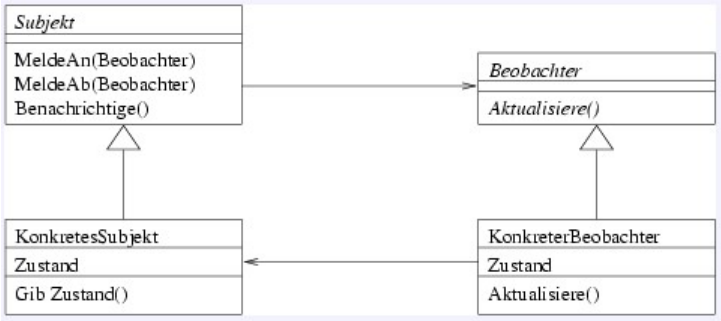
Aufgabe: Entwurfsmuster

Ihr Chef übergibt Ihnen die Aufgabe ein Objekt zu programmieren, dass den Status der Kaffeemaschine (Kanne voll/leer) verwalten kann. Die Kaffeemaschine verfügt über eine USB-Schnittstelle und ihr Status kann über eine mitgelieferte DLL-Datei ausgelesen werden. Außerdem sollen alle Mitarbeiter des Teams aktiv über den Status der Kaffeemaschine benachrichtigt werden können.
Programmieren Sie eine einfache Kaffeemaschinensimulation in einer Programmiersprache Ihrer Wahl. Verwenden Sie für die aktive Benachrichtigung der Mitarbeiter das **Observer-Pattern**.



Lösung in C#:

```
class Program
    {
        static void Main(string[] args)
        {
            KonkretesSubject s = new KonkretesSubject();
            KonkreterBeobachter b1 = new KonkreterBeobachter();
            KonkreterBeobachter b2 = new KonkreterBeobachter();

            b1.Aktualisiere(s);
            b2.Aktualisiere(s);
            s.SetzeZustand(1);
            s.SetzeZustand(2);
            Console.ReadKey();
        }
    }

public abstract class Subject
    {
        public abstract void Benachrichtige();
        public abstract void MeldeAb(Beobachter b);
        public abstract void MeldeAn(Beobachter b);

    }

    public class KonkretesSubject:Subject
    {
        private System.Collections.SortedList beobachterList = new
System.Collections.SortedList();
        private int subjektZustand = 0;
        private int bCnt = 0;

        public override void Benachrichtige()
        {
            foreach (Beobachter b in beobachterList.GetValueList())
            {
                b.Aktualisiere();
            }
```

```
        }

        public override void MeldeAb(Beobachter b)
        {
            beobachterList.Remove(b);
            if (bCnt > 0) { bCnt--; }
        }

        public override void MeldeAn(Beobachter b)
        {
            bCnt++;
            beobachterList.Add(bCnt, b);
        }

        public int GibZustand()
        {
            return subjektZustand;
        }

        public void SetzeZustand(int neu)
        {
            subjektZustand = neu;
            Benachrichtige();
        }


    }

public abstract class Beobachter {
       public abstract void Aktualisiere();
       public abstract void Aktualisiere(KonkretesSubject s);
   }


class KonkreterBeobachter:Beobachter
    {
       protected KonkretesSubject Ksubjekt = null;
       private int beobachterZustand;

       public override void Aktualisiere(KonkretesSubject s) {
         if (this.Ksubjekt != null)
           this.Ksubjekt.MeldeAb(this);

         this.Ksubjekt = s;

         if (this.Ksubjekt != null)
           this.Ksubjekt.MeldeAn(this);
       }

       public override void Aktualisiere() {
           beobachterZustand=this.Ksubjekt.GibZustand();
           Console.WriteLine("Zustand: " + beobachterZustand);

       }
   }
```