

Decorator Pattern

Decorator er et godt alternativ til underklasser når man vil udvide et objects funktionalitet.

Component (IUnit)

- Interface til de objekter en decorator kan tilføje funktionalitet til.

ConcreteComponent (Soldier, Tank)

- Det objekt en decorator kan tilføjet til.

Decorator (UnitDecorator)

- Har en reference til et IUnit object og metoder der svarer til IUnits metoder.

ConcreteDecorator (WeaponUpgrade, ArmorUpgrade)

- Det objekt som tilføjer funktionalitet til en IUnit.

```
interface IUnit
{
    int GetDamage();
    int GetArmor();
}
```

Decorator er et godt alternativ til underklasser når man vil udvide et objects funktionalitet.

Component (IUnit)

- Interface til de objekter en decorator kan tilføje funktionalitet til.

ConcreteComponent (Soldier, Tank)

- Det objekt en decorator kan tilføjet til.

Decorator (UnitDecorator)

- Har en reference til et IUnit object og metoder der svarer til IUnits metoder.

ConcreteDecorator (WeaponUpgrade, ArmorUpgrade)

- Det objekt som tilføjer funktionalitet til en IUnit.

```
class Soldier : IUnit
{
    public int GetArmor()
    {
        return 0;
    }

    public int GetDamage()
    {
        return 5;
    }
}
```

```
class Tank : IUnit
{
    public int GetArmor()
    {
        return 10;
    }

    public int GetDamage()
    {
        return 5;
    }
}
```

Decorator er et godt alternativ til underklasser når man vil udvide et objects funktionalitet.

Component (IUnit)

- Interface til de objekter en decorator kan tilføje funktionalitet til.

ConcreteComponent (Soldier, Tank)

- Det objekt en decorator kan tilføjet til.

Decorator (UnitDecorator)

- Har en reference til et IUnit object og metoder der svarer til IUnits metoder.

ConcreteDecorator (WeaponUpgrade, ArmorUpgrade)

- Det objekt som tilføjer funktionalitet til en IUnit.

```
class UnitDecorator : IUnit
{
    private IUnit unit;

    public UnitDecorator(IUnit unit)
    {
        this.unit = unit;
    }

    public virtual int GetArmor()
    {
        return unit.GetArmor();
    }

    public virtual int GetDamage()
    {
        return unit.GetDamage();
    }
}
```

Decorator er et godt alternativ til underklasser når man vil udvide et objects funktionalitet.

Component (IUnit)

- Interface til de objekter en decorator kan tilføje funktionalitet til.

ConcreteComponent (Soldier, Tank)

- Det objekt en decorator kan tilføjet til.

Decorator (UnitDecorator)

- Har en reference til et IUnit object og metoder der svarer til IUnits metoder.

ConcreteDecorator (WeaponUpgrade, ArmorUpgrade)

- Det objekt som tilføjer funktionalitet til en IUnit.

```
class WeaponUpgrade : UnitDecorator
{
    public WeaponUpgrade(IUnit unit) : base(unit)
    {
    }

    public override int GetDamage()
    {
        return base.GetDamage() + 1;
    }
}

class ArmorUpgrade : UnitDecorator
{
    public ArmorUpgrade(IUnit unit) : base(unit)
    {
    }

    public override int GetArmor()
    {
        return base.GetArmor() + 1;
    }
}
```

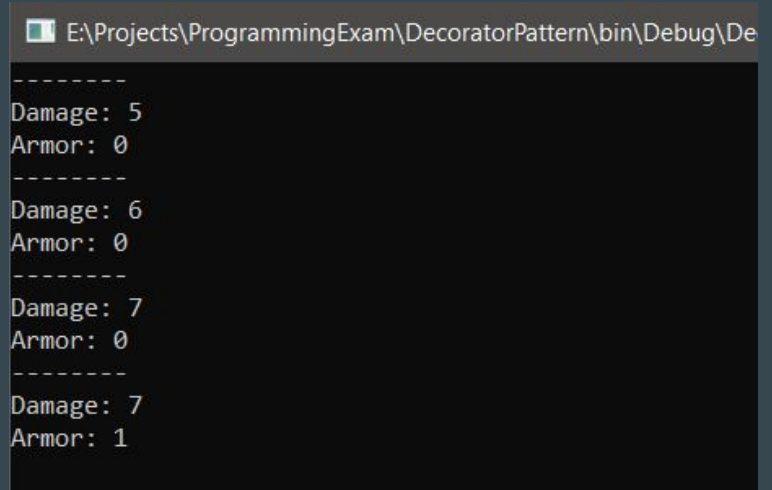
```

static void Main(string[] args)
{
    while (true)
    {
        IUnit soldier = new Soldier();
        ShowUnitStats(soldier);
        soldier = new WeaponUpgrade(soldier);
        ShowUnitStats(soldier);
        soldier = new WeaponUpgrade(soldier);
        ShowUnitStats(soldier);
        soldier = new ArmorUpgrade(soldier);
        ShowUnitStats(soldier);

        Console.ReadLine();
    }
}

private static void ShowUnitStats(IUnit unit)
{
    Console.WriteLine("-----");
    Console.WriteLine("Damage: " + unit.GetDamage());
    Console.WriteLine("Armor: " + unit.GetArmor());
}

```



```

E:\Projects\ProgrammingExam\DecoratorPattern\bin\Debug\De
-----
Damage: 5
Armor: 0
-----
Damage: 6
Armor: 0
-----
Damage: 7
Armor: 0
-----
Damage: 7
Armor: 1

```

