



# Modul- Fortgeschrittene Programmierkonzepte

Bachelor Informatik

## 08- Design Patterns, pt. 2

Prof. Dr. Marcel Tilly

Fakultät für Informatik, Cloud Computing



# Singleton


Structure to enforce the use of a *unique* instance.

 dp-singleton

# Strategy



# Strategy

kara-explorer



# Strategy

```
public class Kara extends JavaKaraProgram {
    public static void main(String[] args) throws Exception {
        Kara k = new Kara();
        k.run("src/main/resources/world2.world");
    }

    @Override
    public void myMainProgram() {
        kara.move();           // one step forward
        kara.turnLeft();       // you guessed it...
        kara.turnRight();
        kara.treeFront();      // tree ahead?
        kara.putLeaf();        // take a clover leaf
        kara.removeLeav();     // remove a clover leaf
    }
}
```

How place leafs on every field?

# Strategy



Mechanism to provide different implementations to achieve the same outcome.

 dp-strategy

# Factory

Technische  
Hochschule  
**Rosenheim**



# Factory



Composite pattern:

```
{  
  "key": "value",  
  "nested": {  
    "key": "value"  
  }  
}
```

```
<element>  
  <key>value</key>  
  <element>  
    <key>value</key>  
  </element>  
</element>
```



# Factory



```
interface Component {
    String toString();
}
interface Composite extends Component {
    void add(Component c);
}
interface Leaf extends Component {
}
```

```
JsonComposite root = new JsonComposite("root");
root.add(new JsonLeaf("key", "value"));

Composite nested = new JsonComposite("nested");
nested.add(new JsonLeaf("key", "value"));
root.add(nested);

System.out.println(root);
// "root": {"key": "value", "nested": {"key": "value"}}
```

# Factory



Structure to enforce the use of abstract factories and products, by hiding the actual instantiation of the concrete factory and products.

 dp-factory

# Command



# Command



Mechanism to organize, execute and undo operations on certain objects.