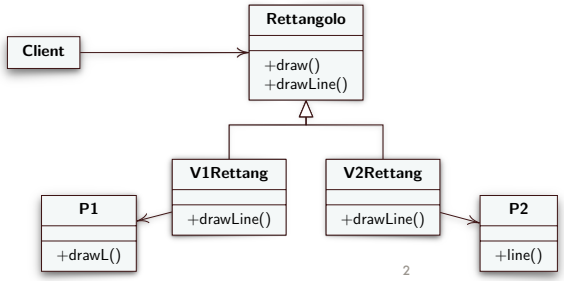
Bridge

- **Intento**
  - Disaccoppiare una astrazione dalla sua implementazione così che le due possano variare indipendentemente
- **Motivazione**
  - Quando una astrazione può avere varie implementazioni, di solito si usa l'ereditarietà. Una classe astratta definisce l'interfaccia dell'astrazione, le sottoclassi concrete la implementano in modi diversi
  - Questo approccio non è flessibile poiché collega l'astrazione all'implementazione permanentemente, e rende difficile modificare, estendere e usare astrazioni e implementazioni indipendentemente

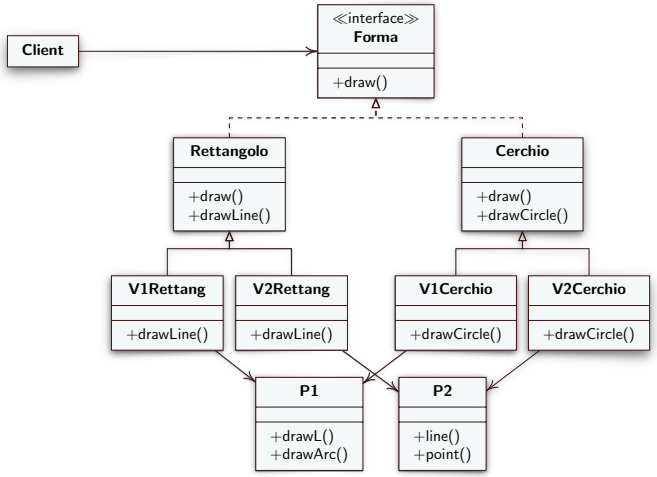
Bridge

- Esempio
  - In un sistema, occorre avere Rettangoli e Cerchi (astrazioni). Inoltre, occorre che Rettangoli e Cerchi siano disponibili per due piattaforme P1 e P2 (implementazioni)
  - La piattaforma P1 mette a disposizione drawL() per disegnare linee, mentre la piattaforma P2 fornisce line()
- Struttura classi iniziale (prima di usare Bridge)



Esempio

- Struttura classi iniziale (prima di usare Bridge)



Considerazioni

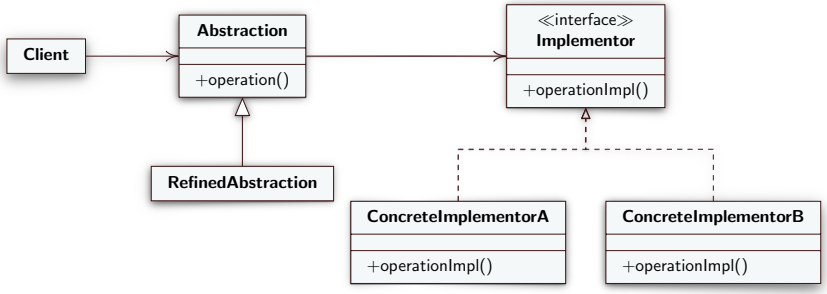
- La soluzione appena vista porterebbe ad una proliferazione di classi
- Se introducessi un'altra piattaforma P3 (implementazione) avrei bisogno di 6 classi concrete (2 forme x 3 piattaforme)
- Se introducessi un'altra forma (astrazione), avrei bisogno di 9 classi concrete (3 x 3)
- Per ogni variazione da introdurre vorrei invece un incremento lineare del numero di classi
- Inoltre, presa una classe, per esempio la classe V1Rettang, essa è legata a una certa piattaforma, es. P1, in modo permanente, ovvero una istanza di V1Rettang non può usare una piattaforma diversa da P1

Bridge

- Soluzione
  - **Abstraction** definisce l'interfaccia per i client e mantiene un riferimento ad un oggetto di tipo **Implementor**. **Abstraction** inoltra le richieste del client al suo oggetto **Implementor**
  - **RefinedAbstraction** estende l'interfaccia definita da **Abstraction**
  - **Implementor** definisce l'interfaccia per le classi dell'implementazione. Questa interfaccia non deve corrispondere esattamente ad **Abstraction**, di solito **Implementor** fornisce operazioni primitive, mentre **Abstraction** definisce operazioni di più alto livello basate su tali primitive
  - **ConcreteImplementor** implementa l'interfaccia di Implementor e fornisce le operazioni concrete

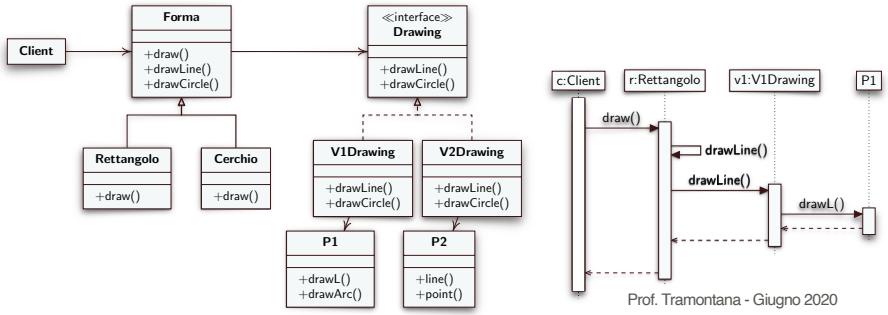
Bridge

- Struttura



Bridge

- La soluzione suggerita dal design pattern Bridge, per il suddetto esempio, avrà il diagramma delle classi disegnato sotto
- `Rettangolo.draw()` chiama `drawLine()` della superclasse `Forma`, quest'ultima chiama `drawLine()` su un'istanza di una sottoclasse di `Drawing`. Analoghe chiamate avvengono per `Cerchio`
- Una nuova classe, es. `Rombo`, sarà implementata come sottoclasse di `Forma`. La classe `Rombo` chiamerà `drawLine()` su `Forma`. Non occorre nessuna nuova classe della gerarchia Implementor



Bridge

- Conseguenze
  - Bridge permette a una implementazione di non essere connessa permanentemente a una interfaccia, l'implementazione può essere configurata e anche cambiata a runtime
  - Il disaccoppiamento permette di cambiare l'implementazione senza dover ricompilare Abstraction ed i Client
  - Solo certi strati del software devono conoscere Abstraction e Implementor
  - I Client non devono conoscere Implementor
  - Le gerarchie di Abstraction e Implementor possono evolvere in modo indipendente

Esempio Minimale Del Bridge

```
// Forma è una Abstraction
public class Forma {
  private Drawing impl;
  public void setImplementor(Drawing imp) {   this.impl = imp;   }
  public void drawLine(int x, int y, int z, int t) {
    impl.drawLine(x, y, z, t);
  }
}
```

```
// Drawing è un Implementor
public interface Drawing {
   public void drawLine(int x1, int y1, int x2, int y2);
}
```

```
// Rettangolo è una RefinedAbstraction
public class Rettangolo extends Forma {
  private int a, b, c, d;
  public Rettangolo(int xi, int yi, int xf, int yf) {
    a = xi; b = yi; c = xf; d = yf;
  }
  public void draw() {
    drawLine(a, b, c, b); drawLine(a, b, a, d);
    drawLine(c, b, c, d); drawLine(a, d, c, d);
  }
}
```


Esempio Di Bridge

- Versione più completa