|  |
| --- |
| SEW |
| Strategy Pattern |
| Linked List Speichermethode |

|  |
| --- |
| Felix  [Datum]  1h |

# Inhaltsverzeichnis

Inhalt

[Inhaltsverzeichnis 1](#_Toc56699300)

[Angabe 1 1](#_Toc56699301)

[Quellcode 1 1](#_Toc56699302)

[Testfälle 1 1](#_Toc56699303)

[Angabe 2 1](#_Toc56699304)

[Klassendiagramm 2 1](#_Toc56699305)

[Quellcode 2 1](#_Toc56699306)

[Testfälle 2 1](#_Toc56699307)

[Angabe 3 1](#_Toc56699308)

[Klassendiagramm 3 1](#_Toc56699309)

[Quellcode 3 1](#_Toc56699310)

[Testfälle 3 1](#_Toc56699311)

# Angabe 1 <<…>>

Strategy Pattern Enten programmieren

# Theorie 1 <<…>>

Strategy Pattern hat eine Variable eines Interfaces, die auf QuackVerhalten, Schwimmverhalten der von dem Interface erbenden Klassen zeigt, sodass immer bei dem Aufruf von der Methode der Variable die Methode der richtigen Klasse aufgerufen wird.

# Klassendiagramm 1 <<…>>

# Quellcode 1 <<…>>

using System;

using System.Collections.Generic;

using System.Text;

namespace StrategyPattern

{

interface IFlyBehaviour

{

void Fly();

}

class FlyWithWings : IFlyBehaviour

{

public void Fly()

{

Console.WriteLine("I'm flying!");

}

}

class FlyNoWay : IFlyBehaviour

{

public void Fly()

{

Console.WriteLine("I can't fly!");

}

}

interface IQuackBehaviour

{

void Quack();

}

class QuackLoud : IQuackBehaviour

{

public void Quack()

{

Console.WriteLine("Quaaaaaaack!");

}

}

class QuackQuiet : IQuackBehaviour

{

public void Quack()

{

Console.WriteLine("<< Quiet >>: Quack!");

}

}

class QuackNoWay : IQuackBehaviour

{

public void Quack()

{

Console.WriteLine("<< Silence >>");

}

}

abstract class ADuck

{

private IFlyBehaviour FlyBehaviour;

private IQuackBehaviour QuackBehaviour;

public void SetFlyBehaviour(IFlyBehaviour behaviour)

{

FlyBehaviour = behaviour;

}

public void SetQuackBehaviour(IQuackBehaviour behaviour)

{

QuackBehaviour = behaviour;

}

public void PerformFly()

{

FlyBehaviour.Fly();

}

public void PerformQuack()

{

QuackBehaviour.Quack();

}

}

class MallardDuck : ADuck

{

public MallardDuck(IFlyBehaviour flybehaviour, IQuackBehaviour quackbehaviour)

{

SetFlyBehaviour(flybehaviour);

SetQuackBehaviour(quackbehaviour);

}

public MallardDuck():this(new FlyWithWings(), new QuackLoud()) { }

}

class DecoyDuck : ADuck

{

public DecoyDuck(IFlyBehaviour flybehaviour, IQuackBehaviour quackbehaviour)

{

SetFlyBehaviour(flybehaviour);

SetQuackBehaviour(quackbehaviour);

}

public DecoyDuck() : this(new FlyNoWay(), new QuackNoWay()) { }

}

}