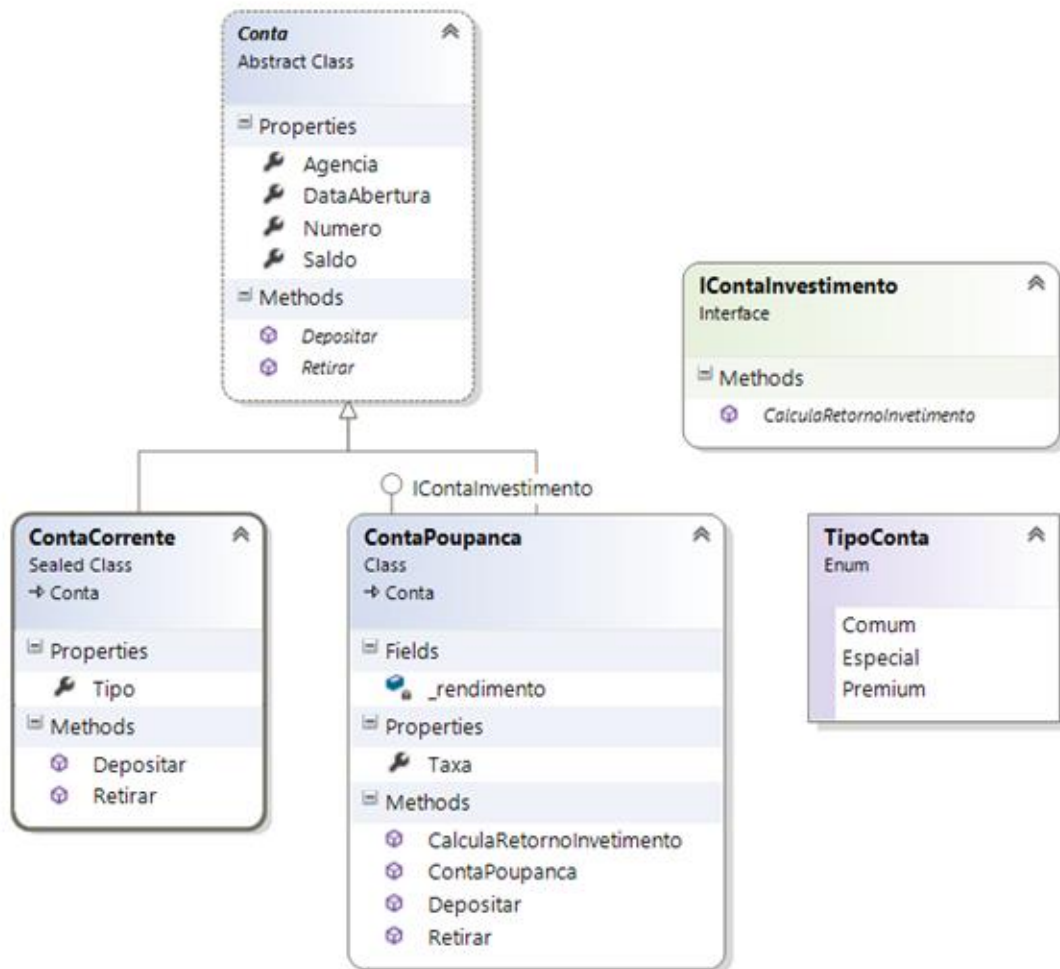


**FIAP – Faculdade de Informática e Administração Paulista**  
**Fábrica de Desenvolvimento**  
 Professor: Thiago T. I. Yamamoto

## Exercício 02 – C# e Orientação a Objetos

Construa as classes de acordo com o diagrama abaixo:



1. Interface **IContaInvestimento**:
  - a. Método: `CalculaRetornoInvestimento()`;
2. Enum **TipoConta**: Comum, Especial, Premium;
3. Classe Abstrata **Conta**:
  - a. Namespace: `Fiap.Banco.Model`
  - b. Propriedades:
    - i. **Agencia**: int
    - ii. **Numero**: int
    - iii. **DataAbertura**: DateTime
    - iv. **Saldo**: decimal

- c. Métodos abstratos:
  - i. **Depositar**(decimal valor)
  - ii. **Retirar**(decimal valor)
- 4. Classe **ContaCorrente**:
  - a. Namespace: Fiap.Banco.Model
  - b. Deve herdar a classe **Conta** e não pode ser herdada;
  - c. Propriedades:
    - i. **Tipo**: Enum TipoConta
  - d. Métodos:
    - i. **Depositar**: adiciona valor ao saldo;
    - ii. **Retirar**: Se a conta for comum e o saldo fica negativo depois do saque deve lançar uma exceção e não permitir a retirada.
- 5. Classe **ContaPoupanca**:
  - a. Namespace: Fiap.Banco.Model
  - b. Deve herdar a classe **Conta** e implementar a interface **IContaInvestimento**
  - c. Propriedades:
    - i. **Taxa**: decimal;
  - d. Fields readonly (somente leitura):
    - i. **\_rendimento**: decimal;
  - e. Construtor: deve receber um valor decimal e atribuir esse valor ao field **\_rendimento**;
  - f. Métodos:
    - i. **Depositar**: atribui valor ao saldo;
    - ii. **Retirar**: verifica se o saldo é suficiente, se for deve descontar também a Taxa de retirada, se não, lance uma exceção;
    - iii. **CalculaRetornoInvestimento**: retorna o valor do saldo vezes o rendimento;
- 6. No método Main, crie uma ContaCorrente e uma ContaPoupanca com valores para todas as propriedades; Teste os métodos de retirada para ver a exception.