

# OBJEKTUMVEZÉRT RENDSZEREK TERVEZÉSE

5. gyakorlat

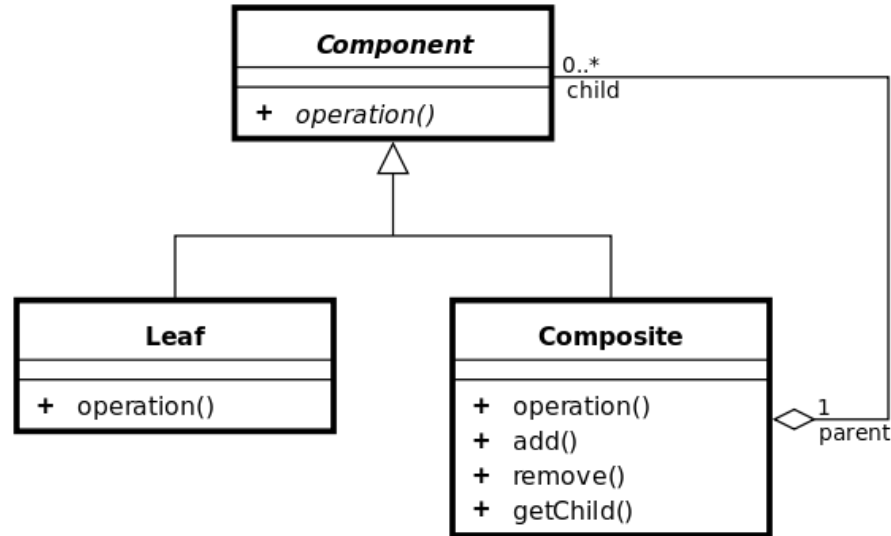
---



# COMPOSITE

Összetétel

---



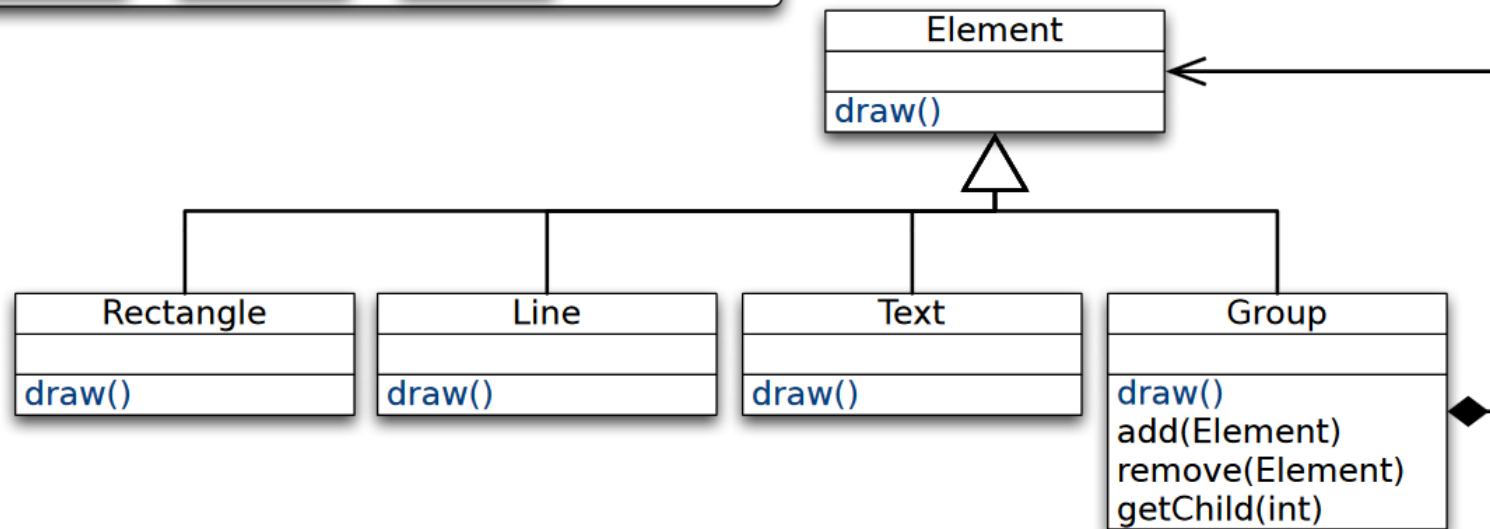
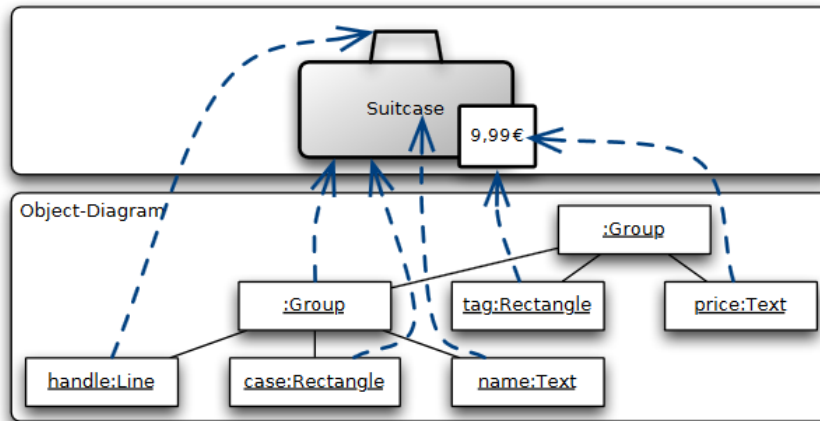
- **Cél:** rész-egész szerkezetek leképezése objektum-hierarchiára (fa-szerkezetre), miközben a kliens a rész és egész egységeket azonosan kezeli
- **Alkalmazhatóság:**
  - rész-egész szerkezetek hierarchiáját kell leírni
  - a kliensnek nem kell hogy észrevegye a különbségeket a kompozíciók és primitívek között

# Composite



Suitcase

9,99€



- Hol használható?
  - Fa-struktúra reprezentálására
    - Csoportba foglalás
    - Hierarchiák
    - Fájlrendszer
  - Ha nem akarunk különbséget tenni összetett és egyedi objektumok között - uniformitás

# Composite

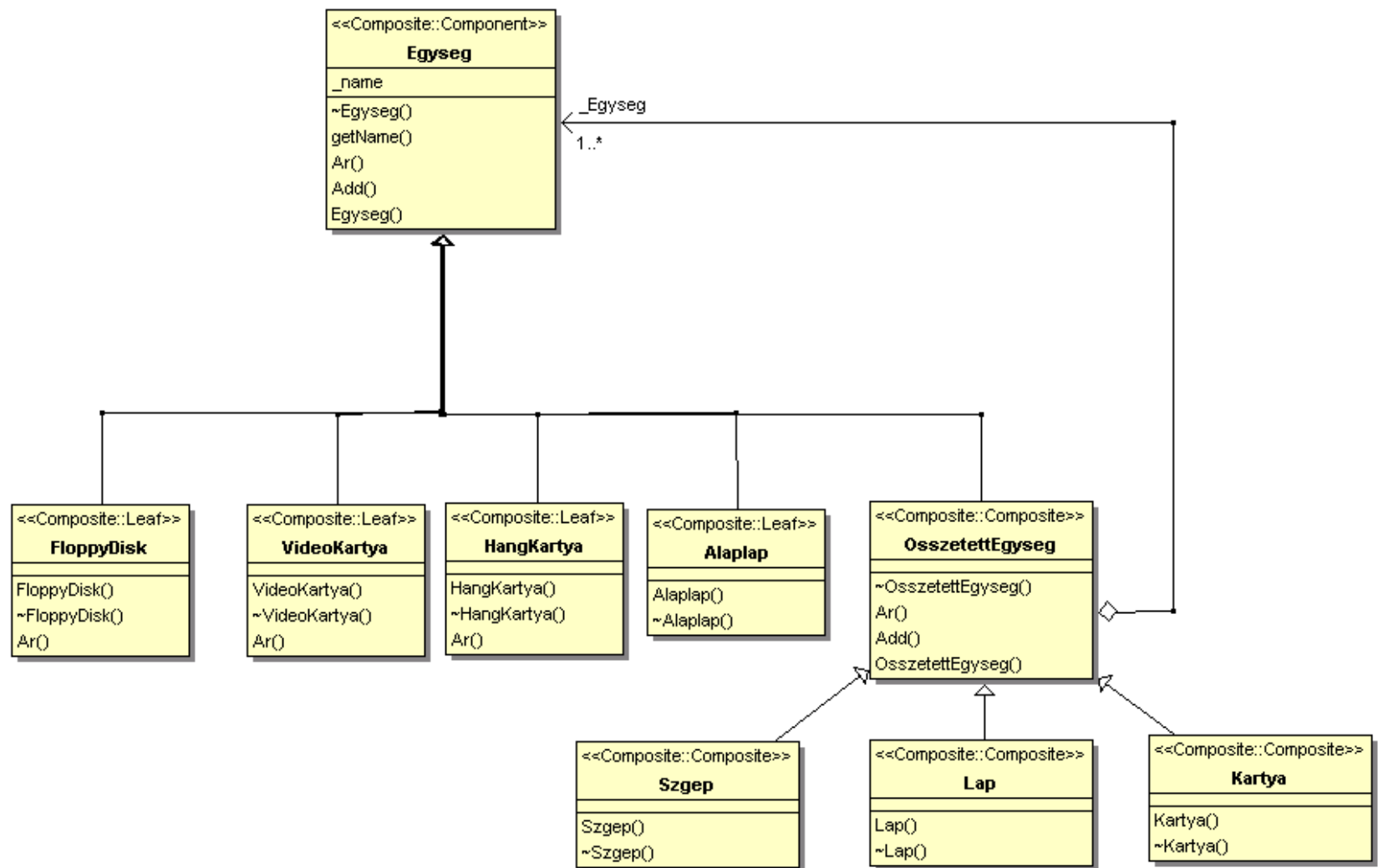
---

- Számítógép alkatrészekből/egységekből épül fel
- Adott számítógép árát az alkatrészek árai határozzák meg
  - Számoljuk ki, de hogyan?
- Próbáljuk megoldani!



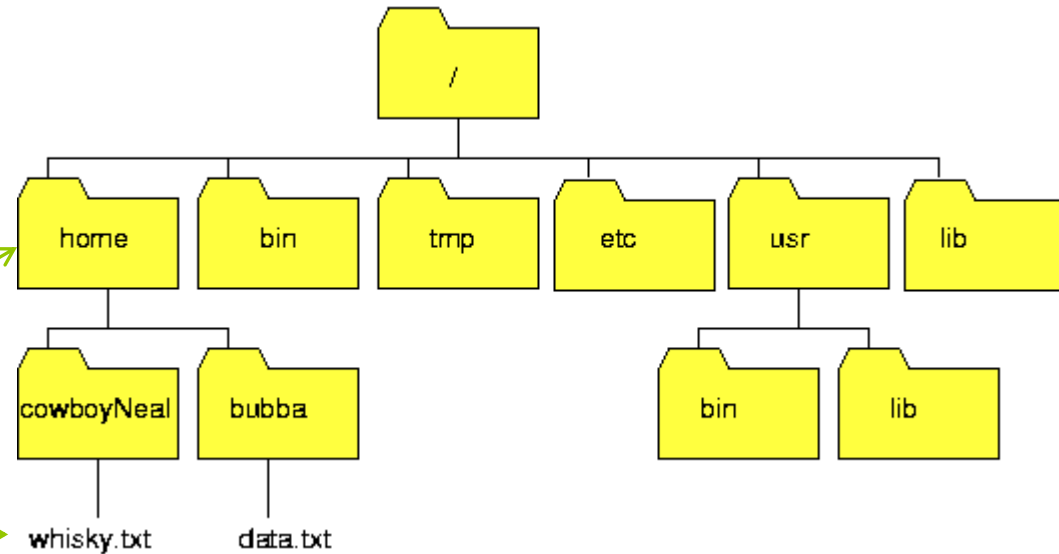
# Composite Példa

---





- Client: **Commander**
- Component: **Resource**
  - Operation: **getSize()**
- Leaf: **File**
- Composite: **Folder**



# Példa: fájlrendszer

## Resource.java

```
1 package hu.u_szeged.inf.ovrt.composite;  
2  
3 public interface Resource {  
4  
5     String getName();  
6  
7     int getSize();  
8  
9     boolean isDirectory();  
10  
11 }
```

---

## File.java

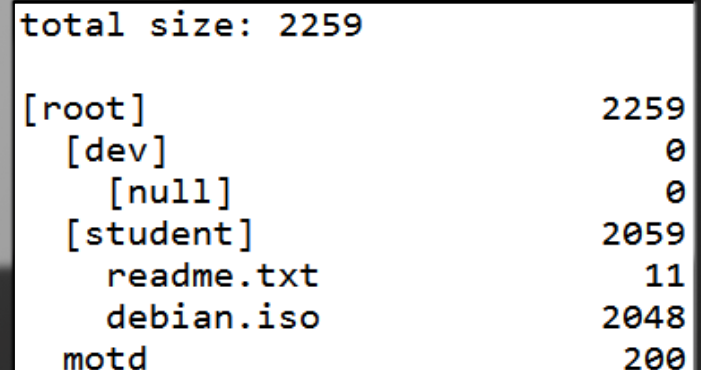
```
3 public class File implements Resource {
4
5     private String name;
6     private int size;
7
8     public File(String name, int size) {
9         super();
10        this.name = name;
11        this.size = size;
12    }
13
14    @Override
15    public String getName() {
16        return this.name;
17    }
18
19    public void setName(String name) {
20        this.name = name;
21    }
22
23    public void setSize(int size) {
24        this.size = size;
25    }
26
27    @Override
28    public int getSize() {
29        return this.size;
30    }
31
32    @Override
33    public boolean isDirectory() {
34        return false;
35    }
36
37    @Override
38    public String toString() {
39        return this.name;
40    }
41
42 }
```

## Folder.java

```
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Folder implements Resource {
7
8     private String name;
9     private List<Resource> resources = new ArrayList<>();
10
11    public Folder(String name) {
12        super();
13        this.name = name;
14    }
15
16    public void addResource(Resource resource) {
17        this.resources.add(resource);
18    }
19
20    public List<Resource> getResources() {
21        return this.resources;
22    }
23
24    @Override
25    public String getName() {
26        return this.name;
27    }
28
29    public void setName(String name) {
30        this.name = name;
31    }
32
33    @Override
34    public int getSize() {
35        int size = 0;
36        for (Resource resource : this.resources) {
37            size += resource.getSize();
38        }
39        return size;
40        // In Java 8 you can use this one line:
41        // return this.resources.stream().mapToInt(
42    }
```

## Commander.java

```
9      public static void main(String[] args) {
10          Folder root = new Folder("root");
11          Folder dev = new Folder("dev");
12          root.addResource(dev);
13          Folder nul = new Folder("null");
14          dev.addResource(nul);
15          Folder user = new Folder("student");
16          root.addResource(user);
17          File readme = new File("readme.txt", 11);
18          File linux = new File("debian.iso", 2048);
19          user.addResource(readme);
20          user.addResource(linux);
21          File welcome = new File("motd", 200);
22          root.addResource(welcome);
23
24          System.out.println("total size: " + root.getSize());
25      }
```



```
total size: 2259

[root]                2259
  [dev]                 0
    [null]              0
  [student]            2059
    readme.txt         11
    debian.iso         2048
    motd               200
```

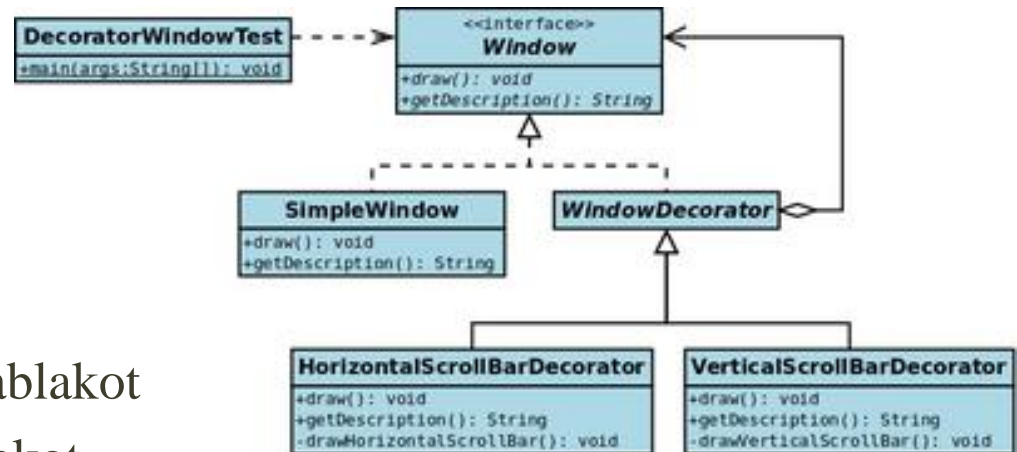


# DECORATOR

Díszítő

---

- Ablakkezelő rendszer
  - A megrendelő szeretne
    - Sima ablakot
    - Horizontális csúszkás ablakot
    - Vertikális csúszkás ablakot
    - Horizontális és vertikális csúszkás ablakot
  - Majd később bejelenti, hogy szeretne ikonos, keretes ablakot is...

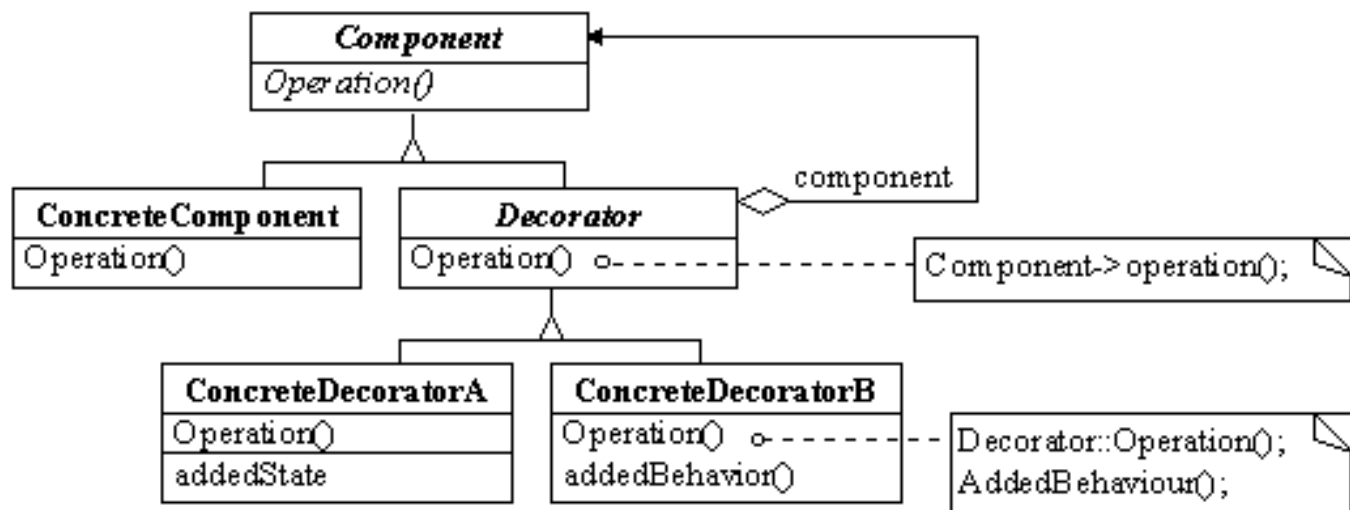


# Decorator Példa

- *Cél:* további felelősségek/tulajdonságok dinamikus csatolása az objektumhoz. Flexibilis alternatíva a funkcionalitás öröklődéses bővítése mellett
- *Alkalmazhatóság:*
  - új felelősségeket kell hozzáadni egyes objektumokhoz dinamikusan, miközben a kliens objektumok ezt transzparensen érzékelik

# Decorator

---



# Decorator



- Hol használható?
  - Felhasználói felületi elemeknél
    - Ablakok
    - Képek, ikonok dinamikus összeállításánál
      - Pl.: Eclipse projektkeresőben fájlok ikonjai
  - Bárhol ahol futás közbeni dinamikus objektum-típusok kellenek.
    - Pl.: ilyenek a Java-s fájlkezelő streamek is

```
FileInputStream fis = new FileInputStream("/objects.gz");  
BufferedInputStream bis = new BufferedInputStream(fis);  
GzipInputStream gis = new GzipInputStream(bis);  
ObjectInputStream ois = new ObjectInputStream(gis);
```

# Decorator

---

- Szövegmegjelenítő program
- Tetszőleges szöveget akarunk módosítani
- A módosításnak flexibilisnek kell lennie
  - A felhasználó futási időben (dinamikusan) közli a rendszerrel, hogy *nagybetűsíteni szeretné* a szöveget (megnyom egy gombot a user interfacen, amit a kiválasztott szövegre érvényesíteni kell)

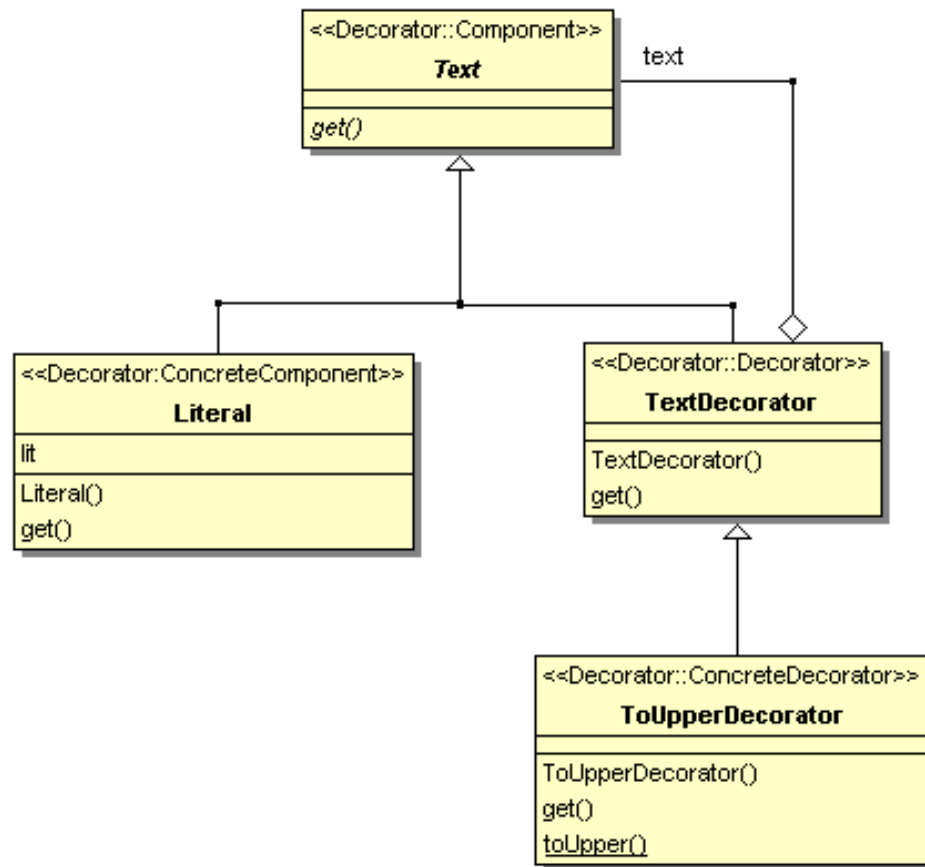
# Decorator példa

---

- Component – **Text**
- ConcreteComponent – **Literal**
- Decorator – **TextDecorator**
- ConcreteDecorator – **ToUpperDecorator**
- Operation – **get()**

# Decorator példa

---



# Decorator példa

Text.java

```
1 package hu.u_szeged.inf.ovrt.decorator;  
2  
3 public interface Text {  
4  
5     public String get();  
6  
7 }
```

```
1 package hu.u_szeged.inf.ovrt.decorator;  
2  
3 public class Literal implements Text {  
4  
5     private String literal;  
6  
7     public Literal(String literal) {  
8         super();  
9         this.literal = literal;  
10    }  
11  
12    @Override  
13    public String get() {  
14        return this.literal;  
15    }  
16  
17 }
```

Literal.java

```
1 package hu.u_szeged.inf.ovrt.decorator;  
2  
3 public abstract class TextDecorator  
4     implements Text {  
5     private Text text;  
6  
7     public TextDecorator(Text text) {  
8         super();  
9         this.text = text;  
10    }  
11  
12    @Override  
13    public String get() {  
14        return this.text.get();  
15    }  
16  
17 }
```

TextDecorator.java

```
1 package hu.u_szeged.inf.ovrt.decorator;
2
3 public class ToUpperDecorator extends TextDecorator {
4
5     public ToUpperDecorator(Text text) {
6         super(text);
7     }
8
9     @Override
10    public String get() {
11        return super.get().toUpperCase();
12    }
13
14 }
```

ToUpperDecorator.java

ReverseDecorator.java

```
1 package hu.u_szeged.inf.ovrt.decorator;
2
3 public class ReverseDecorator extends TextDecorator {
4
5     public ReverseDecorator(Text text) {
6         super(text);
7     }
8
9     @Override
10    public String get() {
11        return new StringBuilder(super.get()).reverse().toString();
12    }
13
14 }
```

```
1 package hu.u_szeged.inf.ovrt.decorator;
2
3 public class Client {
4
5     public static void main(String[] args) {
6         Text upperPalindrome =
7             new ToUpperDecorator(
8                 new ReverseDecorator(
9                     new Literal("Red rum, sir, is murder!")));
10        System.out.println(upperPalindrome.get());
11    }
12
13 }
```

Client.java



!REDRUM SI ,RIS ,MUR DER

- Minden hallgató mondjon a teremben egy további lehetséges konkrét dekorátort az előző példához!
- Minden csapat mondjon a teremben saját ötleten alapuló **Composite** vagy **Decorator** példát! (nem hangzott el órán és nem volt választható projekttema)

# Feladat

---

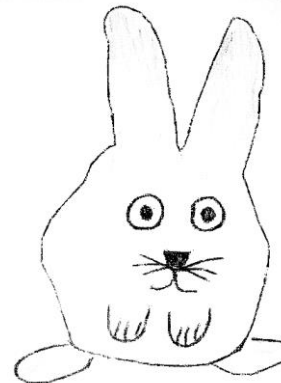
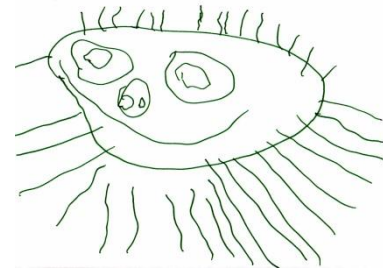


- Készítsünk egy **HTML generátor** alkalmazást, amely különböző objektumokkal dolgozzon, de képes legyen kigenerálni az objektumokból a HTML dokumentumot. A generátor támogassa:
  - Egyszerű tag, aminek van egy törzse
    - `<p>Ez egy bekezdés. </p>`
  - Összetett tag, ami más tag-eket tartalmaz
    - `<div>... <p> ABC </p> ...</div>`
- Építsünk fel egy HTML dokumentumot ezekből a tag-ekből!
- Melyik mintát használjuk? Composite vagy Decorator?

# Feladat

---

- Készítsünk egy **gyermek rajzot leíró programot**. Minden rajzon lehet:
  - üres rajzlap (ez a kiindulási alapja az egésznek, erre rajzolunk)
  - nyuszika
  - fa (akár több fa is lehessen egy rajzon)
  - napsütés (hétágra süt vagy nem)
  - Melyik mintát használjuk? Composite vagy Decorator?



# Feladat