Entwurfsmuster Singleton

Zunächst die Frage: Muss die Anwendung überhaupt das Singletonmuster verwenden?

Das Problem:

Die Anwendung greift zum Zeichnen mehrfach auf das Zeichenflaeche-Objekt zu. Das geschieht mit dem Ziel, alle Möbelobjekte in der selben Zeichenfläche darzustellen. Wird dazu aber jeweils ein neues Objekt erzeugt, finden wir alle Objekte auf verschiedenen Zeichenflächen wieder. Um das zu verhindern, verwendet man das Singletonmuster.

Singleton

Die Anwendung des Singleton-Entwurfsmusters sichert ab, dass es zu einer Klasse genau eine Instanz geben kann.

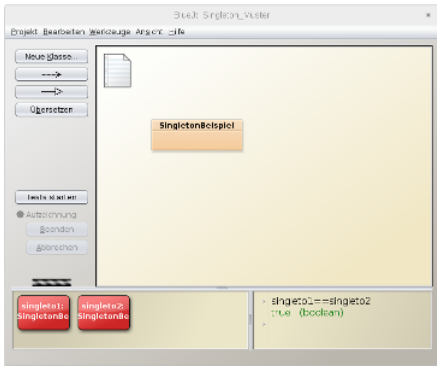
Dazu ist es notwendig, dass der Konstruktor genau einmal aufgerufen werden kann oder dass er immer wieder dasselbe Objekt zurückgibt.
Bei Java wird das gewünschte Ziel dadurch erreicht, dass der Konstruktor mit dem Modifizierer private gekennzeichnet wird und dadurch nicht von außerhalb der Klasse aufgerufen werden kann. Ein Objekt der Klasse kann nur durch eine Klassenmethode erhalten werden. Dazu muss die Klasse eine Klassenvariable (static) haben, in der das einzige existierende Objekt gehalten wird und das beim Aufruf der Klassenmethode zurückgeliefert wird, wenn es bereits existiert. Anderenfalls ruft die Klassenmethode vorher den Konstruktor auf.

Mustertext für Java:

```
public class SingletonBeispiel {
    // Klassenvariable
    private static SingletonBeispiel dasSingletonObjekt = null;

    /**
     * Eine Klassenmethode liefert das einzige Exemplar,
     * erzeugt es gegebenenfalls
     */
    public static SingletonBeispiel gibDasSingleton() {
        if (dasSingletonObjekt == null)
        dasSingletonObjekt = new SingletonBeispiel();
        return dasSingletonObjekt;
    }

    /**
     * Konstruktor für Objekte der Klasse SingletonBeispiel ist private
     */
    private SingletonBeispiel() {
        // keine weiteren Initialisierungen
    }
}
```



Problem bei Python

Bei der Umsetzung gibt es ein Problem bei Python: Der Konstruktor __init__ kann nicht als private gekennzeichnet werden. Es gibt einige Überlegungen, wie man bei Python das Muster umsetzen kann[1]. Ich schlage dazu die folgende leicht verständliche Lösung vor:

Mustertext für Python

Eine Musterklasse

```
class SingletonBeispiel(object):
    """Die Klasse wendet das Singleton-Muster an, also kein Konstruktoraufruf,
    sondern Methode GibDasSingleton()"""

    __dasSingletonObjekt = None

    @staticmethod
    def GibDasSingleton():
        if SingletonBeispiel.__dasSingletonObjekt == None:
            SingletonBeispiel() # erzeugt und sichert selbst
        return SingletonBeispiel.__dasSingletonObjekt

    def __init__(self):
        """Konstruktor !!!nicht!!! direkt aufrufen!"""
        if SingletonBeispiel.__dasSingletonObjekt != None:
            raise Exception('Konstruktor nicht direkt aufrufen')
        SingletonBeispiel.__dasSingletonObjekt = self
```

Die Tests:

```
beispiel0=SingletonBeispiel() # nicht schoen, aber zulaessig
beispiel1=SingletonBeispiel.GibDasSingleton()
beispiel2=SingletonBeispiel.GibDasSingleton()
print beispiel0,'\n', beispiel1,'\n', beispiel2
try: SingletonBeispiel()
except Exception as e:
    print str(e)
```

Sie liefern:

```
<__main__.SingletonBeispiel object at 0x7fd6c9822590>
<__main__.SingletonBeispiel object at 0x7fd6c9822590>
<__main__.SingletonBeispiel object at 0x7fd6c9822590>
Konstruktor nicht direkt aufrufen
```

1 Siehe dazu das Muster Borg.