

11β Δομικά πρότυπα σχεδίασης

Τεχνολογία Λογισμικού

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Χειμερινό εξάμηνο 2017-18

Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Δομικά πρότυπα σχεδίασης

Πρότυπα σχετικά με τη δομή και τη συσχέτιση κλάσεων

Περιεχόμενα

1. Adapter (Wrapper)
2. Bridge
3. Composite
4. Decorator
5. Facade
6. Proxy

Adapter (Wrapper)

Σκοπός: Να κάνει μια κλάση συμβατή με κάποιο interface που αναμένει ένας χρήστης της

Μια υπάρχουσα κλάση γίνεται συμβατή με ένα νέο interface

Παράδειγμα

java.lang.Runnable

```
public interface Runnable {  
    public void run()  
}
```

java.util.concurrent.Callable

```
public interface Callable<V> {  
    public V call() throws Exception  
}
```

```
public class RunnableAsCallable implements Callable<Void> {
    private final Runnable runnable;

    public RunnableAsCallable(Runnable runnable) {
        this.runnable = runnable;
    }

    @Override
    public Void call() throws Exception{
        runnable.run();
        return null;
    }
}
```

Συζήτηση

- Μηχανισμός για να κάνουμε δύο ξεχωριστά συστατικά (με διαφορετικό σχεδιασμό) να λειτουργήσουν μαζί, αφού έχουν υλοποιηθεί.
- "Ντύνουμε" ένα interface με τα ρούχα ενός άλλου.

Bridge

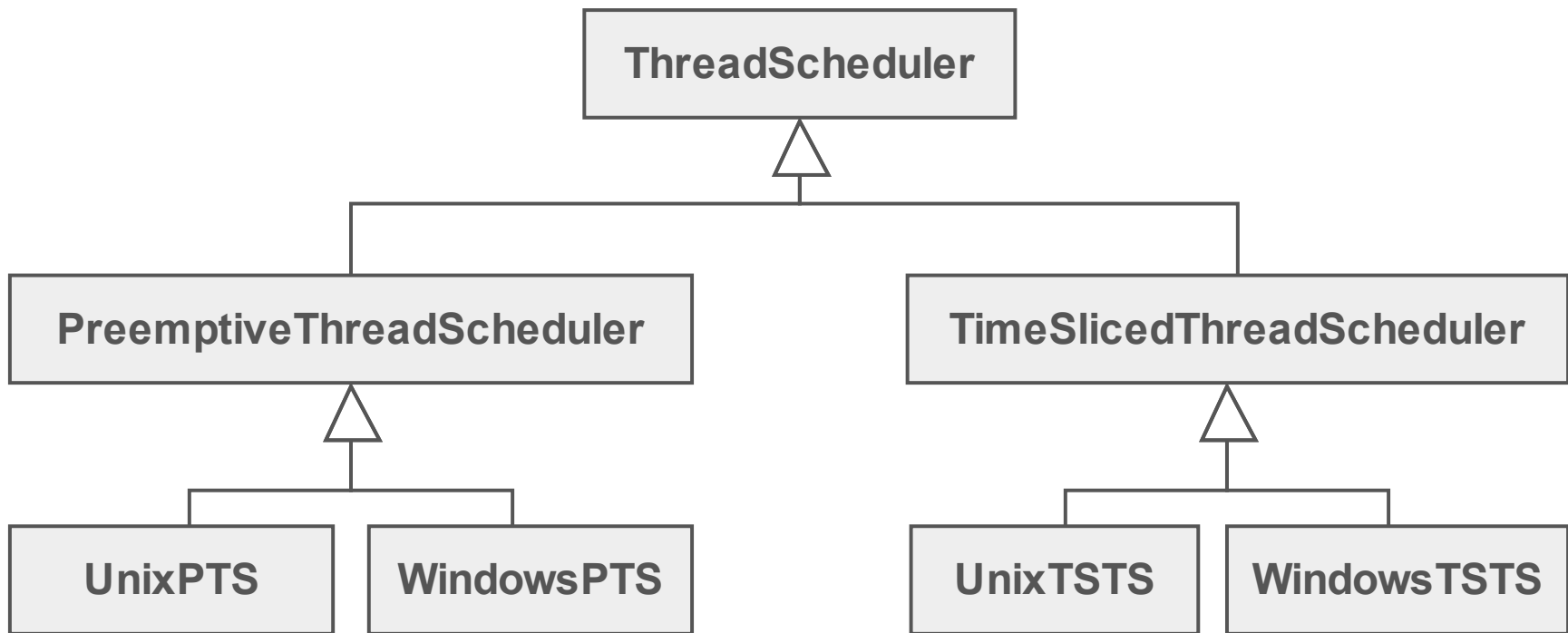
Σκοπός: Να διαχωρίσουμε την αφαίρεση (interface) από την υλοποίηση (class), ώστε να χρησιμοποιούνται ανεξάρτητα.

Παράδειγμα

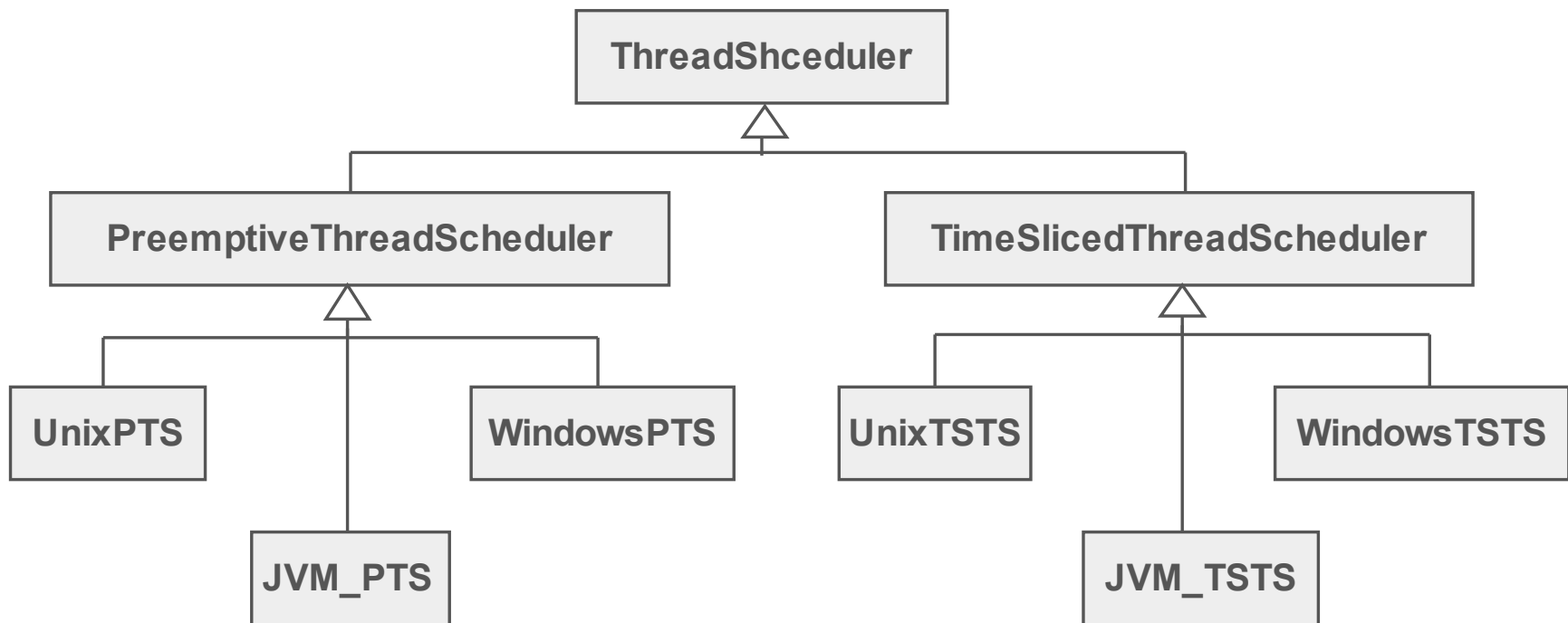
Δύο βασικές έννοιες

- Thread scheduler
- Platform

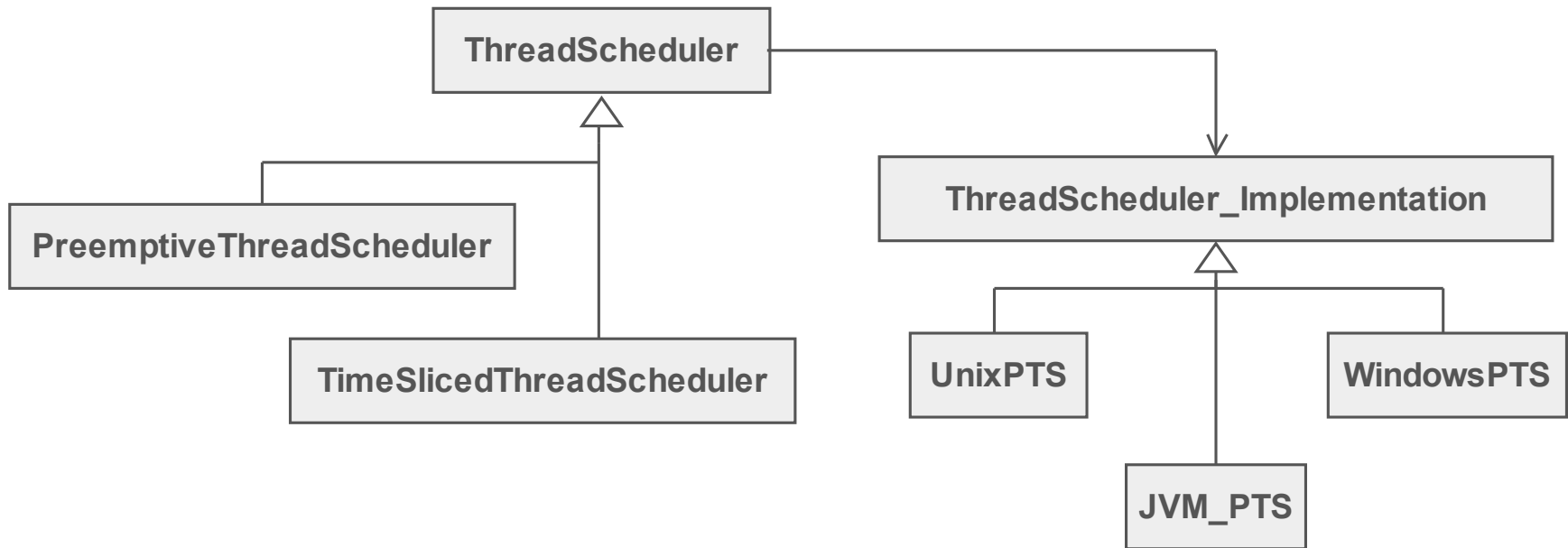
Παράδειγμα (Subclassing)



Παράδειγμα (Subclassing)



Παράδειγμα (Bridge)



Συζήτηση

- Έμφαση στη σύνθεση και όχι στην κληρονομικότητα
- Χρήση δύο ιεραρχιών κληρονομικότητας: μία "δημόσια" και μία "ιδιωτική"
- Οι "δημόσιες" αφαιρέσεις χρησιμοποιούν τις "ιδιωτικές" υλοποιήσεις
- Οι δύο ιεραρχίες μπορούν να εξελιχθούν ανεξάρτητα μεταξύ τους

Composite

Σκοπός: Να επιτρέψουμε σε μια κλάση-χρήστη να χειρίζεται απλά και σύνθετα αντικείμενα με ομοιόμορφο τρόπο

Παράδειγμα

```
interface Resolver {  
    File resolve(String text) throws NotFoundException;  
}  
  
class LocalFileResolver implements Resolver {  
    ...  
}  
  
class URLResolver implements Resolver {  
    ...  
}
```

```
class CompositeResolver implements Resolver {
    private final Resolver[] resolvers;
    CompositeResolver(Resolver... r) {
        this.resolvers = r;
    }
    File resolve(text) throws NotFoundException {
        File f = null;
        for(resolver: resolvers) {
            try {
                f = resolver.resolve(text);
            }
            catch(NotFoundException nfe) {
                ..
            }
            if (f == null) {
                throw new NotFoundException(text);
            }
            else {
                return f;
            }
        }
    }
}
```


Decorator

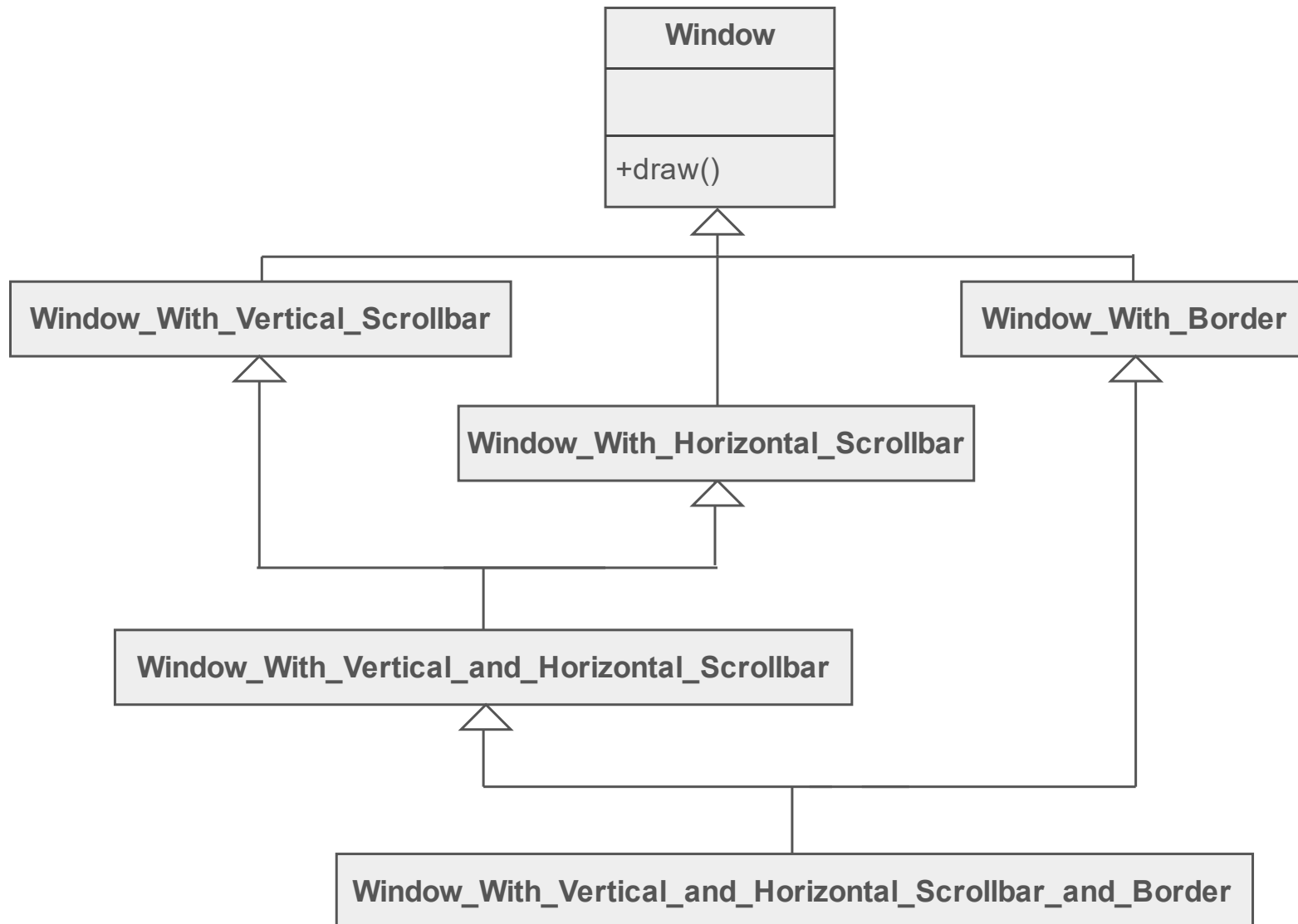
Σκοπός: Να προσθέσουμε επιλεκτικές και προαιρετικές δυνατότητες σε ένα αντικείμενο χωρίς κληρονομικότητα

Παράδειγμα

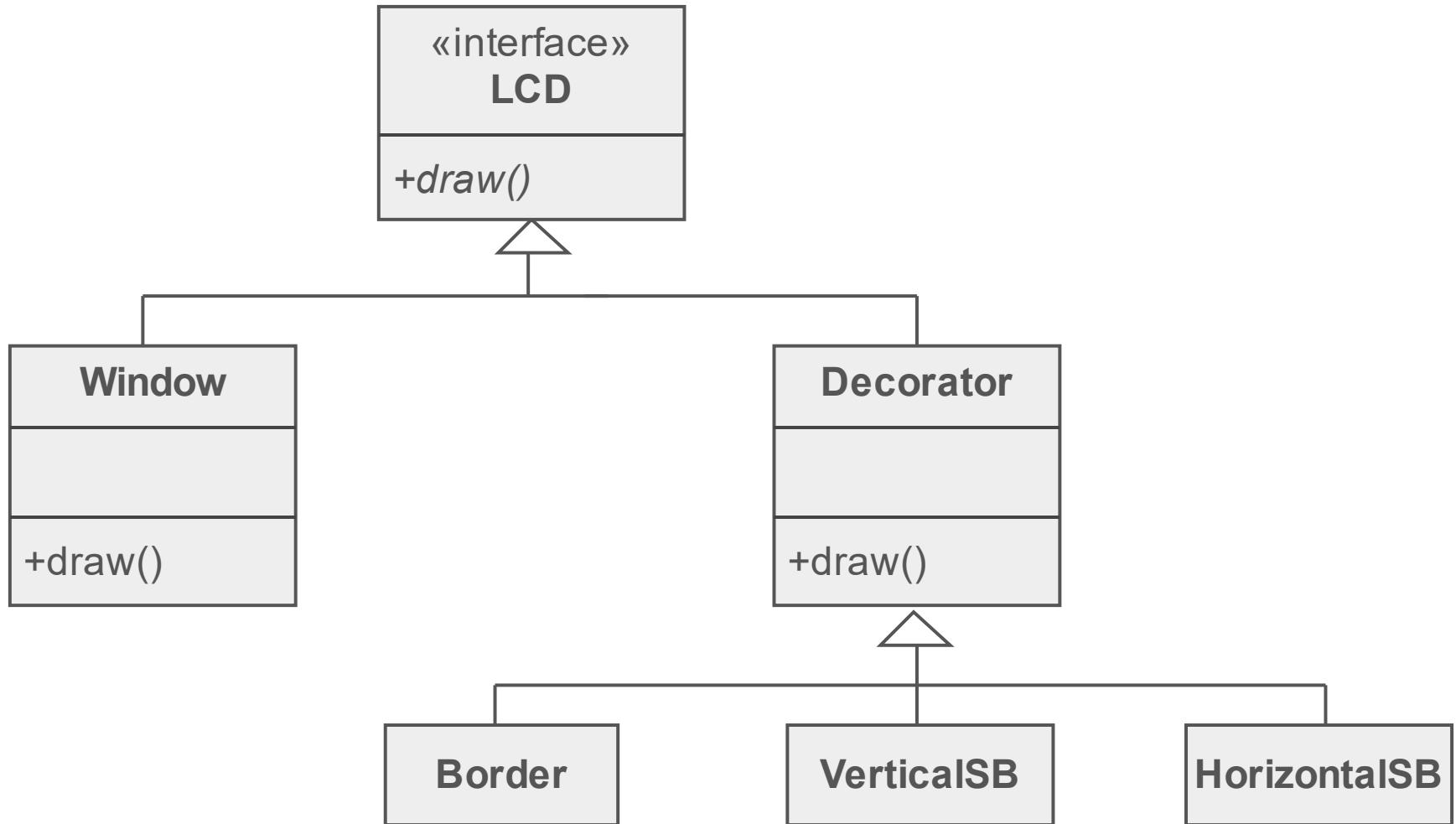
Τέσσερις βασικές έννοιες

- Window
- Horizontal Scrollbar
- Vertical Scrollbar
- Border

Παράδειγμα (Subclassing)



Παράδειγμα (Decorator)



```
Widget w = new BorderDecorator(  
    new HorizontalScrollBarDecorator(  
        new VerticalScrollBarDecorator(  
            new Window("Title")  
        )  
    )  
);  
w.draw();
```

Συζήτηση

- Παρέχει μια πιο ευέλικτη εναλλακτική για την προσθήκη λειτουργικότητας σε μια κλάση σε σχέση με το subclassing (κληρονομικότητα)
- Recursive wrapping

Πραγματικό παράδειγμα (Java IO)

```
OutputStream os = new BufferedOutputStream(  
    new FileOutputStream("/path/to/file")  
);
```

Facade

Σκοπός: Να παρασχεθεί ένα απλό interface σε ένα σύνθετο υποσύστημα.

Παράδειγμα

```
class VideoManagerFacade {  
  
    File convert(String fileName, String format) {  
        VideoFile file = new VideoFile(fileName);  
        Codec srcCodec = CodecFactory.of(file);  
        Codec destCodec;  
        if (format.equals("ogg")) {  
            destCodec = new OggCompressionCodec();  
        } else {  
            destCodec = new MPEG4CompressionCodec();  
        }  
        VideoFile buffer = BitrateReader.read(file, srcCodec);  
        VideoFile tmp = BitrateReader.convert(buffer, destCodec);  
        File result = (new AudioMixer()).fix(tmp);  
        return result;  
    }  
  
}
```

Proxy

Σκοπός: Να προσφέρει έναν ενδιάμεσο για ένα άλλο αντικείμενο, ελέγχοντας την πρόσβαση σε αυτό

Εφαρμογές

- Οκνηρή αρχικοποίηση (virtual proxy)
- Έλεγχος πρόσβασης (access proxy)
- Εκτέλεση απομακρυσμένων υπηρεσιών (remote proxy)
- Χρήση ενδιάμεσης μνήμη (caching)
- Καταγραφή ενεργειών (logging proxy)

Παράδειγμα

```
class Router {
    void setup() {
        route("/index.html", new PublicResource())
        route("/cart.html", new ProtectedResource())
    }

    void service(HttpServletRequest req, HttpServletResponse res) {
        MVCResource res = getResourceFor(req)
        LoggingProxy proxy = new LoggingProxy(res)
        proxy.doService(req, res)
    }
}
```

```

class LoggingProxy extends MVCResource {
    private static final Logger log = ...
    private final MVCResource resource;
    LoggingProxy(MVCResource resource) {
        this.resource = resource;
    }
    User createUser(HttpServletRequest req) {
        return res.createUser
    }
    void doService(HttpServletRequest req, HttpServletResponse res) {
        log.log("Starting service of $resource")
        long t = System.currentTimeMillis()
        try {
            resource.doService(req, res)
        }
        finally {
            t = System.currentTimeMillis() - t
            log.log("Ending service of $resource (duration $t ms)")
        }
    }
}

```