Tervezési minták

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

Utolsó módosítás: 2018. március 9.

Ábrák

- Az ábrák az *ObjectAid UML Explorer* szoftverrel készültek. http://www.objectaid.com/

Felhasznált irodalom

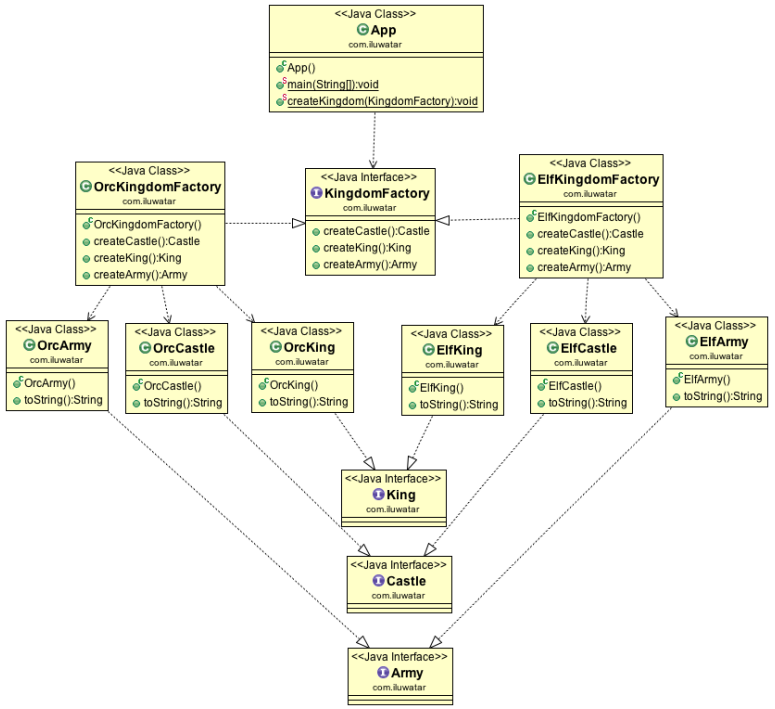
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Programtervezési minták: Újrahasznosítható elemek objektumközpontú programokhoz*. Kiskapu, 2004.
- *java-design-patterns: Design patterns implemented in Java* https://github.com/iluwatar/java-design-patterns

Létrehozási minták (GoF)

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

Elvont gyár (1)

- **Cél**: Kapcsolódó vagy egymástól függő objektumok családjának létrehozására szolgáló felületet biztosít a konkrét osztályok megadása nélkül.



Elvont gyár (2)

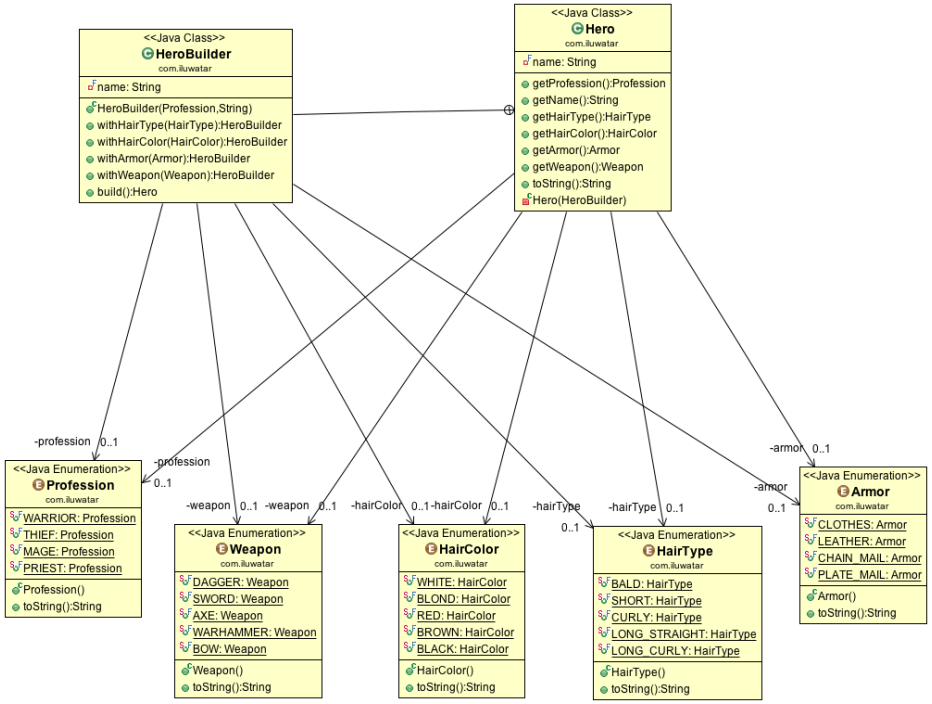
- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/abstract-factory

Elvont gyár (4)

- **Ismert felhasználások**:
  - `java.sql.Connection` https://docs.oracle.com/javase/9/docs/api/java/sql/Connection.html
  - `javax.xml.datatype.DatatypeFactory` https://docs.oracle.com/javase/9/docs/api/javax/xml/datatype/DatatypeFactory.html
  - `javax.xml.transform.TransformerFactory` https://docs.oracle.com/javase/9/docs/api/javax/xml/transform/TransformerFactory.html
  - `javax.xml.stream.XMLInputFactory` https://docs.oracle.com/javase/9/docs/api/javax/xml/stream/XMLInputFactory.html

Építő (1)

- **Cél**: Az összetett objektumok felépítését függetleníti az ábrázolásuktól, így ugyanazzal az építési folyamattal különböző ábrázolásokat hozhatunk létre.



Építő (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/builder

Építő (4)

- **Ismert felhasználások**:
  - `java.lang.StringBuilder` http://docs.oracle.com/javase/9/docs/api/java/lang/StringBuilder.html
  - `java.time.format.DateTimeFormatterBuilder` https://docs.oracle.com/javase/9/docs/api/java/time/format/DateTimeFormatterBuilder.html
  - `java.util.Locale.Builder` https://docs.oracle.com/javase/9/docs/api/java/util/Locale.Builder.html
  - `java.lang.ProcessBuilder` https://docs.oracle.com/javase/9/docs/api/java/lang/ProcessBuilder.html

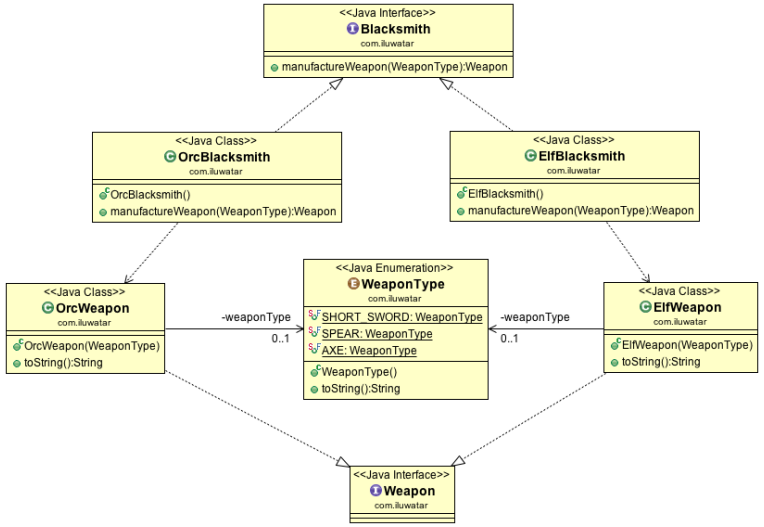
Gyártófüggvény (1)

- **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.

Gyártófüggvény (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/factory-method

Gyártófüggvény (3)



Gyártófüggvény (4)

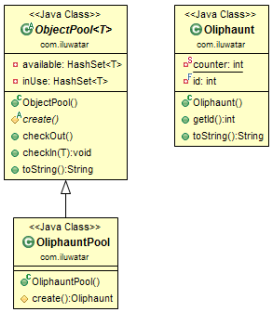
- **Ismert felhasználások**:
  - `javax.xml.parsers.DocumentBuilderFactory#newDocumentBuilder()` https://docs.oracle.com/javase/9/docs/api/javax/xml/parsers/DocumentBuilderFactory.html#newDocumentBuilder--
  - `javax.xml.parsers.SAXParserFactory#newSAXParser()` https://docs.oracle.com/javase/9/docs/api/javax/xml/parsers/SAXParserFactory.html#newSAXParser--
  - `java.util.ResourceBundle` https://docs.oracle.com/javase/9/docs/api/java/util/ResourceBundle.html#getBundle-java.lang.String-

Objektumkészlet (1)

- **Cél**: Inicializált objektumok egy halmazát tartja nyilván az igények kiszolgálásához, ahelyett, hogy létrehozná és megsemmisítené az objektumokat.

Objektumkészlet (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/object-pool



Objektumkészlet (3)

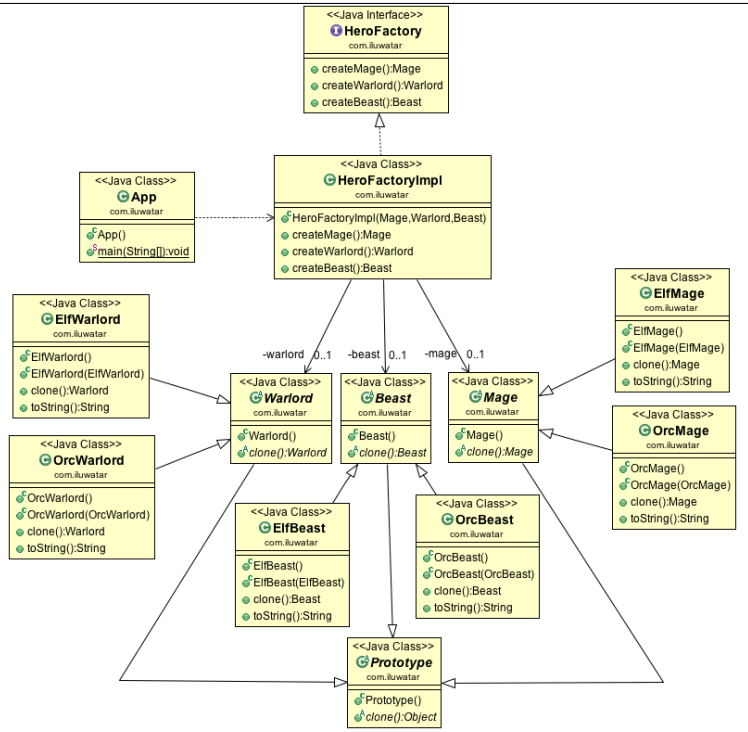
- **Ismert felhasználások**:
  - Adatbázis kapcsolatok gyorsítótárazása (*connection pooling*)
    - Például:
      - *Apache Commons DBCP* https://commons.apache.org/proper/commons-dbcp/
      - *HikariCP* https://brettwooldridge.github.io/HikariCP/
      - *Vibur DBCP* http://www.vibur.org/

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.

Prototípus (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/prototype



Prototípus (4)

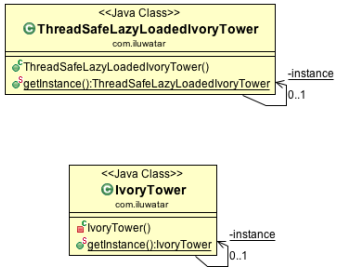
- **Ismert felhasználások**:
  - `java.lang.Cloneable` https://docs.oracle.com/javase/9/docs/api/java/lang/Cloneable.html

Egyke (1)

- **Cél**: Egy osztályból csak egy példányt engedélyez, és ehhez globális hozzáférési pontot ad meg.

Egyke (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/singleton



Egyke (3)

- **Ismert felhasználások**:
  - `java.lang.Runtime` https://docs.oracle.com/javase/9/docs/api/java/lang/Runtime.html
  - `java.time.chrono.IsoChronology` https://docs.oracle.com/javase/9/docs/api/java/time/chrono/IsoChronology.html

További létrehozási minták

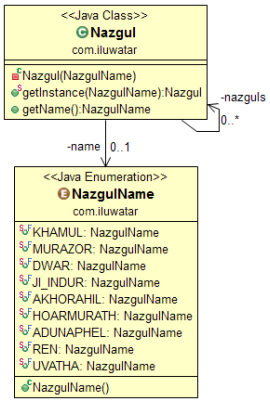
- Gyár készlet (*Factory Kit*) https://github.com/iluwatar/java-design-patterns/tree/master/factory-kit
- Modul (*Module*) https://github.com/iluwatar/java-design-patterns/tree/master/module
- Egyállapotú (*Monostate*) https://github.com/iluwatar/java-design-patterns/tree/master/monostate
- Többke (*Multiton*) https://github.com/iluwatar/java-design-patterns/tree/master/multiton
- Objektum létrehozó (*Object Mother*) https://github.com/iluwatar/java-design-patterns/tree/master/object-mother
- Lépés építő (*Step Builder*) https://github.com/iluwatar/java-design-patterns/tree/master/step-builder
- Érték objektum (*Value Object*) https://github.com/iluwatar/java-design-patterns/tree/master/value-object
- …

Többke (1)

- **Cél**: Egy osztályból csak adott számú (egynél több) példányt engedélyez, és ezekhez globális hozzáférési pontot ad meg.

Többke (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/multiton



Többke (3)

- **Ismert felhasználások**:
  - `java.lang.Thread.State` https://docs.oracle.com/javase/9/docs/api/java/lang/Thread.State.html
  - `java.lang.annotation.RetentionPolicy` https://docs.oracle.com/javase/9/docs/api/java/lang/annotation/RetentionPolicy.html
  - `java.math.RoundingMode` http://docs.oracle.com/javase/9/docs/api/java/math/RoundingMode.html
  - `java.time.DayOfWeek` https://docs.oracle.com/javase/9/docs/api/java/time/DayOfWeek.html

Szerkezeti minták (GoF)

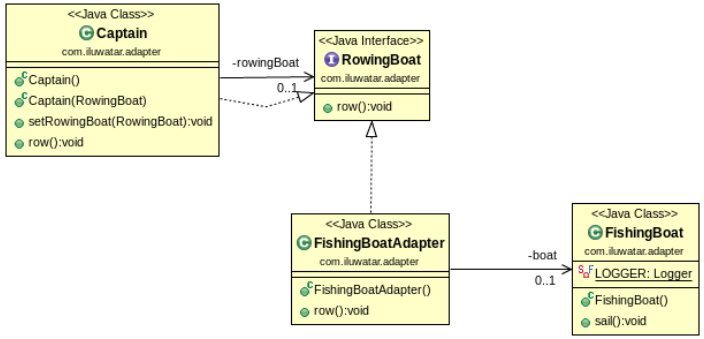
- Illesztő (*Adapter*)
- Híd (*Bridge*)
- Összetétel (*Composite*)
- Díszítő (*Decorator*)
- Homlokzat (*Facade*)
- Pehelysúlyú (*Flyweight*)
- Helyettes (*Proxy*)

Illesztő (1)

- **Cél**:
  - Az adott osztály interfészét az ügyfelek által igényelt interfésszé alakítja.
  - E módszerrel az egyébként összeférhetetlen interfészű osztályok együttműködését biztosíthatjuk.

Illesztő (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/adapter



Illesztő (3)

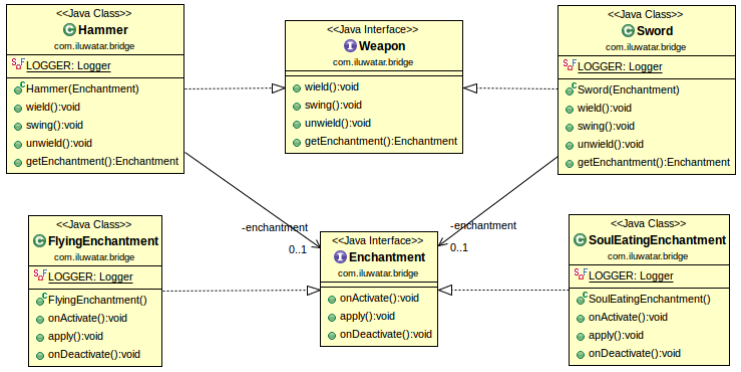
- **Ismert felhasználások**:
  - `java.io.InputStreamReader` http://docs.oracle.com/javase/9/docs/api/java/io/InputStreamReader.html
  - `java.io.OutputStreamWriter` http://docs.oracle.com/javase/9/docs/api/java/io/OutputStreamWriter.html
  - `javax.xml.bind.annotation.adapters.XmlAdapter` https://docs.oracle.com/javase/9/docs/api/javax/xml/bind/annotation/adapters/XmlAdapter.html
  - `org.xml.sax.helpers.ParserAdapter` https://docs.oracle.com/javase/9/docs/api/org/xml/sax/helpers/ParserAdapter.html
  - `org.xml.sax.helpers.XMLReaderAdapter` https://docs.oracle.com/javase/9/docs/api/org/xml/sax/helpers/XMLReaderAdapter.html

Híd (1)

- **Cél**: Az elvont ábrázolást elválasztja a megvalósítástól, hogy a kettő egymástól függetlenül módosítható legyen.

Híd (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/bridge



Híd (3)

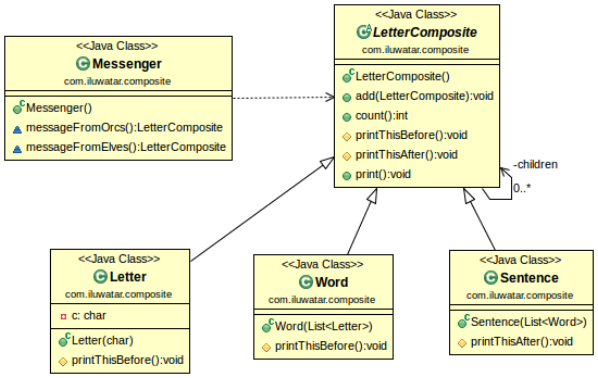
- **Ismert felhasználások**:
  - TODO

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.

Összetétel (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/composite



Összetétel (3)

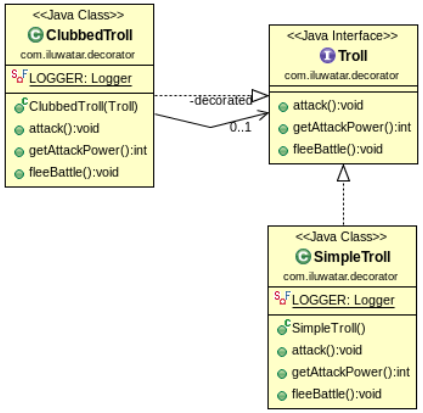
- **Ismert felhasználások**:
  - `javafx.scene.Node` https://docs.oracle.com/javase/9/docs/api/javafx/scene/Node.html
  - *jsoup: Java HTML Parser* https://jsoup.org/
    - `org.jsoup.nodes.Node` https://jsoup.org/apidocs/org/jsoup/nodes/Node.html

Díszítő (1)

- **Cél**:
  - Az objektumokhoz dinamikusan további felelősségi köröket rendel.
  - A kiegészítő szolgáltatások biztosítása terén e módszer rugalmas alternatívája az alosztályok létrehozásának.

Díszítő (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/decorator



Díszítő (3)

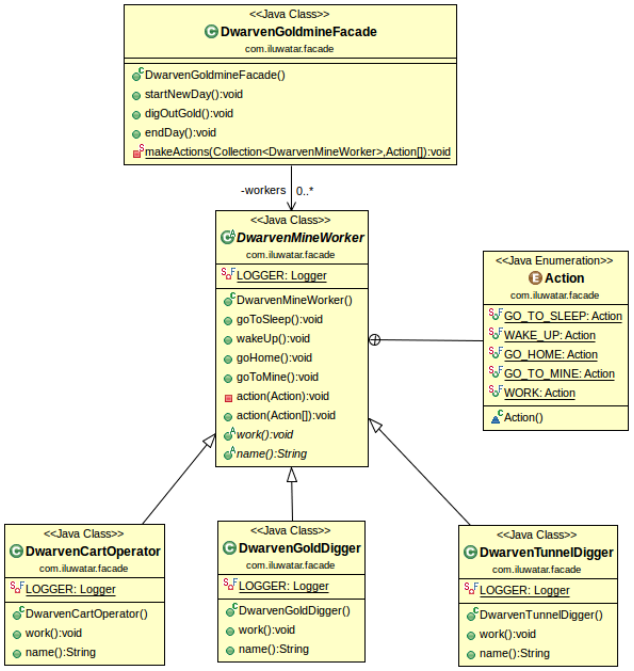
- **Ismert felhasználások**:
  - `java.io.InputStream` http://docs.oracle.com/javase/9/docs/api/java/io/InputStream.html
    - Lásd azon alosztályait, melyek konstruktora `InputStream` objektumot fogad paraméterként, mint például a `java.io.ObjectInputStream`.
  - `java.io.OutputStream` http://docs.oracle.com/javase/9/docs/api/java/io/OutputStream.html
    - Lásd azon alosztályait, melyek konstruktora `OutputStream` objektumot fogad paraméterként, mint például a `java.io.ObjectOutputStream`.
  - `java.util.Collections#unmodifiable`*`XXX`*`()` https://docs.oracle.com/javase/9/docs/api/java/util/Collections.html

Homlokzat (1)

- **Cél**:
  - Egy alrendszerben interfészek egy halmazához egységes interfészt biztosít.
  - A módszerrel magasabb szintű interfészt határozunk meg, amelynek révén az adott alrendszer könnyebben használhatóvá válik.

Homlokzat (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/facade



Homlokzat (4)

- **Ismert felhasználások**:
  - `java.lang.Class` https://docs.oracle.com/javase/9/docs/api/java/lang/Class.html

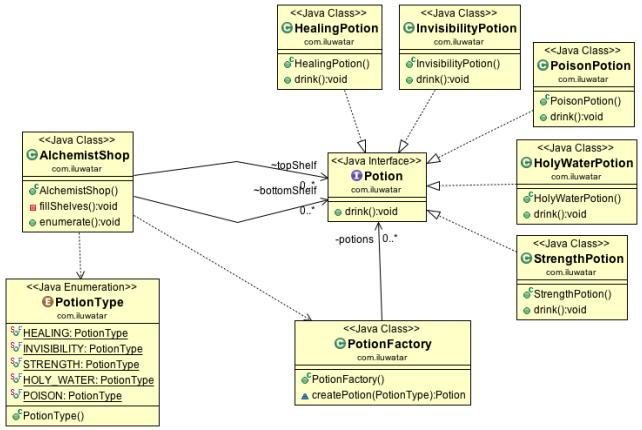
Pehelysúlyú (1)

- **Cél**: Megosztás révén támogatja a nagy finomságú objektumok tömegeinek hatékony felhasználását.

Pehelysúlyú (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/flyweight

Pehelysúlyú (3)



Pehelysúlyú (4)

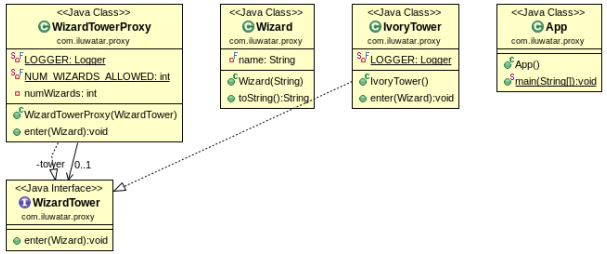
- **Ismert felhasználások**:
  - `java.lang.Byte#valueOf(byte b)` https://docs.oracle.com/javase/9/docs/api/java/lang/Byte.html#valueOf-byte-
  - `java.lang.Character#valueOf(char c)` https://docs.oracle.com/javase/9/docs/api/java/lang/Character.html#valueOf-char-
  - `java.lang.Integer#valueOf(int i)` https://docs.oracle.com/javase/9/docs/api/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.

Helyettes (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/proxy



Helyettes (3)

- **Ismert felhasználások**:
  - `java.lang.reflect.Proxy` http://docs.oracle.com/javase/9/docs/api/java/lang/reflect/Proxy.html
  - *Apache Commons Proxy* https://commons.apache.org/proper/commons-proxy/
  - *EasyMock* http://easymock.org/
  - *Mockito* http://site.mockito.org/

További szerkezeti minták

- Absztrakt dokumentum (*Abstract Document*) https://github.com/iluwatar/java-design-patterns/tree/master/abstract-document
- Modul (*Module*) https://github.com/iluwatar/java-design-patterns/tree/master/module
- Privát adatosztály (*Private Class Data*) https://github.com/iluwatar/java-design-patterns/tree/master/private-class-data
- …

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*)

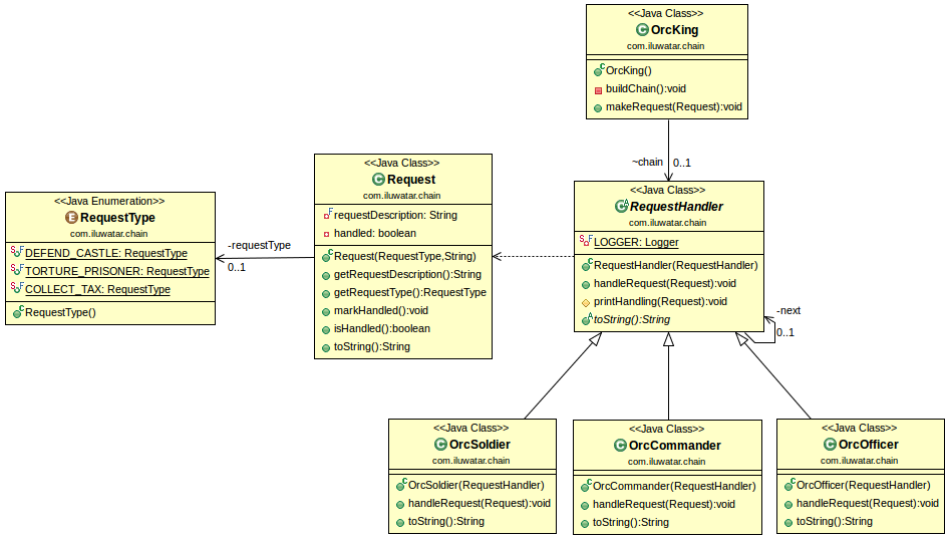
Felelősséglánc (1)

- **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.

Felelősséglánc (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/chain

Felelősséglánc (3)



Felelősséglánc (4)

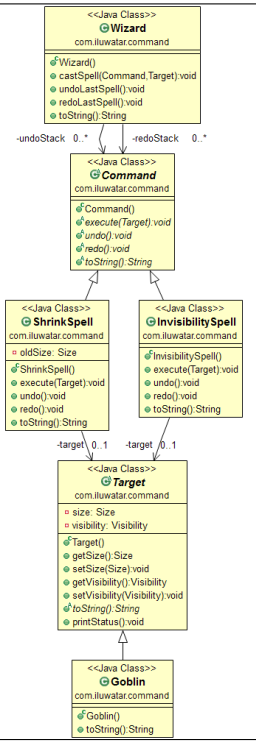
- **Ismert felhasználások**:
  - `java.util.logging.Logger` https://docs.oracle.com/javase/9/docs/api/java/util/logging/Logger.html
  - `javax.servlet.Filter` https://javaee.github.io/javaee-spec/javadocs/javax/servlet/Filter.html

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.

Parancs (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/command



Parancs (4)

- **Ismert felhasználások**:
  - `java.lang.Runnable` https://docs.oracle.com/javase/9/docs/api/java/lang/Runnable.html
  - `java.util.concurrent.Callable` https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Callable.html

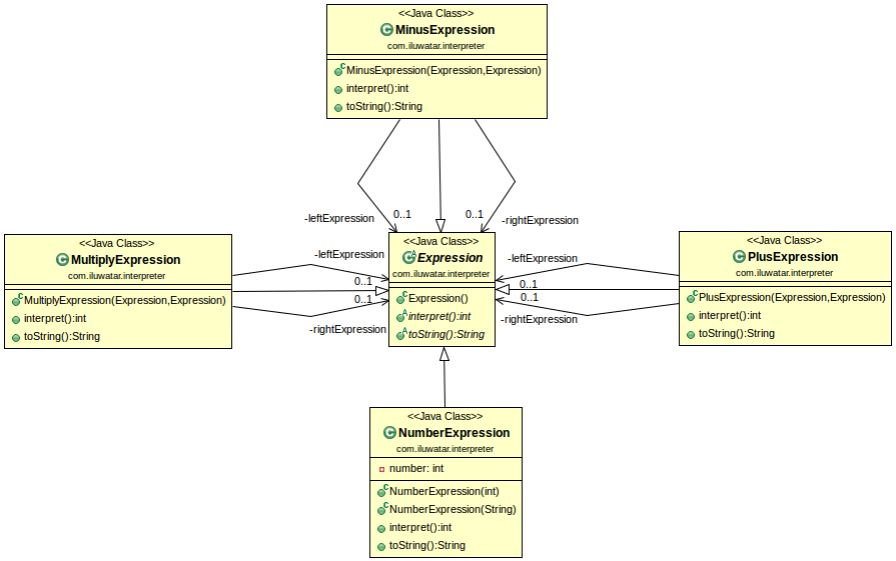
Értelmező (1)

- **Cél**: Egy adott nyelv nyelvtanát ábrázoljuk, illetve ehhez az ábrázoláshoz értelmezőt biztosítunk, amely annak alapján képes a nyelv mondatait megérteni.

Értelmező (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/interpreter

Értelmező (3)



Értelmező (4)

- **Ismert felhasználások**:
  - `java.text.Format` https://docs.oracle.com/javase/9/docs/api/java/text/Format.html
  - `java.time.format.DateTimeFormatter` https://docs.oracle.com/javase/9/docs/api/java/time/format/DateTimeFormatter.html
  - `java.util.regex.Pattern` https://docs.oracle.com/javase/9/docs/api/java/util/regex/Pattern.html

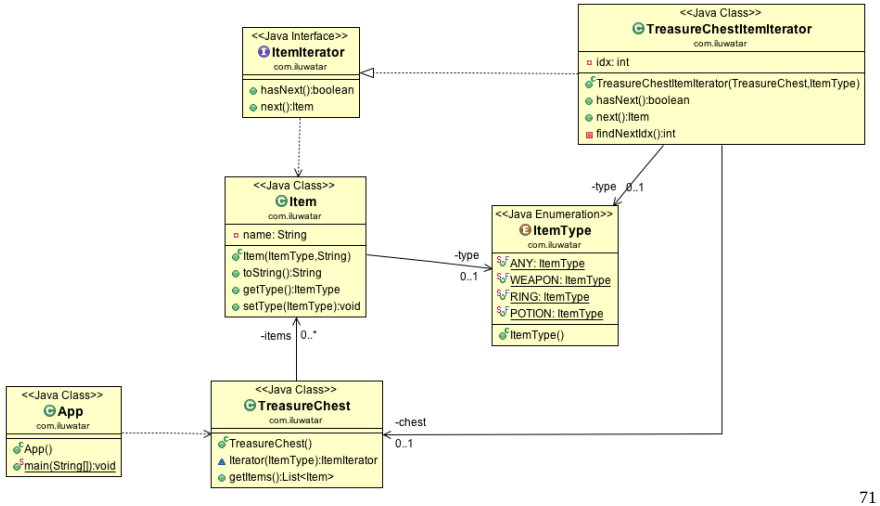
Bejáró (1)

- **Cél**: Az összetett objektumok elemeinek soros elérését a háttérben megbúvó ábrázolás felfedése nélkül biztosító módszer kialakítása.

Bejáró (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/iterator

Bejáró (3)



Bejáró (4)

- **Ismert felhasználások**:
  - `java.sql.ResultSet` https://docs.oracle.com/javase/9/docs/api/java/sql/ResultSet.html
  - `java.util.Enumeration` https://docs.oracle.com/javase/9/docs/api/java/util/Enumeration.html
  - `java.util.Iterator` https://docs.oracle.com/javase/9/docs/api/java/util/Iterator.html

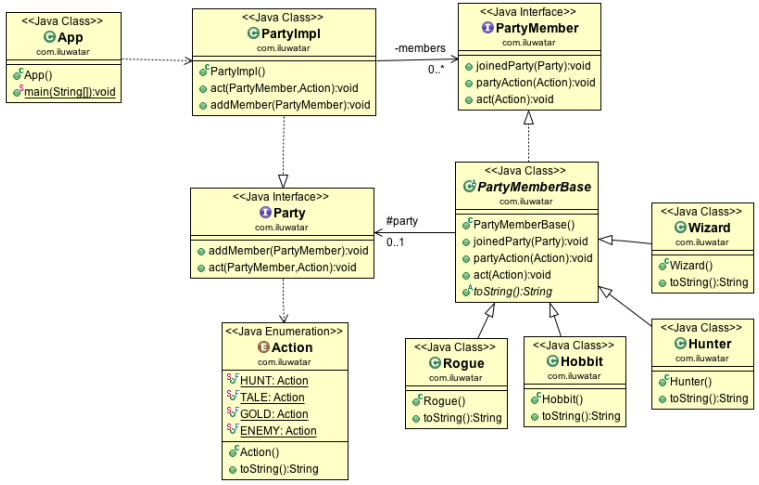
Közvetítő (1)

- **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.

Közvetítő (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/mediator

Közvetítő (3)



Közvetítő (4)

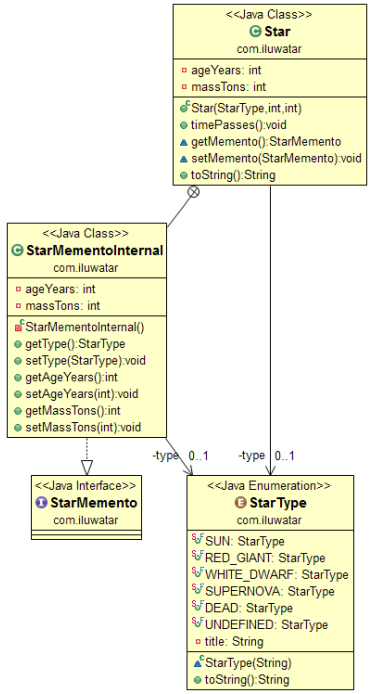
- **Ismert felhasználások**:
  - `java.util.concurrent.ExecutorService` https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/ExecutorService.html
  - `java.util.Timer` https://docs.oracle.com/javase/9/docs/api/java/util/Timer.html

Emlékeztető (1)

- **Cél**: Az egységbe zárás megsértése nélkül rögzíteni és felfedni egy objektum belső állapotát, hogy az később ebbe az állapotba visszaállítható legyen.

Emlékeztető (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/memento



Emlékeztető (4)

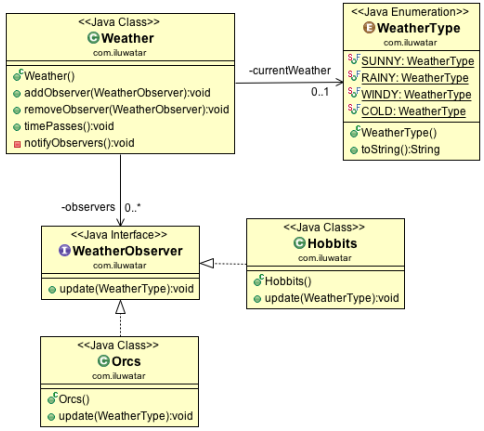
- **Ismert felhasználások**:
  - `java.util.Date` https://docs.oracle.com/javase/9/docs/api/java/util/Date.html
    - Lásd a `getTime()` és `setTime(long time)` metódusokat.

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.

Megfigyelő (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/observer



Megfigyelő (3)

- **Ismert felhasználások**:
  - `java.util.EventListener`
    https://docs.oracle.com/javase/9/docs/api/java/util/EventListener.html
  - `java.util.Observer`
    https://docs.oracle.com/javase/9/docs/api/java/util/Observer.html
    `java.util.Observable`
    https://docs.oracle.com/javase/9/docs/api/java/util/Observable.html
  - `javafx.beans.Observable`
    https://docs.oracle.com/javase/9/docs/api/javafx/beans/Observable.html
  - `java.util.concurrent.Flow`
    https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.html
    - Lásd:
      - *Reactive Streams* http://www.reactive-streams.org/
      - *Reactive Programming with JDK 9 Flow API* https://community.oracle.com/docs/DOC-1006738

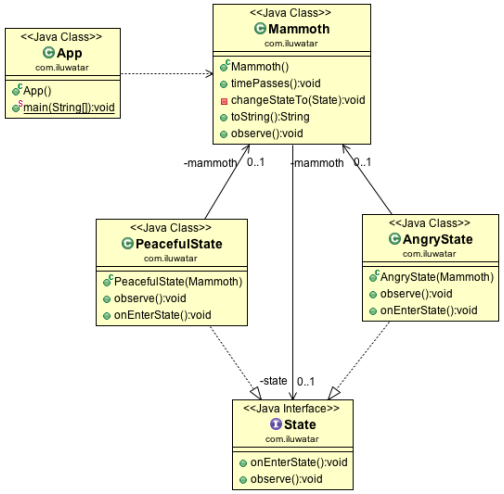
Állapot (1)

- **Cél**:
  - Egy adott objektum számára engedélyezni, hogy belső állapotának megváltozásával megváltoztathassa viselkedését is.
  - Az objektum ekkor látszólag módosítja az osztályát.

Állapot (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/state

Állapot (3)

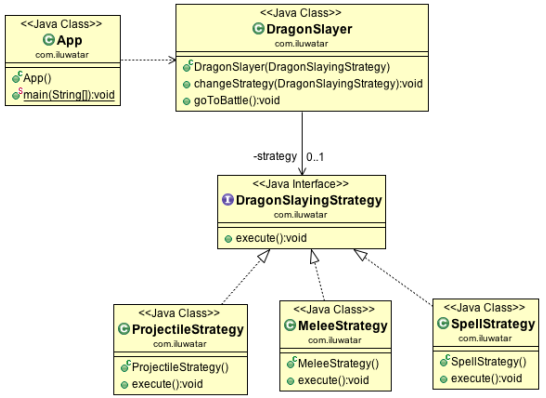


Stratégia (1)

- **Cél**:
  - Algoritmus-család meghatározása, melyben az algoritmusokat egyenként egységbe zárjuk és egymással felcserélhetővé tesszük.
  - E módszer révén az algoritmus az ügyféltől függetlenül módosítható.

Stratégia (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/strategy



Stratégia (3)

- **Ismert felhasználások**:
  - `java.util.Collections` https://docs.oracle.com/javase/9/docs/api/java/util/Collections.html
    - Lásd a `binarySearch()`, `max()`, `min()` és `sort()` metódusokat.
    - Kapcsolódó interfészek:
      - `java.lang.Comparable` https://docs.oracle.com/javase/9/docs/api/java/lang/Comparable.html
      - `java.util.Comparator` https://docs.oracle.com/javase/9/docs/api/java/util/Comparator.html

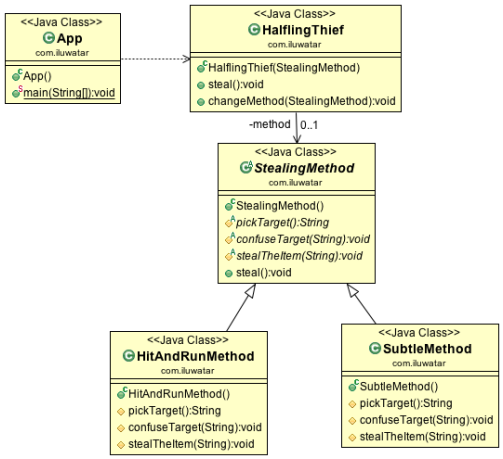
Sablonfüggvény (1)

- **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.

Sablonfüggvény (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/template-method

Sablonfüggvény (3)



Sablonfüggvény (4)

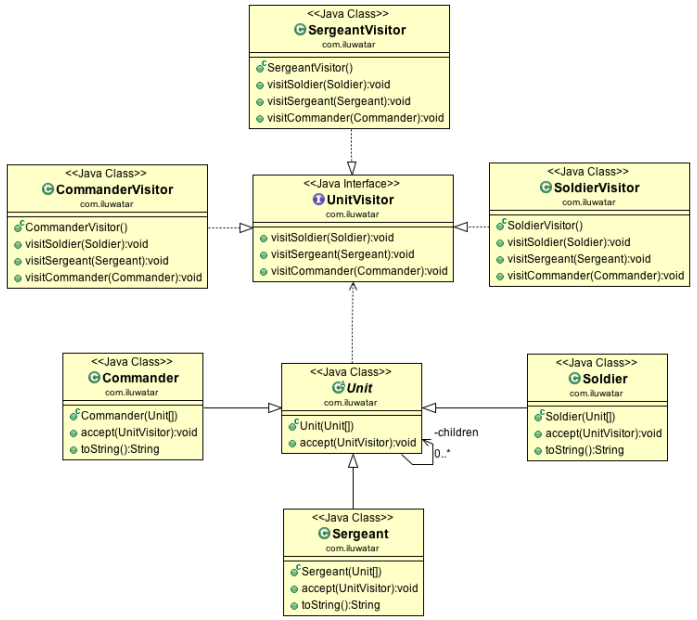
- **Ismert felhasználások**:
  - `java.io.InputStream` https://docs.oracle.com/javase/9/docs/api/java/io/InputStream.html
  - `java.io.OutputStream` https://docs.oracle.com/javase/9/docs/api/java/io/OutputStream.html
  - `java.util.AbstractList` https://docs.oracle.com/javase/9/docs/api/java/util/AbstractList.html
  - `java.util.AbstractMap` https://docs.oracle.com/javase/9/docs/api/java/util/AbstractMap.html
  - `java.util.AbstractQueue` https://docs.oracle.com/javase/9/docs/api/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.

Látogató (2)

- **Példakód**: https://github.com/iluwatar/java-design-patterns/tree/master/visitor



Látogató (4)

- **Ismert felhasználások**:
  - `java.nio.file.FileVisitor` https://docs.oracle.com/javase/9/docs/api/java/nio/file/FileVisitor.html
  - `javax.lang.model.element.ElementVisitor` https://docs.oracle.com/javase/9/docs/api/javax/lang/model/element/ElementVisitor.html

További viselkedési minták

- Null objektum (*Null Object*) https://github.com/iluwatar/java-design-patterns/tree/master/null-object
- Specifikáció (*Specification*) https://github.com/iluwatar/java-design-patterns/tree/master/specification
- …

Null objektum (1)

- **Cél**: A `null` referencia alternatíváját biztosítja egy objektum hiányának jelzésére egy olyan objektum révén, mely az elvárt interfészt üres metódustörzsekkel implementálja.

Null objektum (2)

- **Mintakód**: https://github.com/iluwatar/java-design-patterns/tree/master/null-object