

CS 525 - ASD

# Advanced Software Development

**MS.CS Program**  
Department of Computer Science  
Rene de Jong, MsC.



Maharishi University  
OF MANAGEMENT

# CS 525 - ASD

## Advanced Software Development


© 2019 Maharishi University of Management

**All course materials are copyright protected by international copyright laws and remain the property of the Maharishi University of Management. The materials are accessible only for the personal use of students enrolled in this course and only for the duration of the course. Any copying and distributing are not allowed and subject to legal action.**



Maharishi University  
OF MANAGEMENT

# Lesson 4

- 
- L1: ASD Introduction
  - L2: Strategy, Template method
  - L3: Observer pattern
  - L4: Composite pattern, iterator pattern
  - L5: Command pattern
  - L6: State pattern
  - L7: Chain Of Responsibility pattern

## Midterm

- L8: Proxy, Adapter, Mediator
- L9: Factory, Builder, Decorator, Singleton
- L10: Framework design
- L11: Framework implementation
- L12: Framework example: Spring framework
- L13: Framework example: Spring framework

## Final

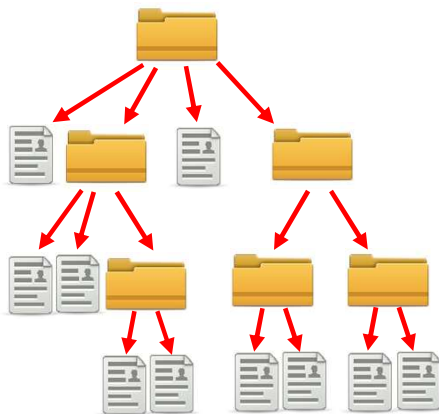
# Composite pattern

---

- The intent of this pattern is to compose objects into tree structures to represent part-whole hierarchies.
- Composite lets clients treat individual objects and compositions of objects uniformly.

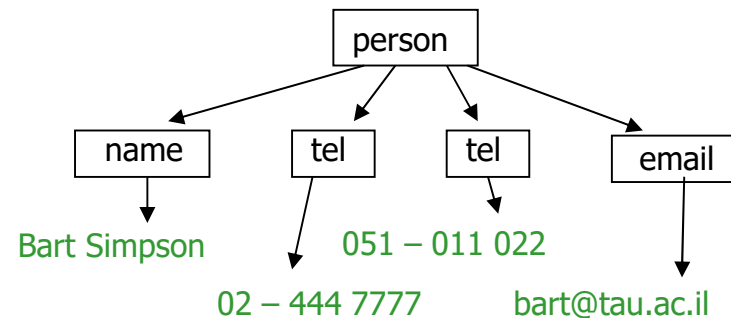
# Tree structures

File system

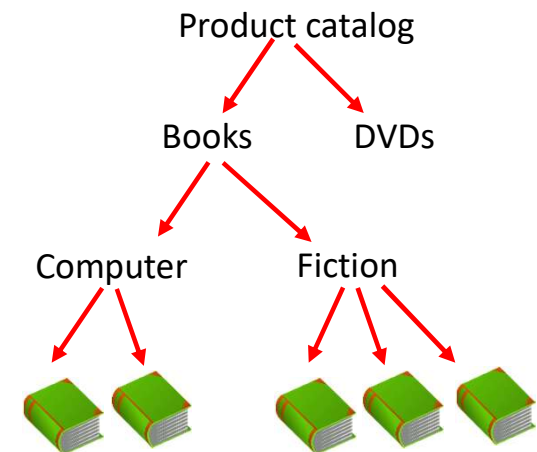


XML structure

```
<person>  
  <name> Bart Simpson </name>  
  <tel> 02 – 444 7777 </tel>  
  <tel> 051 – 011 022 </tel>  
  <email> bart@tau.ac.il </email>  
</person>
```

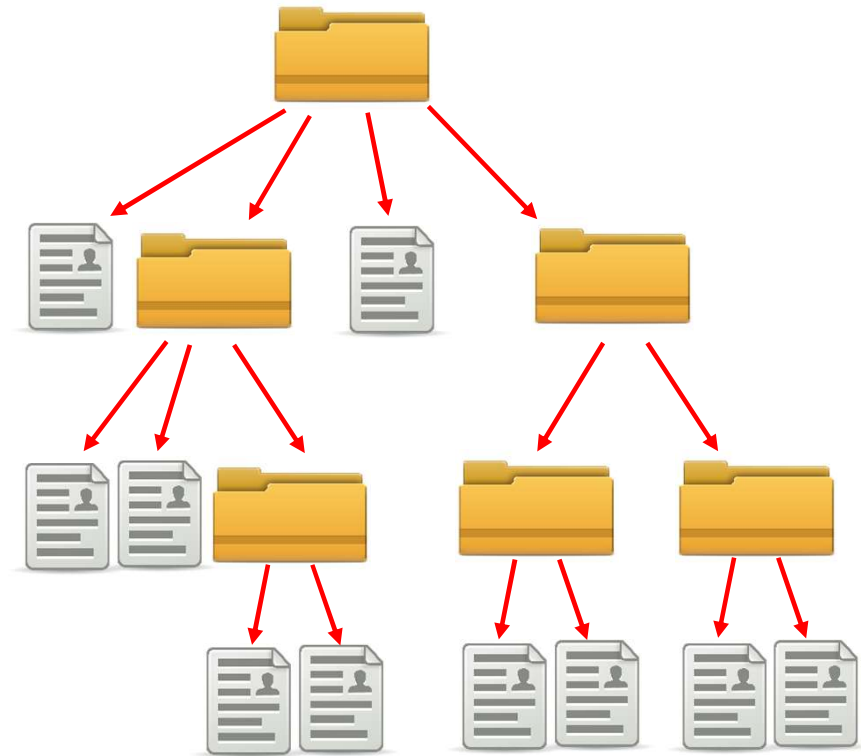


Product catalog

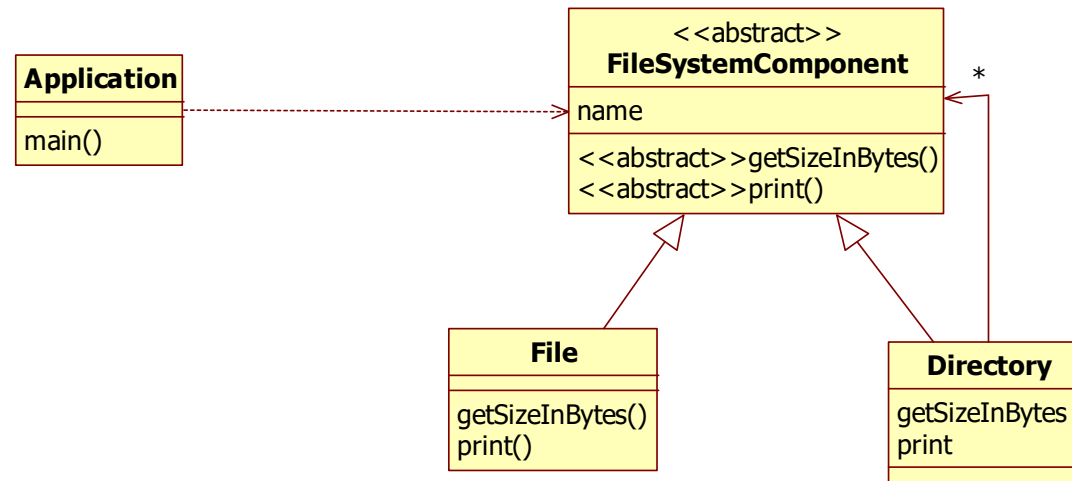


# Example application

- Filesystem
  - Common functionality
    - Compute the size
    - Print whole directory



# Composite pattern



# FileSystemComponent & File

---

```
public abstract class FileSystemComponent {  
    protected String name;  
  
    public FileSystemComponent(String name) {  
        this.name = name;  
    }  
  
    public abstract void print();  
  
    public abstract int getSizeInBytes();  
}
```

```
public class File extends FileSystemComponent {  
    private int sizeInBytes;  
  
    public File(String name, int sizeInBytes) {  
        super(name);  
        this.sizeInBytes = sizeInBytes;  
    }  
  
    public int getSizeInBytes() {  
        return sizeInBytes;  
    }  
  
    public void print() {  
        System.out.println("--- file " + name + " size=" +  
            getSizeInBytes() + " bytes");  
    }  
}
```



# Directory

```
public class Directory extends FileSystemComponent {
    protected Collection<FileSystemComponent> fileSystemComponents = new
        ArrayList<FileSystemComponent>();

    public Directory(String name) {
        super(name);
    }

    public void addComponent(FileSystemComponent component) {
        fileSystemComponents.add(component);
    }

    public int getSizeInBytes() {
        int sizeInBytes = 0;
        for (FileSystemComponent component : fileSystemComponents) {
            sizeInBytes += component.getSizeInBytes();
        }
        return sizeInBytes;
    }

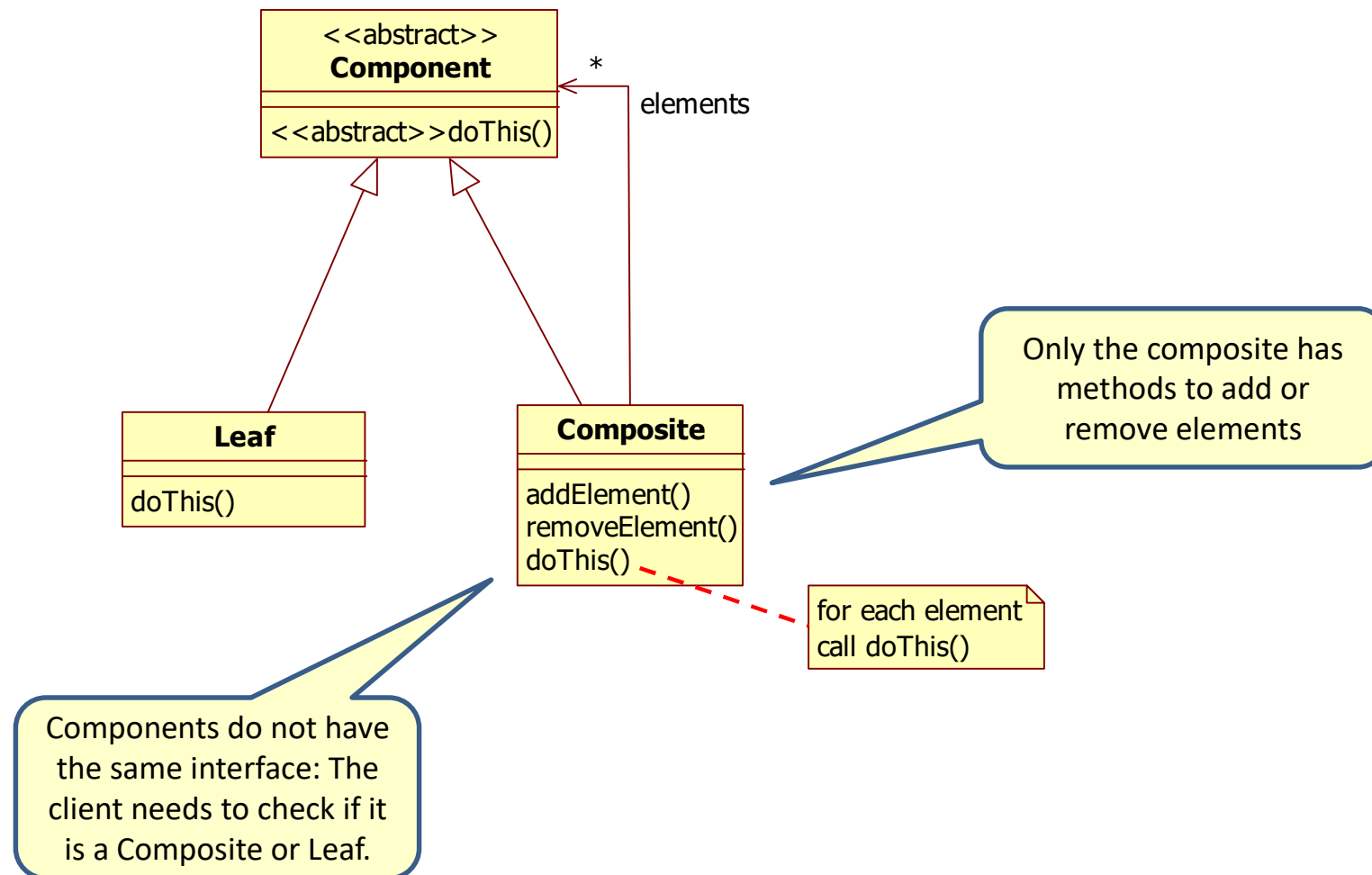
    public void print() {
        System.out.println("-- dir " + name + " size=" + getSizeInBytes() + " bytes");
        for (FileSystemComponent component : fileSystemComponents) {
            component.print();
        }
    }
}
```

# The application

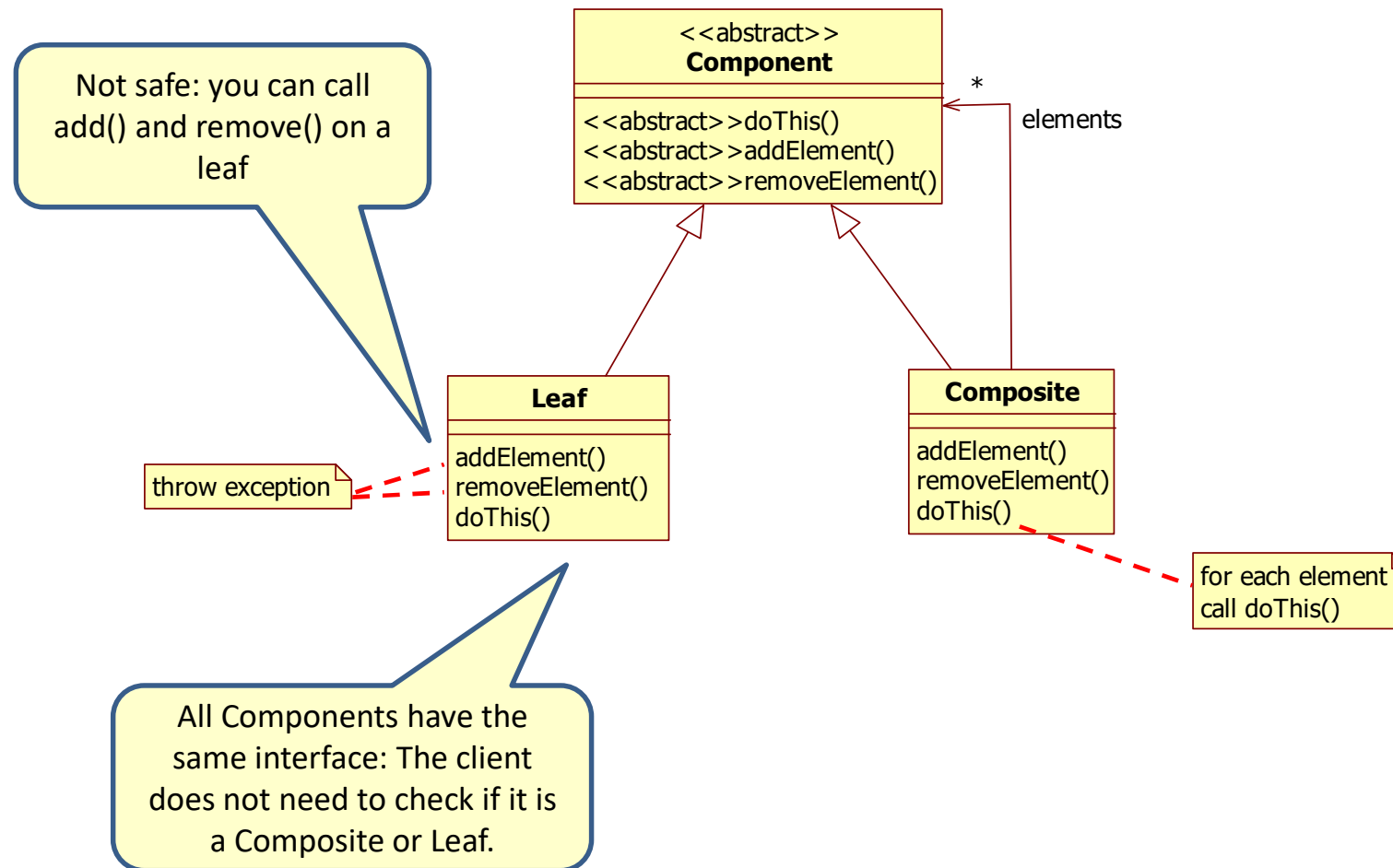
```
public class Application {  
  
    public static void main(String[] args) {  
        Directory cdir = new Directory("C");  
        Directory appdir = new Directory("applications");  
        Directory datadir = new Directory("my data");  
        Directory coursedir = new Directory("cs525");  
        File excelfile = new File("msexcel.exe", 2353256);  
        File wordfile = new File("msword.exe", 3363858);  
        File studentsfile = new File("students.doc", 34252);  
        cdir.addComponent(appdir);  
        cdir.addComponent(datadir);  
        datadir.addComponent(coursedir);  
        appdir.addComponent(excelfile);  
        appdir.addComponent(wordfile);  
        coursedir.addComponent(studentsfile);  
        cdir.print();  
    }  
}
```

```
-- dir C size=5751366 bytes  
-- dir applications size=5717114 bytes  
--- file msexcel.exe size=2353256 bytes  
--- file msword.exe size=3363858 bytes  
-- dir my data size=34252 bytes  
-- dir cs525 size=34252 bytes  
--- file students.doc size=34252 bytes
```

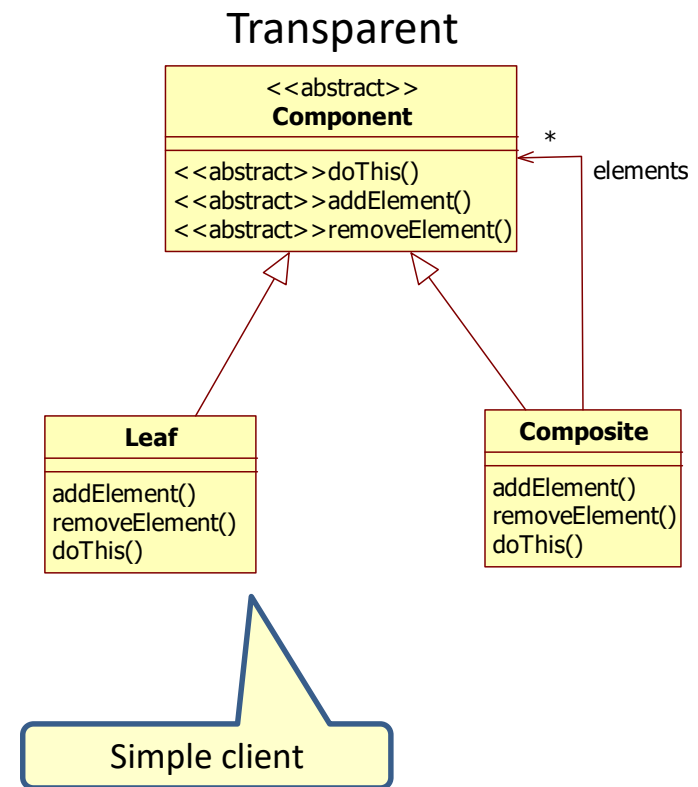
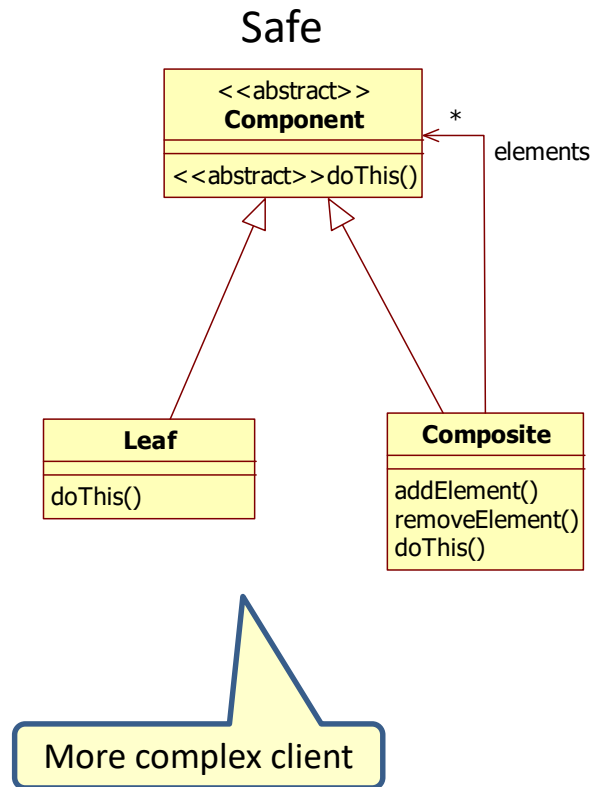
# Safe Composite



# Transparent Composite



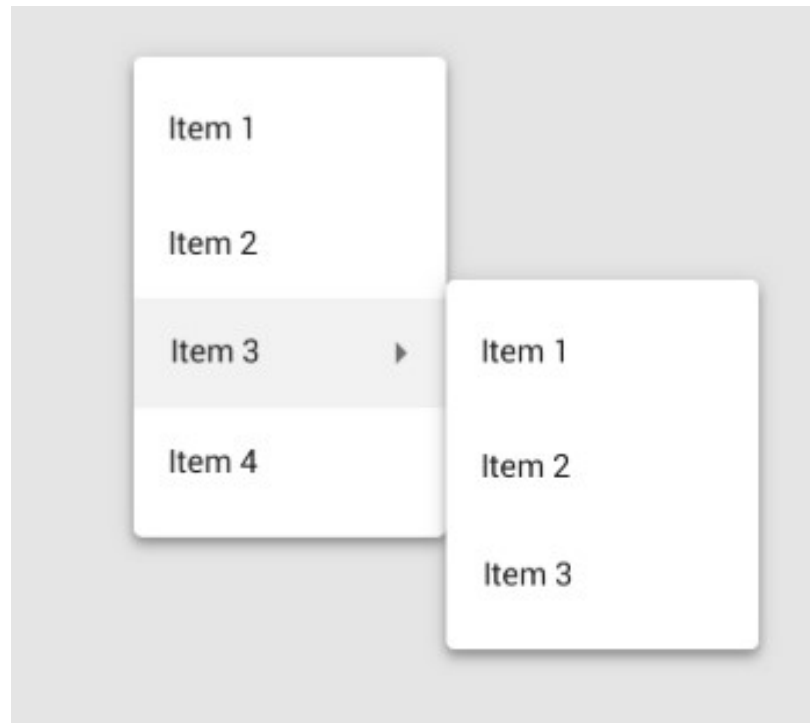
# Safe or transparent



# Example of composite pattern

---

- Menus with sub-menus



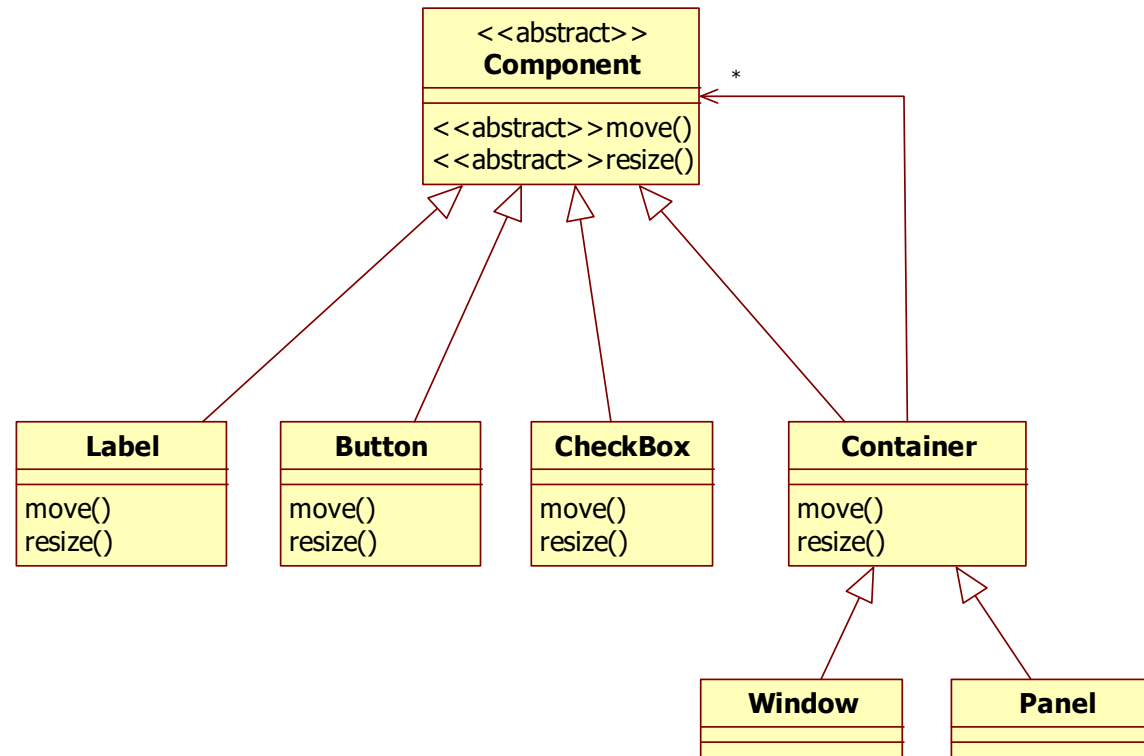
# Example of composite pattern

---

- Drawing consist of simple drawings (lines, circle, rectangle)
- Group and ungroup of drawings
  - Copy-and-paste on simple drawings or grouped drawing
  - Drag-and-drop with simple drawings or grouped drawing

# Example of composite pattern

- UI framework





# Main point

---

- The composite pattern can be used to represent tree structures and we want to handle the tree elements in an unformal manner.
- The structure of the universe emerges from pure consciousness which is the basis of all life.