1. Classe Ponto 2D:

Crie uma classe Ponto2D que represente um ponto no plano cartesiano (x, y). Implemente os seguintes métodos mágicos:

- __init__: Inicializa o ponto com coordenadas x e y.
- __str__: Retorna uma string representando o ponto no formato "(x, y)".
- __eq__: Compara dois pontos para verificar se são iguais (mesmas coordenadas).
- __add__: Define a adição de dois pontos, resultando em um novo ponto com a soma das coordenadas.

2. Classe Fração:

Crie uma classe Fracao que represente uma fração matemática (numerador/denominador). Implemente os seguintes métodos mágicos:

- __init__: Inicializa a fração com numerador e denominador (valores inteiros).
- __str__: Retorna uma string representando a fração no formato "n/d".
- __eq__: Compara duas frações para verificar se são iguais (mesmos valores após simplificação).
- __add__: Define a adição de duas frações, resultando em uma nova fração com a soma dos numeradores e um denominador comum (mínimo múltiplo comum).
- __mul__: Define a multiplicação de duas frações, resultando em uma nova fração com o produto dos numeradores e o produto dos denominadores.

3. Classe Círculo:

Crie uma classe Circulo que represente um círculo com centro e raio. Implemente os seguintes métodos mágicos:

- __init__: Inicializa o círculo com coordenadas do centro (x, y) e raio.
- __str__: Retorna uma string representando o círculo no formato "Círculo: centro (x, y), raio r".
- __eq__: Compara dois círculos para verificar se são iguais (mesmos centros e raios).
- __add__: Define a adição de dois círculos, resultando em um novo círculo com o raio da soma dos raios originais (se os centros coincidem, soma os raios; caso contrário, retorna None).
- __area__: Define o método mágico __area__ para calcular a área do círculo

4. Classe Conta Bancária:

Crie uma classe ContaBancaria que represente uma conta bancária com saldo e titular. Implemente os seguintes métodos mágicos:

- __init__: Inicializa a conta com o nome do titular e saldo inicial (valor positivo).
- __str__: Retorna uma string representando a conta no formato "Conta de: titular, Saldo: ".
- __eq__: Compara duas contas para verificar se são iguais (mesmos titulares e saldos).
- __add__: Define a adição de duas contas, resultando em uma nova conta com o titular do primeiro e o saldo total das duas contas.
- __sub__: Define a subtração de duas contas, resultando em uma nova conta com o titular do primeiro e a diferença entre os saldos (se o saldo for insuficiente, retorna None).
- __depositar__: Define o método mágico __depositar__ para depositar um valor na conta (valor positivo).
- __sacar__: Define o método mágico __sacar__ para sacar um valor da conta (valor positivo, verificando se o saldo é suficiente).

5. Classe Data:

Crie uma classe Data que represente uma data composta por dia, mês e ano. Implemente os seguintes métodos mágicos:

- __init__: Inicializa a data com valores de dia, mês e ano (válidos).
- __str__: Retorna uma string representando a data no formato "dd/mm/aaaa".
- __eq__: Compara duas datas para verificar se são iguais (mesmos dias, meses e anos).
- __lt__: Define a comparação de menor que entre duas datas, verificando a ordem cronológica (considerando ano, mês e dia).
- __gt__: Define a comparação de maior que entre duas datas, inversamente à comparação de menor que.
- __add__: Define a adição de dias a uma data, resultando em uma nova data com os dias adicionados