**REGRAS**:

* Todos os exercícios devem funcionar com qualquer valor de entrada. Faça testes.
* Todas as variáveis, exceção das constantes, devem ter o valor solicitado ao usuário (input).

**EXERCÍCIOS**

1. Classe Triângulo. Crie uma classe Triangulo com atributos base e altura. Adicione o método area() para calcular a área do triângulo.
2. Classe Carro. Crie uma classe Carro com atributos modelo, ano e quilometragem. Adicione o método detalhes() que imprime todas as informações do carro.
3. Classe Animal. Crie uma classe Animal com atributos especie e som. Adicione o método emitir\_som() que imprime o som do animal.
4. Classe Aluno. Crie uma classe Aluno com atributos nome e notas (lista de notas). Adicione o método media() para calcular e retornar a média das notas do aluno.
5. Classe TransacaoBancaria. Crie uma classe TransacaoBancaria com atributos saldo e historico\_transacoes (uma lista vazia). Adicione métodos depositar(valor), sacar(valor) e extrato(). No método depositar, adicione o valor ao saldo e registre a transação em historico\_transacoes. No método sacar, diminua o valor e registre a transação, garantindo que não permite saldo negativo. O método extrato() deve imprimir todas as transações e o saldo atual.
6. Classe GerenciadorFinanceiro.Crie uma classe GerenciadorFinanceiro com os seguintes atributos e métodos:

**Atributos**:

* contas: um dicionário onde as chaves são IDs de conta e os valores são instâncias de uma classe Conta (que você deve criar como uma classe interna ou separada). Cada Conta deve possuir os atributos saldo, historico\_transacoes, e tipo\_conta (como 'Poupança', 'Corrente').

**Métodos**:

* criar\_conta(id\_conta, tipo\_conta): cria uma nova instância de Conta e a adiciona ao dicionário contas.
* depositar(id\_conta, valor): realiza um depósito na conta especificada e registra a transação no histórico.
* sacar(id\_conta, valor): realiza um saque, garantindo que o saldo não fique negativo. Registra a transação.
* transferir(id\_origem, id\_destino, valor): realiza uma transferência entre duas contas diferentes, garantindo que a conta de origem tenha saldo suficiente. Registra a transação em ambas as contas.
* consultar\_extrato(id\_conta): imprime o histórico de transações e o saldo da conta especificada.
* calcular\_juros(id\_conta): adiciona juros ao saldo se o tipo\_conta for 'Poupança'. Esse método simula juros simples e aumenta o saldo em uma porcentagem fixa.
* Esse exercício envolve vários conceitos avançados, como manipulação de dicionários, gerenciamento de múltiplas instâncias de classe, controle de operações entre contas e condicionais específicas por tipo de conta.