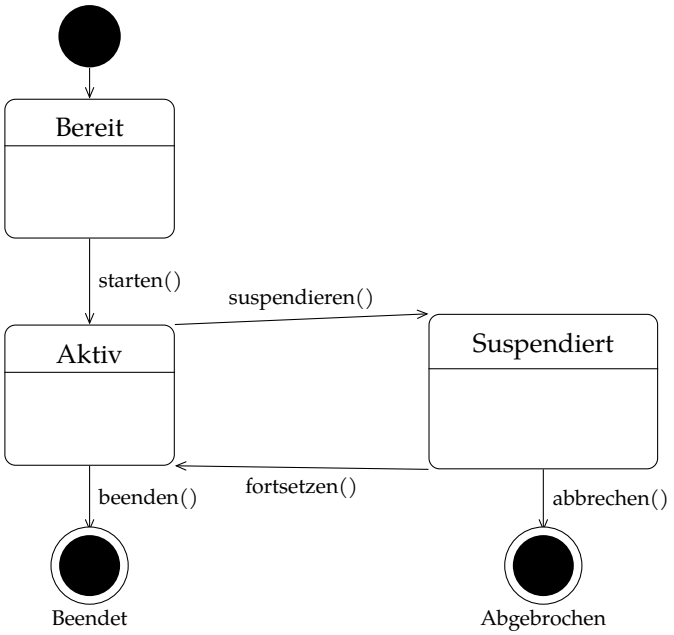
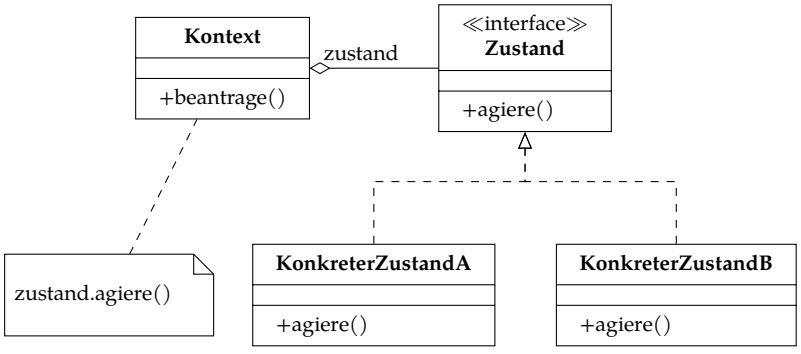
Aufgabe 4: (Entwurfsmuster)

Zu den Aufgaben eines Betriebssystems zählt die Verwaltung von Prozessen. Jeder Prozess durchläuft verschiedene Zustände; Transitionen werden durch Operationsaufrufe ausgelöst. Folgendes Zustandsdiagramm beschreibt die Verwaltung von Prozessen:



Implementieren Sie dieses Zustandsdiagramm in einer Programmiersprache Ihrer Wahl mit Hilfe des Zustandsmusters; geben Sie die gewählte Sprache an. Die Methoden für die Transitionen sollen dabei die Funktionalität der Prozessverwaltung simulieren, indem der Methodenaufruf auf der Standardausgabe protokolliert wird. Falls Transitionen im aktuellen Zustand undefiniert sind, soll eine Fehlermeldung ausgegeben werden.

Exkurs: Zustand-(State)-Entwurfsmuster
UML-Klassendiagramm



Teilnehmer

Kontext (Context) definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

State (Zustand) definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

KontreterZustand (ConcreteState) implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Implementierung in der Programmiersprache „Java“:

Methoden	Zustände	Klassennamen
	Bereit	ZustandBereit
starten(), fortsetzen()	Aktiv	ZustandAktiv
suspendieren()	Suspendiert	ZustandSuspendiert
beenden()	Beendet	ZustandBeendet
abbrechen()	Abgebrochen	ZustandAbgebrochen

```
3  /**
4   * Entspricht der „Kontext"-Klasse in der Terminologie der „Gang of
5   * Four".
6   */
7  public class Prozess {
8
9    private ProzessZustand aktuellerZustand;
10
11   public Prozess() {
12     aktuellerZustand = new ZustandBereit(this);
13   }
14
15   public void setzeZustand(ProzessZustand zustand) {
16     aktuellerZustand = zustand;
17   }
18
19   public void starten() {
20     aktuellerZustand.starten();
21   }
22
23   public void suspendieren() {
24     aktuellerZustand.suspendieren();
25   }
26
27   public void fortsetzen() {
28     aktuellerZustand.fortsetzen();
29   }
30
31   public void beenden() {
32     aktuellerZustand.beenden();
33   }
34
35   public void abbrechen() {
36     aktuellerZustand.abbrechen();
37   }
38
39   public static void main(String[] args) {
40     Prozess prozess = new Prozess();
41     prozess.starten();
```

```
42      prozess.suspendieren();
43      prozess.fortsetzen();
44      prozess.beenden();
45      prozess.starten();
46
47      // Ausgabe:
48      // Der Prozess ist im Zustand „bereit"
49      // Der Prozess wird gestartet.
50      // Der Prozess ist im Zustand „aktiv"
51      // Der Prozess wird suspendiert.
52      // Der Prozess ist im Zustand „suspendiert"
53      // Der Prozess wird fortgesetzt.
54      // Der Prozess ist im Zustand „aktiv"
55      // Der Prozess wird beendet.
56      // Der Prozess ist im Zustand „beendet"
57      // Im Zustand „beendet" kann die Transition „starten" nicht ausführt
        ↪  werden!
58    }
59  }
```

```
3   /**
4    * Entspricht der „Zustand"-Klasse in der Terminologie der "Gang of
5    * Four".
6    */
7   abstract class ProzessZustand {
8
9     Prozess prozess;
10
11    String zustand;
12
13    public ProzessZustand(String zustand, Prozess prozess) {
14      this.zustand = zustand;
15      this.prozess = prozess;
16      System.out.println(String.format("Der Prozess ist im Zustand „%s"",
        ↪  zustand));
17    }
18
19    private void gibFehlermeldungAus(String transition) {
20      System.err.println(
21
            ↪  String.format("Im Zustand „%s" kann die Transition „%s" nicht ausführt werden!",
22              zustand, transition));
23    }
24
25    public void starten() {
26      gibFehlermeldungAus("starten");
27    }
28
29    public void suspendieren() {
30      gibFehlermeldungAus("suspendieren");
31    }
32
33    public void fortsetzen() {
34      gibFehlermeldungAus("fortsetzen");
35    }
36
37    public void beenden() {
38      gibFehlermeldungAus("beenden");
39    }
40
41    public void abbrechen() {
```

```
42        gibFehlermeldungAus("abbrechen");
43      }
44    }

3   /**
4    * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der
     ↪  "Gang of
5    * Four".
6    */
7   public class ZustandAbgebrochen extends ProzessZustand {
8
9     public ZustandAbgebrochen(Prozess prozess) {
10      super("abgebrochen", prozess);
11    }
12  }

3   /**
4    * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der
     ↪  „Gang of
5    * Four".
6    */
7   public class ZustandAktiv extends ProzessZustand {
8
9     public ZustandAktiv(Prozess prozess) {
10      super("aktiv", prozess);
11    }
12
13    public void suspendieren() {
14      System.out.println("Der Prozess wird suspendiert.");
15      prozess.setzeZustand(new ZustandSuspendiert(prozess));
16    }
17
18    public void beenden() {
19      System.out.println("Der Prozess wird beendet.");
20      prozess.setzeZustand(new ZustandBeendet(prozess));
21    }
22  }

3   /**
4    * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der
     ↪  „Gang of
5    * Four".
6    */
7   public class ZustandBeendet extends ProzessZustand {
8
9     public ZustandBeendet(Prozess prozess) {
10      super("beendet", prozess);
11    }
12  }

3   /**
4    * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der
     ↪  „Gang of
5    * Four".
6    */
7   public class ZustandBereit extends ProzessZustand {
8
9     public ZustandBereit(Prozess prozess) {
10      super("bereit", prozess);
11    }
12
```

```
13      public void starten() {
14        System.out.println("Der Prozess wird gestartet.");
15        prozess.setzeZustand(new ZustandAktiv(prozess));
16      }
17    }
```

```
3   /**
4    * Entspricht der „KonkreterZustand"-Unterklasse in der Terminologie der
     ↪ „Gang of
5    * Four".
6    */
7   public class ZustandSuspendiert extends ProzessZustand {
8
9     public ZustandSuspendiert(Prozess prozess) {
10      super("suspendiert", prozess);
11    }
12
13    public void fortsetzen() {
14      System.out.println("Der Prozess wird fortgesetzt.");
15      prozess.setzeZustand(new ZustandAktiv(prozess));
16    }
17
18    public void abbrechen() {
19      System.out.println("Der Prozess wird abgebrochen.");
20      prozess.setzeZustand(new ZustandAbgebrochen(prozess));
21    }
22  }
```