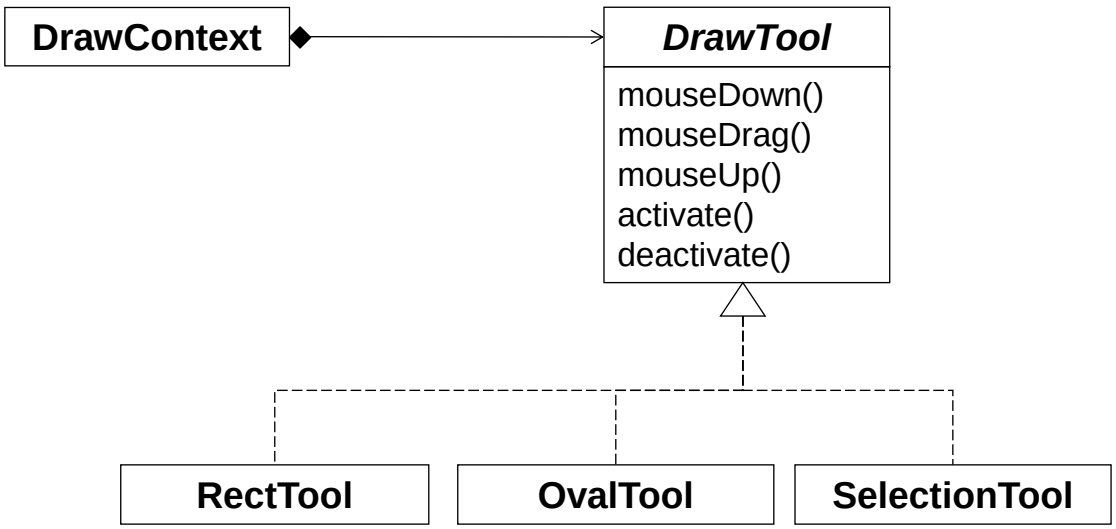
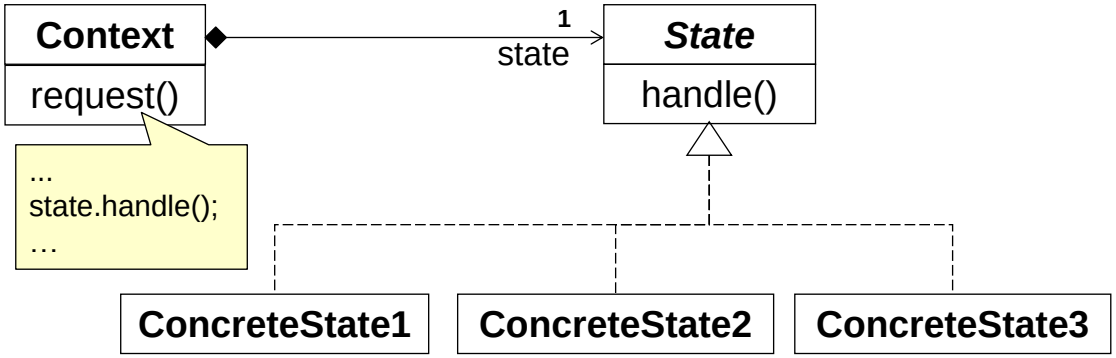
State Pattern

Ziel
Die Auslagerung von zustandsspezifischem Verhalten in einer Klasse. Ermöglicht mit dem Wechsel des Zustandes eines Objektes auch leicht das Verhalten zu ändern.

Motivation
Die View des Grafikeditors hat eine Referenz auf ein abstraktes Tool-Interface. Diese Referenz definiert:
a) Der aktuelle Zeichenmodus
b) Verhalten wenn Maus gedrückt wird



Struktur

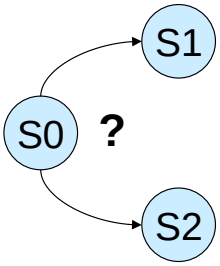


Context:	enthält Referenz auf konkrete Unterklasse von State welche den aktuellen Zustand repräsentiert.
State:	Definiert Interface für das Verhalten in Abhängigkeit des Zustandes.
Concrete State:	Implementiert zustandsspezifisches Verhalten.

Bemerkungen

Setzen des Zustandes:

- Zustandsobjekt löst selber Zustandsänderung aus



- S0 muss die anderen Zustände (S1, S2) kennen
- S0 muss Zugriff auf den Kontext haben damit der neue Zustand gesetzt werden kann.

- Zustandsänderung durch Kontext

  Zustandsmaschine im Kontext definiert, z.B. als Tabelle
  + Konfigurierbar
  + Trennung State/Transition
  + Änderung an Zustandsmaschine möglich ohne States anzufassen

- Zustandsänderung „von aussen“

DrawView:
```
void setTool (DrawTool tool) {
  if (tool != null) {
    this.tool.deactivate();
    this.tool = tool;
    this.tool.activate();
  } else {
    // nur falls null kein gültiger Zustand ist
    throw new IllegalArgumentException();
  }
}
```

Der Test, der prüft ob der Parameter nicht `null` ist, sichert, dass nach dem Aufruf von `setTool` ein Tool gesetzt ist. Der Code, der das Tool verwendet, muss so nicht jedes Mal prüfen, ob aktuell ein Tool gesetzt ist oder nicht (Invariante: *this.tool != null*). Damit diese Invariante immer gilt, muss bereits im Konstruktor ein Default-Tool gesetzt werden.

Anwendung

Man erkennt, dass man ein State-Pattern anwenden sollte, wenn im Code an verschiedenen Stellen if-Anweisungen vorkommen, die von *einem* Attribute abhängen

Beispiel:
```
if (state == RECTANGLE) {
  // Rectangle related code here
else if (state == CIRCLE) {
  // Circle related code here
}
```