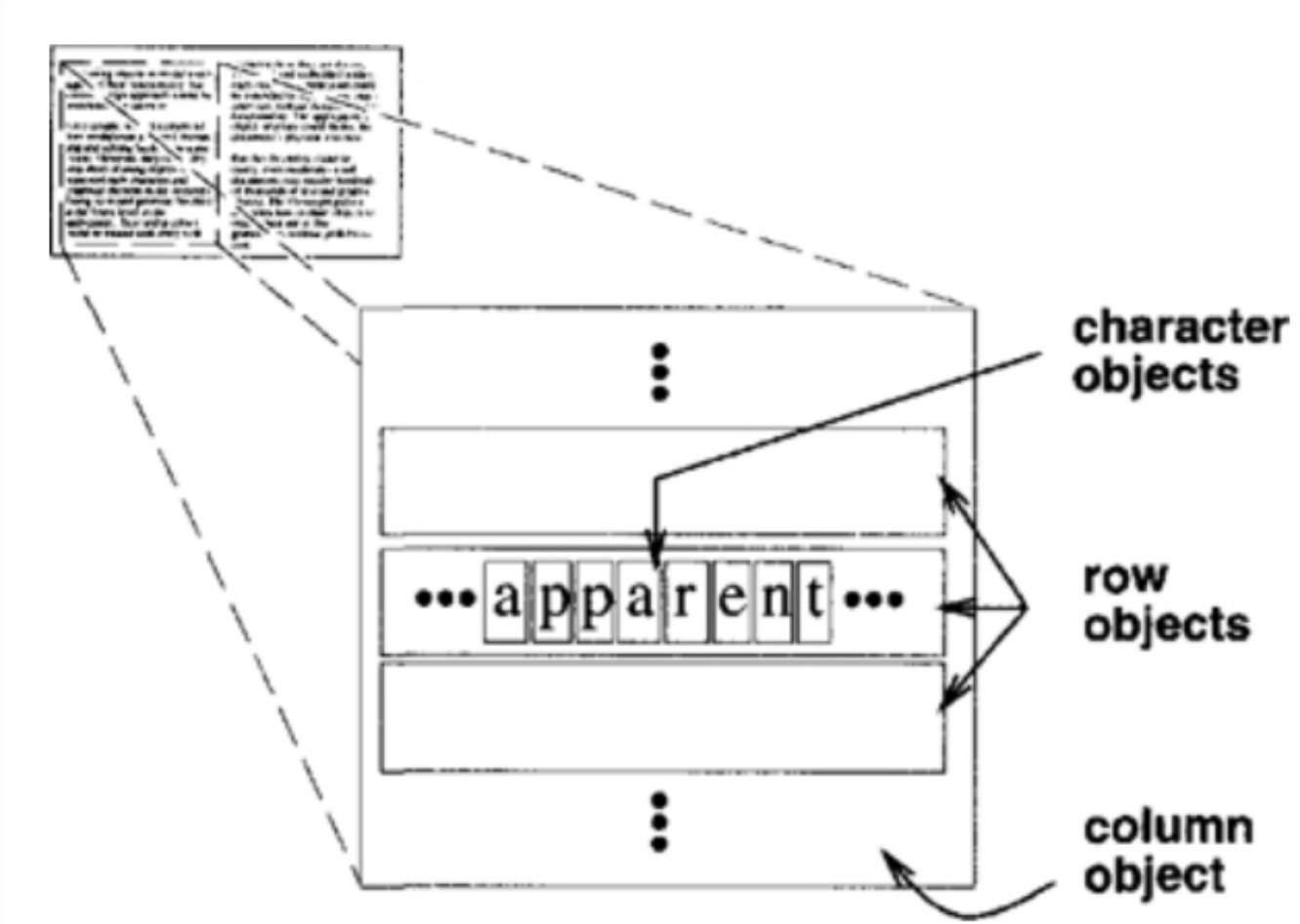
# Flyweight 💻

Análise e Desenvolvimento de Sistemas - 6º Período

# Cenário

- Imagine que estamos construindo um editor de texto usando orientação a objetos
- Cada elemento da janela do editor pode ser considerado um objeto
- Para **modularizar** ainda mais nossa aplicação, podemos imaginar que cada **caracter** é um objeto


character objects
row objects
column object
apparent

# Cenário

- O objeto **caracter** pode ter algumas propriedades, como: linha, coluna, fonte
- Qual o problema nesse cenário? **A quantidade de objetos criados!**
- Imagina seu artigo de TCC, onde cada caracter é um objeto instanciado!

# Cenários

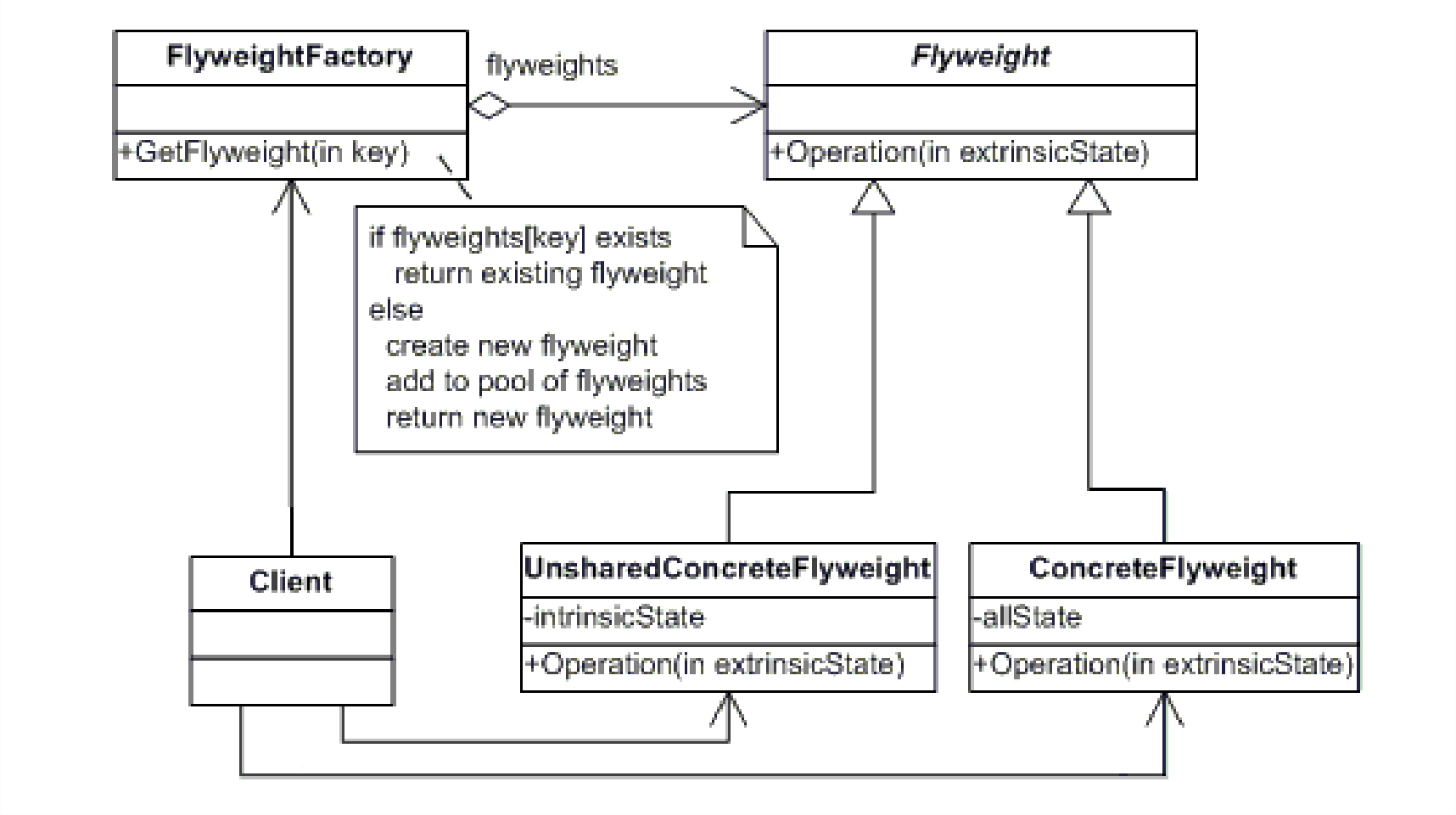
- Temos muitos objetos que compartilham das mesmas características sendo criados
- Isso pode ocasionar um **consumo grande de recursos como memória e um run-time overhead**.

# Flyweight

- Otimizar recursos e memória, **compartilhando** objetos que possuem partes em comum, enquanto mantém o estado exclusivo separado.
- Usado quando grande número de objetos semelhantes precisa ser criado
- Cada objeto compartilha parte de seu estado com os objetos semelhantes
- Esse compartilhamento permite economizar memória e reduzir sobrecarga associada a criação de objetos

# Conceitos Fundamentais

- Estado intrínseco: Estado interno ao contexto do flyweight. Compartilhado entre os objetos;
- Estado extrínseco: Estados que não são compartilhados entre os objetos
- Flyweight factory: Gerencia os objetos. Garante que os objetos que compartilham estados sejam reutilizados em vez de criados novamente.


FlyweightFactory
flyweights
Flyweight
+GetFlyweight(in key)
+Operation(in extrinsicState)
if flyweights[key] exists
return existing flyweight
else
create new flyweight
add to pool of flyweights
return new flyweight
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)

# Elementos do UML

- Flyweight: Interface para o ConcreteFlyweight e o UnsharedConcreteFlyweight;
- ConcreteFlyweight: Implementa característica compartilhadas;
- UnsharedConcreteFlyweight: Pode usar as características do Concrete e contar características que não são compartilhadas;
- FlyweightFactory: Gerencia a criação dos objetos, permitindo a reutilização de objetos já criados.

# Motivações

- Economia de memória: Reduz a quantidade de memória necessária para armazenar objetos, compartilhando partes comuns entre eles.
- Gerenciamento eficiente de recursos: É útil quando os recursos (como conexões de banco de dados) são limitados e precisam ser compartilhados entre várias partes do sistema.