4. main():void
	2. Triangulo
		1. Atributos:
			1. Base,altura
		2. Métodos:
			1. calcularPerimetro(): double;
			2. calcularArea(): double
			3. éTriangulo(): boolean
			4. éEquilatero(): boolean
			5. éEscaleno(): boolean
			6. éEquilatero(): boolean
	3. Construa também as classes abaixo com a mesma ideologia.
		1. Trapézio
		2. Pirâmide
		3. Cone
		4. Cilindro
		5. Cubo
		6. Esfera
2. Crie uma classe que seja capaz de calcular o IMC.
	1. Atributos:
		1. Nome
		2. Massa
		3. Altura
	2. Métodos:
		1. calcularIMC():float
3. Construa a classe com três atributos de notas de um aluno, calcule a média obtida por este aluno e no final escreva o resultado indicando se o mesmo foi **aprovado** ou **reprovado** (considere que aluno aprovado obteve Média >= 7,0 e aluno reprovado Média < 7,0).
4. Elabore uma classe para calcular a equação do segundo grau e imprima as raízes da equação na tela, os valores a, b e c são inseridos pelo usuário: ax2 + bx + c = 0, Δ= b2 - 4ac ,.
5. Construa a classe Ponto2d.
	1. Atributos
		1. x,y
	2. métodos
		1. mostrarPonto(): retorna uma string no formato “(x,y)”
		2. calcularDistancia(): calcula a distância do ponto até (0,0)
		3. calcularDistanciaDe(Ponto2d p2): recebe como parâmetro uma instancia de Ponto2d e calcula a distância do ponto recebido ao (x,y) do próprio objeto.
		4. calcularDistanciaTotal(Ponto2d [] pontos): recebe um vetor de pontos. Calcule a distância do primeiro ponto do vetor ao ponto (0,0), some essa distância à distância do segundo ponto e assim sucessivamente.
6. Crie o diagrama de classes e uma classe Elevador para armazenar as informações de um elevador de prédio. A classe deve armazenar o andar atual (térreo = 0), total de andares no prédio (desconsiderando o térreo), capacidade do elevador e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:
	1. **inicializar**: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazio);
	2. **entrar**: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
	3. **sair**: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
	4. **subir**: para subir um andar (não deve subir se já estiver no último andar);
	5. **descer**: para descer um andar (não deve descer se já estiver no térreo);
	6. **estáVazio**: retorna verdadeiro ou falso.
	7. **pessoasPresentes**: retorna a quantidade de pessoas presente no elevador.