Tervezési minták

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2023. április 21.

Felhasznált irodalom

- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Programtervezési minták: Újrahasznosítható elemek objektumközpontú programokhoz*. Kiskapu, 2004.
- Ilkka Seppälä*. java-design-patterns: Design patterns implemented in Java.* https://java-design-patterns.com/ https://github.com/iluwatar/java-design-patterns

Létrehozási minták (GoF)

- Elvont gyár (*Abstract Factory*)
- Építő (*Builder*)
- Gyártó metódus (*Factory Method*)
- Prototípus (*Prototype*)
- Egyke (*Singleton*)

Elvont gyár (1)

- **Cél**: Kapcsolódó vagy egymástól függő objektumok családjának létrehozására szolgáló felületet biztosít a konkrét osztályok megadása nélkül.
- **Részletek**: https://java-design-patterns.com/patterns/abstract-factory/

Elvont gyár (2)

- **Ismert felhasználások**:
  - `java.sql.Connection`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Connection.html
  - `javax.xml.datatype.DatatypeFactory`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/datatype/DatatypeFactory.html
  - `javax.xml.transform.TransformerFactory`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/transform/TransformerFactory.html
  - `javax.xml.stream.XMLInputFactory`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/stream/XMLInputFactory.html

Építő (1)

- **Cél**: Az összetett objektumok felépítését függetleníti az ábrázolásuktól, így ugyanazzal az építési folyamattal különböző ábrázolásokat hozhatunk létre.
- **Részletek**: https://java-design-patterns.com/patterns/builder/

Építő (2)

- **Ismert felhasználások**:
  - A `java.lang.Appendable` interfész minden implementációja
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Appendable.html
    - Például `java.lang.StringBuilder`
      https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/ProcessBuilder.html
  - `java.lang.ProcessBuilder`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/ProcessBuilder.html
  - `java.time.format.DateTimeFormatterBuilder`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/DateTimeFormatterBuilder.html

Építő (3)

- **Ismert felhasználások**:
  - `java.util.Locale.Builder`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Locale.Builder.html
  - `java.util.StringJoiner`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/StringJoiner.html
  - `java.util.stream.Stream.Builder`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.Builder.html

Építő (4)

- **Ismert felhasználások**:
  - *Apache Commons Lang* (licenc: *Apache License 2.0*)
    https://commons.apache.org/proper/commons-lang/
    https://github.com/apache/commons-lang
    - `org.apache.commons.lang3.builder.EqualsBuilder`
      https://javadoc.io/doc/org.apache.commons/commons-lang3/latest/org/apache/commons/lang3/builder/EqualsBuilder.html
    - `org.apache.commons.lang3.builder.HashCodeBuilder`
      https://javadoc.io/doc/org.apache.commons/commons-lang3/latest/org/apache/commons/lang3/builder/HashCodeBuilder.html
    - `org.apache.commons.lang3.builder.ToStringBuilder`
      https://javadoc.io/doc/org.apache.commons/commons-lang3/latest/org/apache/commons/lang3/builder/ToStringBuilder.html

Építő (5)

- **Ismert felhasználások**:
  - *Project Lombok* (licenc: *MIT License*)
    https://projectlombok.org/
    https://github.com/projectlombok/lombok
    - `@lombok.Builder`
      https://projectlombok.org/features/Builder
      https://projectlombok.org/api/lombok/Builder.html

Építő (6)

- **Példakód**: `java.util.StringJoiner`

```
var s = new StringJoiner(",", "[", "]")
    .add("John")
    .add("Paul")
    .add("George")
    .add("Ringo")
    .toString(); // "[John,Paul,George,Ringo]"
```

Építő (7)

- **Példakód**: Lombok

```
import java.math.BigDecimal;
import java.net.URL;
import java.time.Year;
import java.util.List;

@lombok.NoArgsConstructor
@lombok.AllArgsConstructor
@lombok.Builder
@lombok.Data
public class Movie {

  private String title;
  private Year year;
  private BigDecimal rating;
  private int votes;
  @lombok.Singular private List<String> genres;
  private URL url;

}
```

Építő (8)

- **Példakód**: Lombok (folytatás)

```
var movie = Movie.builder()
    .title("Shrek")
    .year(Year.of(2001))
    .rating(new BigDecimal("7.8"))
    .votes(587180)
    .url(new URL("https://www.imdb.com/title/tt0126029/"))
    .genre("animation")
    .genre("adventure")
    .genre("comedy")
    .build();
```

Gyártófüggvény (1)

- **Cél**:
  - Felületet határoz meg egy objektum létrehozásához, az alosztályokra bízva, melyik osztályt példányosítják.
  - A gyártófüggvények megengedik az osztályoknak, hogy a példányosítást az alosztályokra ruházzák át.
- **Részletek**: https://java-design-patterns.com/patterns/factory-method/

Gyártófüggvény (2)

- **Ismert felhasználások**:
  - `javax.xml.parsers.DocumentBuilderFactory#newDocumentBuilder()`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/parsers/DocumentBuilderFactory.html#newDocumentBuilder()
  - `javax.xml.parsers.SAXParserFactory#newSAXParser()`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/parsers/SAXParserFactory.html#newSAXParser()
  - `javax.xml.xpath.XpathFactory#newXPath()`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/javax/xml/xpath/XPathFactory.html#newXPath()
  - `java.util.ResourceBundle`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ResourceBundle.html

Prototípus (1)

- **Cél**: Prototípus példány használatával határozza meg, hogy milyen típusú objektumokat kell létrehozni, az új objektumokat pedig ennek a prototípusnak a lemásolásával állítja elő.
- **Részletek**: https://java-design-patterns.com/patterns/prototype/

Prototípus (2)

- **Ismert felhasználások**:
  - `java.lang.Cloneable`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Cloneable.html

Egyke (1)

- **Cél**: Egy osztályból csak egy példányt engedélyez, és ehhez globális hozzáférési pontot ad meg.
- **Részletek**: https://java-design-patterns.com/patterns/singleton/

Egyke (2)

- **Ismert felhasználások**:
  - `java.lang.Runtime`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Runtime.html
  - `java.time.chrono.IsoChronology`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/chrono/IsoChronology.html

További létrehozási minták

- Függőség befecskendezés (Dependency Injection) https://java-design-patterns.com/patterns/dependency-injection/
- Többke (*Multiton*) https://java-design-patterns.com/patterns/multiton/
- Objektumkészlet (*Object Pool*) https://java-design-patterns.com/patterns/object-pool/
- Érték objektum (*Value Object*) https://java-design-patterns.com/patterns/value-object/
- …

Többke (1)

- **Cél**: Egy osztályból csak adott számú (egynél több) példányt engedélyez, és ezekhez globális hozzáférési pontot ad meg.
- **Részletek**: https://java-design-patterns.com/patterns/multiton/

Többke (2)

- **Ismert felhasználások**:
  - `java.lang.Thread.State`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Thread.State.html
  - `java.lang.annotation.RetentionPolicy`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/annotation/RetentionPolicy.html
  - `java.nio.file.AccessMode`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/nio/file/AccessMode.html
  - `java.time.DayOfWeek`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/DayOfWeek.html

Objektumkészlet (1)

- **Cél**: Inicializált objektumok egy halmazát tartja nyilván az igények kiszolgálásához, ahelyett, hogy létrehozná és megsemmisítené az objektumokat.
- **Részletek**: https://java-design-patterns.com/patterns/object-pool/

Objektumkészlet (2)

- **Ismert felhasználások**:
  - *Apache Commons Pool* (licenc: *Apache License 2.0*)
    https://commons.apache.org/proper/commons-pool/
    https://github.com/apache/commons-pool
  - *Vibur Object Pool* (licenc: *Apache License 2.0*)
    https://www.vibur.org/vibur-object-pool/
    https://github.com/vibur/vibur-object-pool
  - Adatbázis kapcsolatok gyorsítótárazása (*connection pooling*):
    - *Apache Commons DBCP* (licenc: *Apache License 2.0*)
      https://commons.apache.org/proper/commons-dbcp/
    - *HikariCP* (licenc: *Apache License 2.0*) https://github.com/brettwooldridge/HikariCP
    - *Vibur DBCP* (licenc: *Apache License 2.0*) https://www.vibur.org/
      https://github.com/vibur/vibur-dbcp
    - …

Szerkezeti minták (GoF)

- Illesztő (*Adapter*)
- Híd (*Bridge*)
- Összetétel (*Composite*)
- Díszítő (*Decorator*)
- Homlokzat (*Facade*)
- Pehelysúlyú (*Flyweight*)
- Helyettes (*Proxy*)

Illesztő (1)

- **Cél**:
  - Az adott osztály interfészét az ügyfelek által igényelt interfésszé alakítja.
  - E módszerrel az egyébként összeférhetetlen interfészű osztályok együttműködését biztosíthatjuk.
- **Részletek**: https://java-design-patterns.com/patterns/adapter/

Illesztő (2)

- **Ismert felhasználások**:
  - `java.io.InputStreamReader`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/InputStreamReader.html
  - `java.io.OutputStreamWriter`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/OutputStreamWriter.html
  - `jakarta.xml.bind.annotation.adapters.XmlAdapter`
    https://jakarta.ee/specifications/platform/10/apidocs/jakarta/xml/bind/annotation/adapters/xmladapter

Illesztő (3)

- **Példakód**: `java.io.InputStreamReader` (OpenJDK 17)

```
public class InputStreamReader extends Reader {

  private final StreamDecoder sd;

  public InputStreamReader(InputStream in) {
    super(in);
    sd = StreamDecoder.forInputStreamReader(in, this,
        Charset.defaultCharset()); // ## check lock object
  }
  // ...
```

Híd

- **Cél**: Az elvont ábrázolást elválasztja a megvalósítástól, hogy a kettő egymástól függetlenül módosítható legyen.
- **Részletek**: https://java-design-patterns.com/patterns/bridge/

Összetétel (1)

- **Cél**:
  - Az objektumokat faszerkezetbe rendezi, hogy ábrázolhassuk a rész-egész viszonyokat.
  - A módszer révén az önálló objektumokat és az objektum-összetételeket egységesen kezelhetjük.
- **Részletek**: https://java-design-patterns.com/patterns/composite/

Összetétel (2)

- **Ismert felhasználások**:
  - `javafx.scene.Node`
    https://openjfx.io/javadoc/20/javafx.graphics/javafx/scene/Node.html
  - *jsoup* (licenc: *MIT License*) https://jsoup.org/
    https://github.com/jhy/jsoup
    - `org.jsoup.nodes.Node`
      https://jsoup.org/apidocs/org/jsoup/nodes/Node.html

Díszítő (1)

- **Cél**:
  - Az objektumokhoz dinamikusan további felelősségi köröket rendel.
  - A kiegészítő szolgáltatások biztosítása terén e módszer rugalmas alternatívája az alosztályok létrehozásának.
- **Részletek**: https://java-design-patterns.com/patterns/decorator/

Díszítő (2)

- **Ismert felhasználások**:
  - `java.io.InputStream`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/InputStream.html
    - Lásd azon alosztályait, melyek konstruktora `InputStream` objektumot fogad paraméterként, mint például a `java.io.ObjectInputStream`.
  - `java.io.OutputStream`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/OutputStream.html
    - Lásd azon alosztályait, melyek konstruktora `OutputStream` objektumot fogad paraméterként, mint például a `java.io.ObjectOutputStream`.
  - `java.util.Collections#unmodifiable`*`XXX`*`()`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html

Homlokzat

- **Cél**:
  - Egy alrendszerben interfészek egy halmazához egységes interfészt biztosít.
  - A módszerrel magasabb szintű interfészt határozunk meg, amelynek révén az adott alrendszer könnyebben használhatóvá válik.
- **Részletek**: https://java-design-patterns.com/patterns/facade/

Pehelysúlyú (1)

- **Cél**: Megosztás révén támogatja a nagy finomságú objektumok tömegeinek hatékony felhasználását.
- **Részletek**: https://java-design-patterns.com/patterns/flyweight/

Pehelysúlyú (2)

- **Ismert felhasználások**:
  - `java.lang.Byte#valueOf(byte b)`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Byte.html#valueOf(byte)
  - `java.lang.Character#valueOf(char c)`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Character.html#valueOf(char)
  - `java.lang.Integer#valueOf(int i)`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Integer.html#valueOf(int)

Helyettes (1)

- **Cél**: Egy adott objektumot egy helyettesítő objektummal váltunk fel, amely szabályozza az eredeti objektumhoz történő hozzáférést is.
- **Részletek**: https://java-design-patterns.com/patterns/proxy/

Helyettes (2)

- **Ismert felhasználások**:
  - `java.lang.reflect.Proxy` https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/reflect/Proxy.html
  - *Apache Commons Proxy* (licenc: *Apache License 2.0*) https://commons.apache.org/proper/commons-proxy/ https://github.com/apache/commons-proxy
  - *EasyMock* (licenc: *Apache License 2.0*) https://easymock.org/ https://github.com/easymock/easymock
  - *Mockito* (licenc: *MIT License*) https://site.mockito.org/ https://github.com/mockito/mockito

További szerkezeti minták

- Absztrakt dokumentum (*Abstract Document*)
  https://java-design-patterns.com/patterns/abstract-document/
- Iker (*Twin*)
  https://java-design-patterns.com/patterns/twin/
- …

Viselkedési minták (GoF)

- Felelősséglánc (*Chain of Responsibility*)
- Parancs (*Command*)
- Értelmező (*Interpreter*)
- Bejáró (*Iterator*)
- Közvetítő (*Mediator*)
- Emlékeztető (*Memento*)
- Megfigyelő (*Observer*)
- Állapot (*State*)
- Stratégia (*Strategy*)
- Sablonfüggvény (*Template Method*)
- Látogató (*Visitor*)

Felelősséglánc (1)

- **Cél**:
  - A minta arra szolgál, hogy elkerüljük a kérelem küldőjének a fogadóhoz való kötését.
  - Ezt úgy érjük el, hogy több objektumnak is jogot adunk a kérelem kezelésére.
  - A fogadó objektumokat láncba állítjuk, amelyen a kérelem addig halad, amíg el nem ér egy objektumot, ami képes a kezelésére.
- **Részletek**: https://java-design-patterns.com/patterns/chain-of-responsibility/

Felelősséglánc (2)

- **Ismert felhasználások**:
  - `java.util.logging.Logger`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.logging/java/util/logging/Logger.html
  - *Apache Log4j 2* (licenc: *Apache License 2.0*)
    https://logging.apache.org/log4j/2.x/
    https://github.com/apache/logging-log4j2
    - Lásd:
      https://logging.apache.org/log4j/2.x/manual/architecture.html
  - *Apache Commons Chain* (licenc: *Apache License 2.0*)
    https://commons.apache.org/proper/commons-chain/
    https://github.com/apache/commons-chain

Parancs (1)

- **Cél**: A kérelmeket objektumokba zárjuk, aminek célja, hogy az ügyfeleknek paraméterként különböző kérelmeket adjunk át, ezeket sorba állítsuk vagy naplózzuk, illetve támogassuk a műveletek visszavonását.
- **Részletek**: https://java-design-patterns.com/patterns/command/

Parancs (2)

- **Ismert felhasználások**:
  - `java.lang.Runnable`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Runnable.html
  - `java.util.concurrent.Callable`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/Callable.html

Értelmező (1)

- **Cél**: Egy adott nyelv nyelvtanát ábrázoljuk, illetve ehhez az ábrázoláshoz értelmezőt biztosítunk, amely annak alapján képes a nyelv mondatait megérteni.
- **Részletek**: https://java-design-patterns.com/patterns/interpreter/

Értelmező (2)

- **Ismert felhasználások**:
  - A `java.text.Format` osztály alosztályai
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/text/Format.html
  - `java.time.format.DateTimeFormatter`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/DateTimeFormatter.html
  - `java.util.regex.Pattern`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/regex/Pattern.html

Bejáró (1)

- **Cél**: Az összetett objektumok elemeinek soros elérését a háttérben megbúvó ábrázolás felfedése nélkül biztosító módszer kialakítása.
- **Részletek**: https://java-design-patterns.com/patterns/iterator/

Bejáró (2)

- **Ismert felhasználások**:
  - `java.sql.ResultSet`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/ResultSet.html
  - `java.util.Enumeration`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Enumeration.html
  - `java.util.Iterator`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Iterator.html

Közvetítő (1)

- **Cél**:
  - A cél meghatározni egy objektumot, amely objektumok egy halmazának együttműködését irányítja. (Vagyis ezeket egyetlen objektumba tokozzuk be.)
  - A módszerrel laza csatolást hozunk létre, amelyben az egyes objektumok közvetlenül nem hivatkozhatnak egymásra, a köztük lévő kapcsolatok pedig egymástól függetlenül módosíthatók.
- **Részletek**: https://java-design-patterns.com/patterns/mediator/

Közvetítő (2)

- **Ismert felhasználások**:
  - `java.util.concurrent.ExecutorService`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/ExecutorService.html
  - `java.util.Timer`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Timer.html

Emlékeztető (1)

- **Cél**: Az egységbe zárás megsértése nélkül rögzíteni és felfedni egy objektum belső állapotát, hogy az később ebbe az állapotba visszaállítható legyen.
- **Részletek**: https://java-design-patterns.com/patterns/memento/

Emlékeztető (2)

- **Ismert felhasználások**:
  - `java.util.Date`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Date.html
    - Lásd a `getTime()` és `setTime(long time)` metódusokat.

Megfigyelő (1)

- **Cél**: Objektumok között egy sok-sok függőségi kapcsolatot létrehozni, így amikor az egyik objektum állapota megváltozik, minden tőle függő objektum értesül erről és automatikusan frissül.
- **Részletek**: https://java-design-patterns.com/patterns/observer/

Megfigyelő (2)

- **Ismert felhasználások**:
  - `java.util.EventListener`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/EventListener.html
  - `javafx.beans.Observable`
    https://openjfx.io/javadoc/20/javafx.base/javafx/beans/Observable.html
    - Lásd a `javafx.beans` és `javafx.beans.property` csomagokat.
  - `java.util.concurrent.Flow`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/Flow.html

Állapot

- **Cél**:
  - Egy adott objektum számára engedélyezni, hogy belső állapotának megváltozásával megváltoztathassa viselkedését is.
  - Az objektum ekkor látszólag módosítja az osztályát.
- **Részletek**: https://java-design-patterns.com/patterns/state/

Stratégia (1)

- **Cél**:
  - Algoritmus-család meghatározása, melyben az algoritmusokat egyenként egységbe zárjuk és egymással felcserélhetővé tesszük.
  - E módszer révén az algoritmus az ügyféltől függetlenül módosítható.
- **Részletek**: https://java-design-patterns.com/patterns/strategy/

Stratégia (2)

- **Ismert felhasználások**:
  - `java.util.Collections`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html
    - Lásd a `binarySearch()`, `max()`, `min()` és `sort()` metódusokat.
    - Kapcsolódó interfészek:
      - `java.lang.Comparable`
        https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Comparable.html
      - `java.util.Comparator`
        https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Comparator.html

Sablonfüggvény (1)

- **Cél**:
  - Egy adott művelet algoritmusának vázát elkészíteni, amelynek egyes lépéseit alosztályokra ruházzuk át.
  - Így az alosztályok az algoritmus egyes lépéseit felülbírálhatják, anélkül, hogy az algoritmus szerkezete módosulna.
- **Részletek**: https://java-design-patterns.com/patterns/template-method/

Sablonfüggvény (2)

- **Ismert felhasználások**:
  - `java.io.InputStream`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/InputStream.html
  - `java.io.OutputStream`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/OutputStream.html
  - `java.util.AbstractList`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/AbstractList.html
  - `java.util.AbstractMap`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/AbstractMap.html
  - `java.util.AbstractQueue`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/AbstractQueue.html

Látogató (1)

- **Cél**: Egy objektumszerkezet elemein végrehajtandó műveletet ábrázolni: a Látogató minta segítségével anélkül határozhatunk meg egy új műveletet, hogy a benne részt vevő elemek osztályát meg kellene változtatnunk.
- **Részletek**: https://java-design-patterns.com/patterns/visitor/

Látogató (2)

- **Ismert felhasználások**:
  - `java.nio.file.FileVisitor`
    https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/nio/file/FileVisitor.html
  - *jsoup* (licenc: *MIT License*) https://jsoup.org/
    https://github.com/jhy/jsoup
    - `org.jsoup.select.NodeVisitor`
      https://jsoup.org/apidocs/org/jsoup/select/NodeVisitor.html

Látogató (3)

- **Példakód**: `java.nio.file.FileVisitor`

```
import java.nio.file.Files;
import java.nio.file.FileSystems;
import java.nio.file.FileVisitResult;
import java.nio.file.Path;
import java.nio.file.SimpleFileVisitor;
import java.nio.file.attribute.BasicFileAttributes;

var pathMatcher = FileSystems.getDefault().getPathMatcher("glob:**.class");

var fileVisitor = new SimpleFileVisitor<Path>() {
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) {
        if (pathMatcher.matches(file)) {
            System.out.println(file);
        }
        return FileVisitResult.CONTINUE;
    }
};

Files.walkFileTree(Path.of("."), fileVisitor);
```

További viselkedési minták

- Null objektum (*Null Object*)
  https://java-design-patterns.com/patterns/null-object/
- …

Null objektum

- **Cél**: A `null` referencia alternatíváját biztosítja egy objektum hiányának jelzésére egy olyan objektum révén, mely az elvárt interfészt üres metódustörzsekkel implementálja.
- **Részletek**: https://java-design-patterns.com/patterns/null-object/