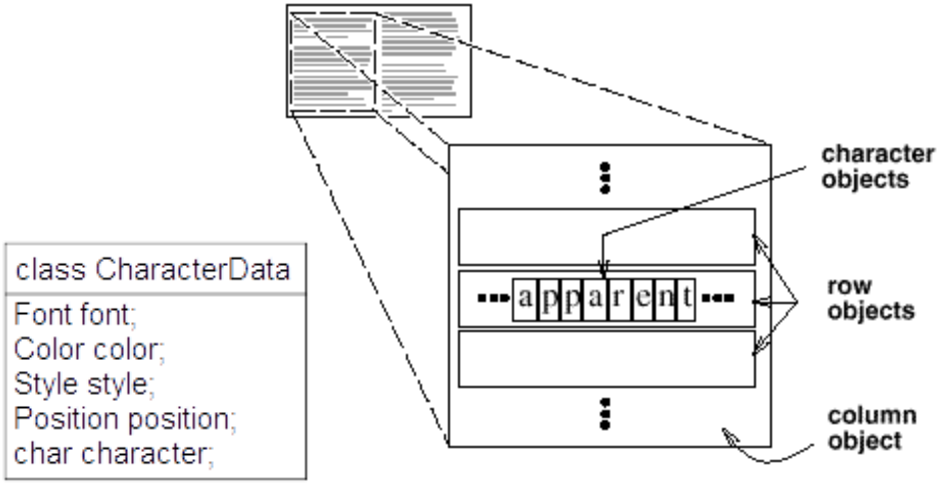
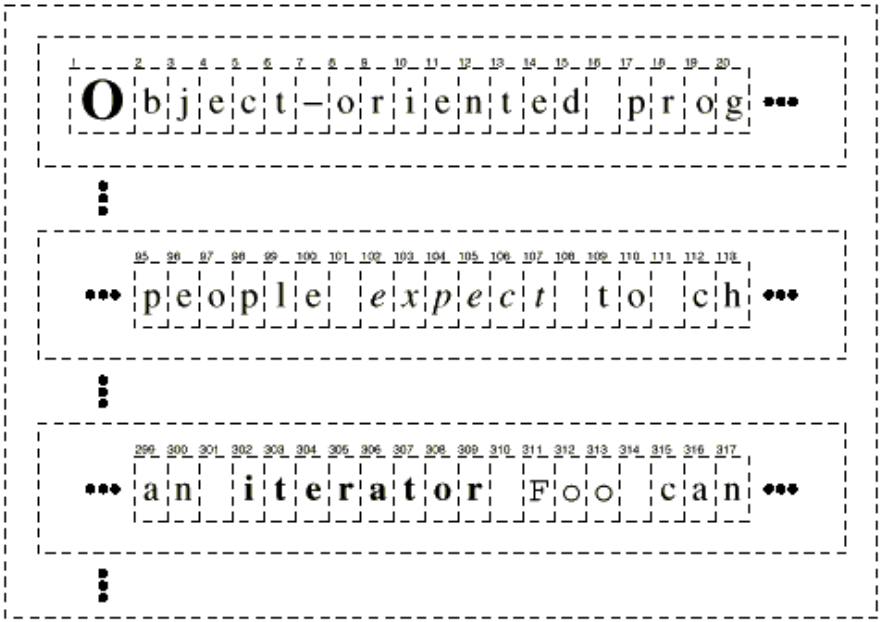
Flyweight

Naivní textový editor

- **Požadavky**
  - nové znakové sady
  - práce s bloky, formátování
  - simulace vzhledu dokumentu
- **Naivní řešení**
  - znak, řádek, stránka = **třída**
  - výskyt v dokumentu = **objekt**
  - objekty nesou svoji identitu
- **Problém**
  - **hodně objektů** (paměť)
  - objekty nesou **hodně dat** (paměť)
  - data se téměř **neliší**



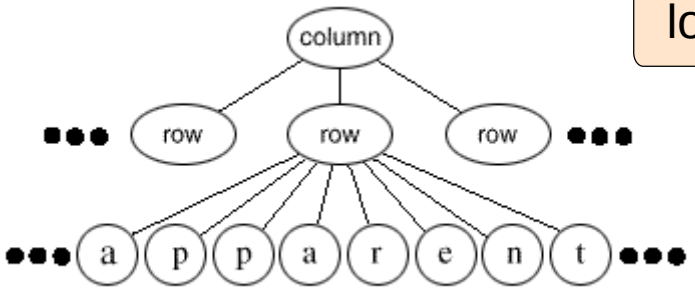


Objekt flyweight – sdílení jedné kopie

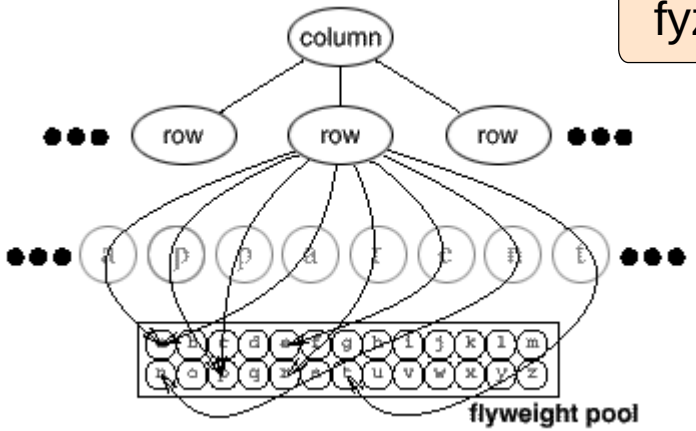
Flyweight je sdílený objekt, který se může současně vyskytovat ve více různých souvislostech.

- **Objekt flyweight**
  - identita bez kontextu
- **Vnitřní (intristic) stav**
  - data v objektu
  - nezávislý na kontextu
  - nebrání sdílení
- **Vnější (extrinsic) kontext**
  - flyweight o něm **neví**
  - ukládá a počítá ho klient
- **Volání**
  - flyweight zná svůj stav
  - volající klient předá kontext

logicky

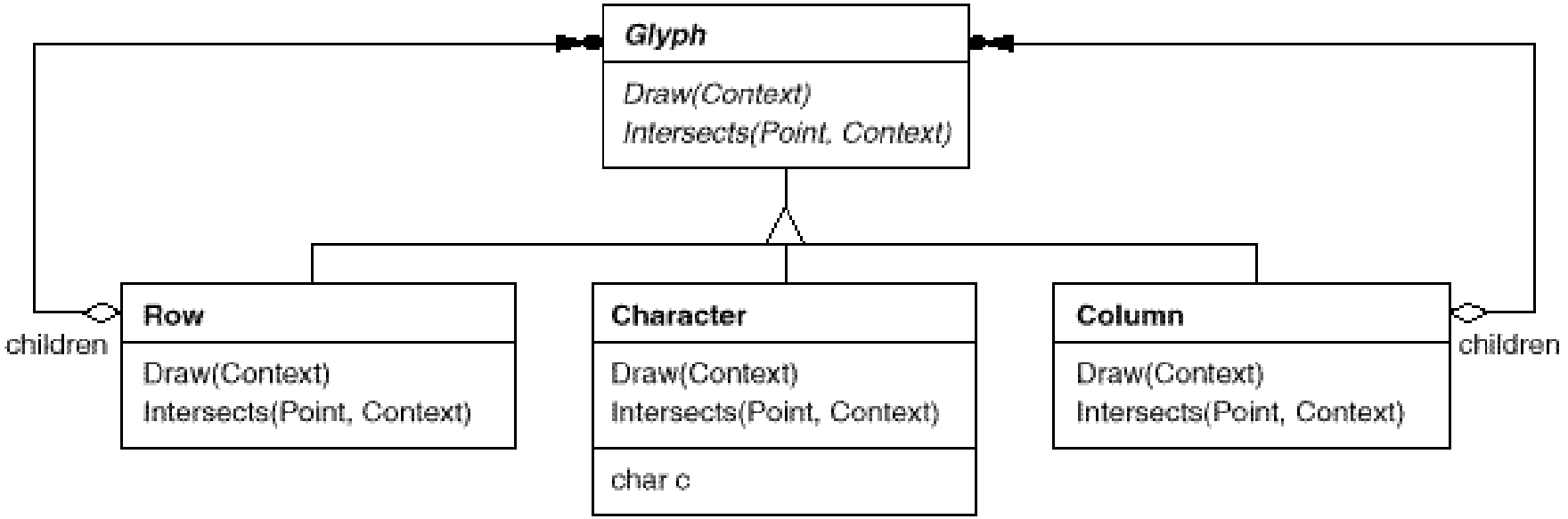


fyzicky

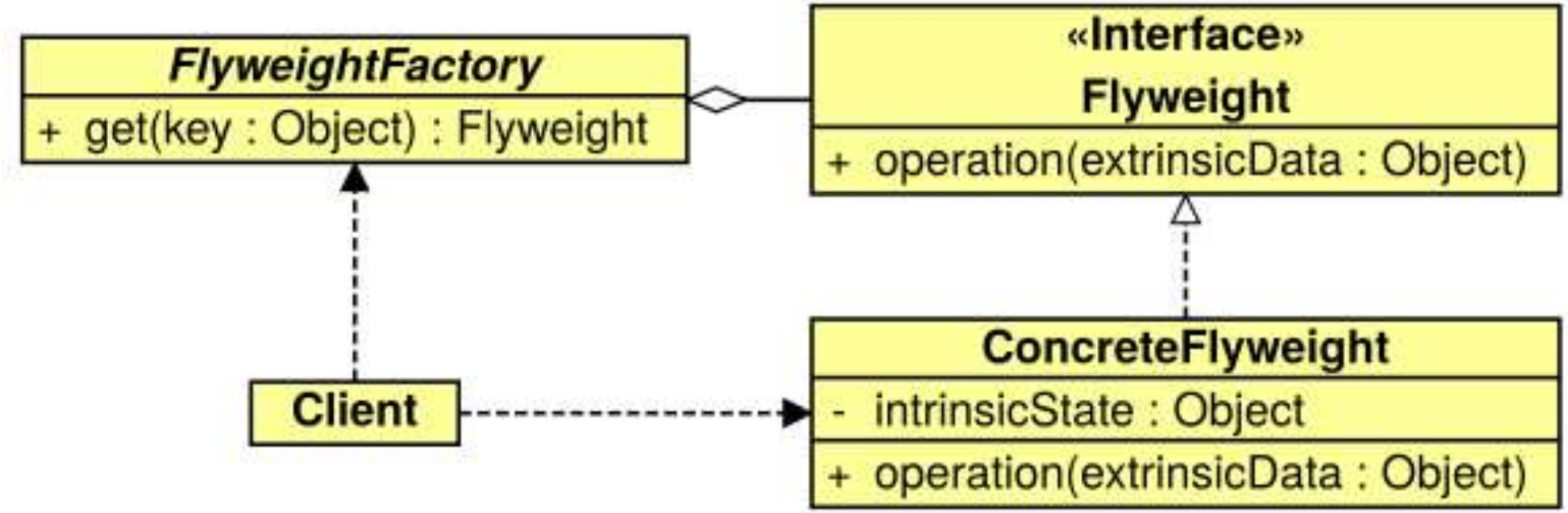


Textový editor pomocí vzoru Flyweight

- **Abstraktní třída Glyph na grafické objekty**
- **Potomek Character je flyweight**
- **Potomci Row a Column nejsou sdílené**

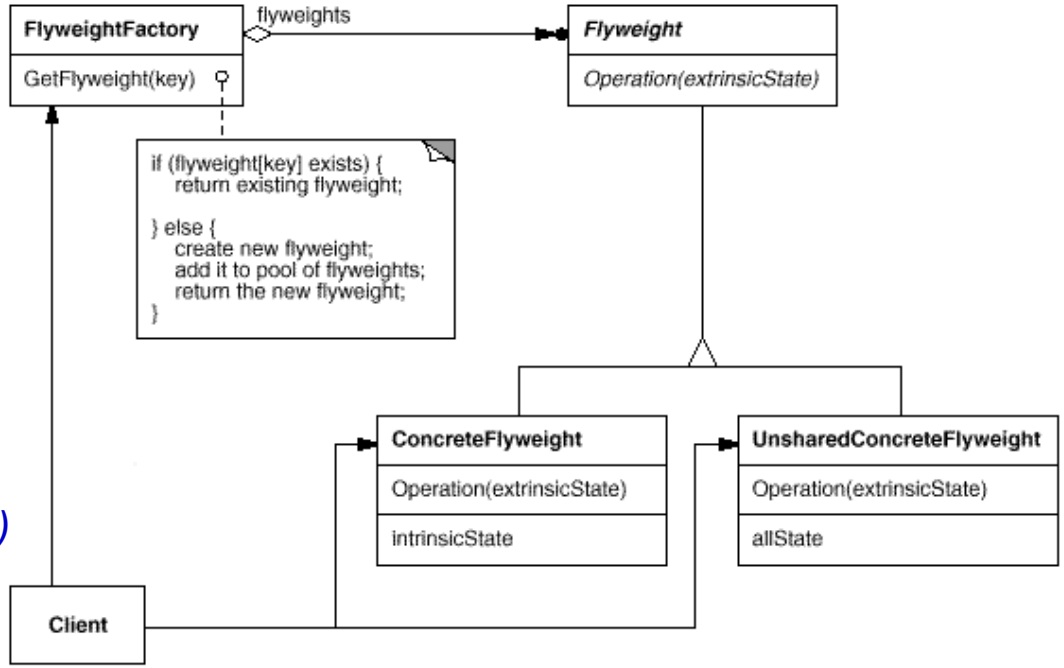


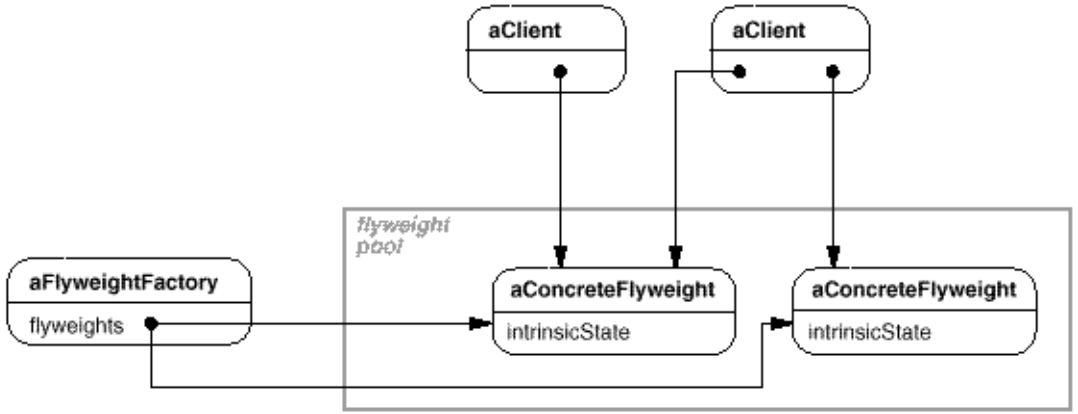
Součásti vzoru Flyweight



Součásti vzoru Flyweight

- **IFlyweight**
  - interface s metodami
- **Flyweight : IFlyweight** *(znak)*
  - přístup k instanci odkudkoliv
  - datové položky na vnitřní stav
  - data nezávisí na kontextu
- **UnsharedFlyweight: IFlyweight**
  (řádek, sloupec)
  - Flyweight bez sdílení
- **FlyweightFactory**
  - vytváří a spravuje flyweighty
  - zabezpečuje sdílení a přístup
- **Client**
  - používá reference na flyweighty
  - poskytuje kontext





Pořadí volání metod

- **FlyweightFactory** vytváří nové instance potomků **Flyweight**
- **FlyweightFactory** vrací referenci na dříve vytvořenou instanci
- **Client** nevytváří instance přímo, aby bylo zajištěné řádné sdílení
- **Client** musí oznámit továrně až objekty přestane používat
- Objekty **Flyweight** by měly být neměnné z pohledu **Clienta**
- **Client** spravuje externí stav a předává ho při volání metod

Předpoklady použití

Návrhový vzor Flyweight má smysl použít jenom pokud jsou splněny všechny z následujících podmínek.

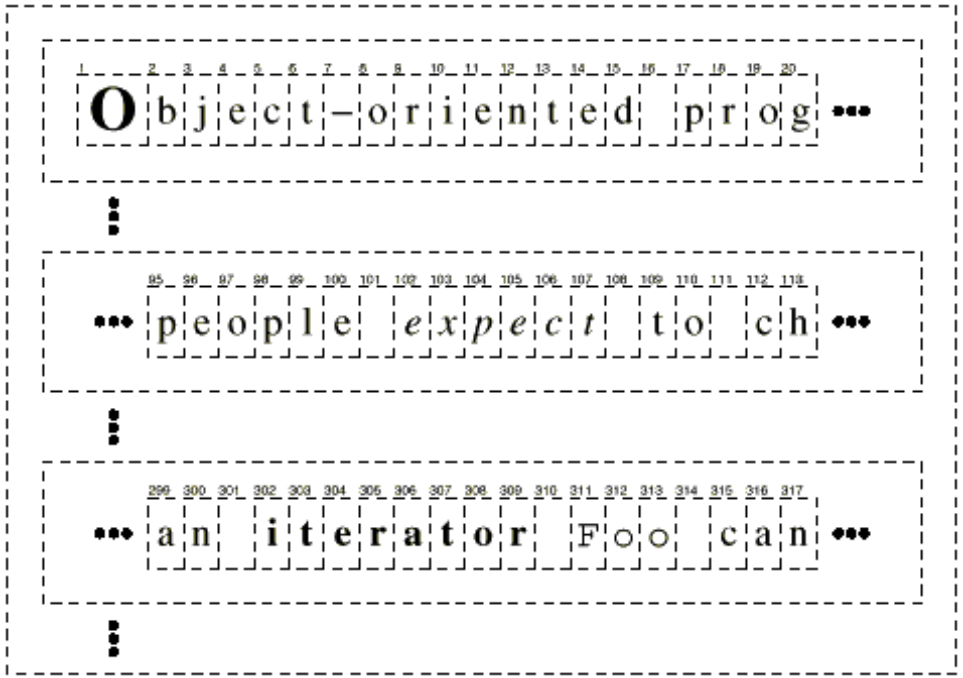
- **Velké množství objektů v aplikaci**
  - chceme prostor pro redukci počtu objektů
- **Objekty uchovávají hodně dat**
  - vzhledem k tomu kolik jich je
  - chceme prostor pro redukci paměti
- **Stav objektů může být uchováván kontextuálně**
  - např. znak se může svůj font dozvědět od svého řádku
- **Po odstranění kontextuálního stavu zůstane malá množina různých objektů**
- **Aplikace nezávisí na identitě objektů**
  - metoda `Equals()` vrátí `true` pro dva výskyty téhož znaku

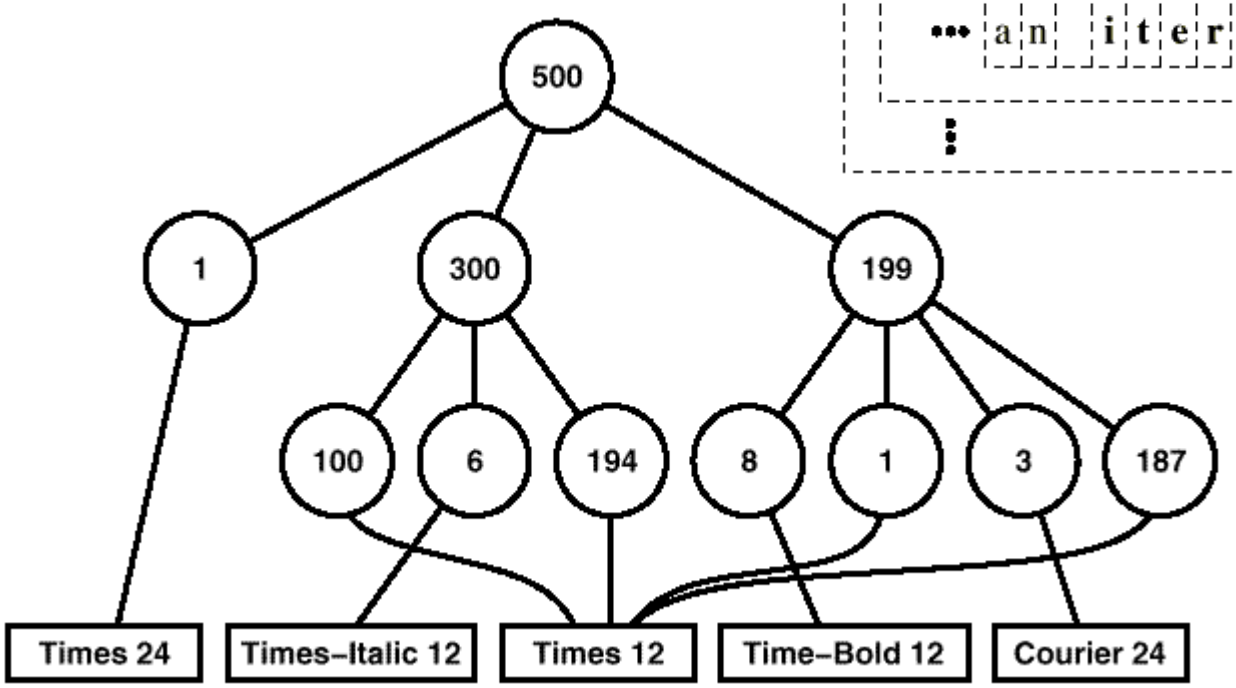
Pravidla pro implementaci

Implementace vyžaduje úplné oddělení vnitřního stavu od vnějšího kontextu pro každý objekt flyweight.

- **Kontext**
  - uchovávat jako metadata
  - vyhledat a dopočítat pro konkrétní flyweight
  - předat jako parametr při volání flyweightu
- **Vnitřní stav**
  - minimalizovat rozmanitost
  - datová položka flyweightu
  - nic víc než identita
- **Správa sdílených flyweightů**
  - instance vytváří jenom factory
  - klienti získávají instance jenom od factory, nevytváří vlastní

Kontext jako metadata





Implementace – 10 000 linek různých barev

```
Graphics graphics = canvas.GetGraphics();
for(int i = 0; i < 10000; i++) {
  Line line = LineFactory.GetLine(GetRandomColor());
  line.Draw(graphics, randomX(), randomY(), randomX(), randomY());
}
```

klient

```
public class LineFactory {
  private static Dictionary<Color, Line> linesByColor = new ...;
  public static Line GetLine(Color color) {
    if(!linesByColor.ContainsKey(color))
        linesByColor.Add(color, new Line(color));
    return line;
  }
}
```

```
public class Line {
  private Color color;
  public Line(Color color) {
    this.color = color;
  }
  public void Draw(Graphics graphics, int x1, int y1, int x2, int y2) {
    graphics.SetColor(color);
    graphics.DrawLine(x1, y1, x2, y2);
  }
}
```

identita = interní stav

kontext

Shrnutí

Návrhový vzor Flyweight je užitečný jenom pro aplikace s velkým počtem jednoduchých a podobných objektů.

- **Příklady použití**
  - textové editory
  - boxing malých *intů* v Javě
  - dekorace ve hrách
  - obrázky na webové stránce
  - buttony, scrollbary, checkboxy…
- **Výhody**
  - úspora paměti
  - méně instancí = menší zátěž GC a alokátoru
- **Nevýhody**
  - dopočítávání kontextu více vytíží CPU
  - specifické použití