Objektumorientált programozás

Adatmentés és –betöltés, segédosztályok, egyebek

Goldschmidt Balázs

balage@iit.bme.hu

Objektumok mentése és betöltése

Szerializálás alapjai

- Feladat
  - mentsük ki az objektumainkat, majd később töltsük vissza őket
  - mit szeretnénk eltárolni?
    - attribútumokat
    - kapcsolatokat
- Egyszerű megoldás: *serialization*
  - beépített funkcionalitás
  - az objektumok tartalmát bit-sorozattá alakítja
  - bitsorozatból felépíti az objektumokat

Szerializálás fogalmai

- Szerializálás (serialization)
  - objektumok bitsorozattá alakítása (bit stream)
    - alternatív nevek: sorosítás, marshalling, deflating
  - az előálló bitsorozat már kiírható, átadható, stb.
- Deszerializálás (deserialization)
  - bitsorozat visszaalakítása objektumokká
    - alternatív nevek: visszasorosítás, unmarshalling, inflating
  - a bitsorozat jöhet fájlból, hálózatról, stb.

Szerializálás példa: kiíratás

```
import java.io.Serializable;
public class MyImportantData implements Serializable
{ ... }
```

```
//import java.io.*;
...
MyImportantData data = ...;
try {
        FileOutputStream f =
                        new FileOutputStream(”filename”);
        ObjectOutputStream out =
                        new ObjectOutputStream(f);
        out.writeObject(data);
        out.close();
}
catch(IOException ex) { ... }
```

Szerializálás példa: visszaolvasás

```
//import java.io.*;
...
MyImportantData data2;
try {
        FileInputStream f =
                        new FileInputStream("filename");
        ObjectInputStream in =
                        new ObjectInputStream(f);
        data2 = (MyImportantData)in.readObject();

        in.close();

} catch(IOException ex) {
        ...
} catch(ClassNotFoundException ex) {
        ...
}
```

Szerializálás szabályai

- Csak a Serializable interfészt megvalósítók
  - ha az ősosztály megvalósítja, OK
  - tömbök, String, Integer, Double, stb. OK
- *Serializable* interfész
  - nincs metódusa
  - a fordítónak jelzi az extra feladatot
- *Nem sorosítható*
  - *Object, Socket, Input/OutputStream, System, stb.*

Szerializálás szabályai

- Mit sorosítunk?
  - primitív attribútumokat
  - szerializálható attribútumokat
    - rekurzívan!
- Mit nem sorosítunk?
  - statikus mezőket
  - *transient* mezőket

```
public class Soros implements Serializable {
        transient private String titok;
        private String nemtitok;
        ...
```

Szerializálás folyamata

- Szerializáció rekurzív
  - hogyan kerüljük el a végtelen ciklust?
  - minden objektum csak egyszer íródik ki
  - minden további hivatkozás csak referál

```
out.writeObject(a); // teljes objektum
out.writeObject(b); // teljes objektum
out.writeObject(a); // csak referencia!
```

- Örökölt mezők kiíratása
  - csak a legelső szerializálható őstől kezdve

Szerializálás folyamata 2

- Legyenek a következő osztályaink!

```
class Student implements
    Serializable {
  String name;
  University uni;
  Address a;
  double average;
  // ctr, set, get
}
```

```
class Address implements
    Serializable {
  String country,
    city, street;
  // ctr, set, get
}
```

```
class University implements
    Serializable {
  String name;
  Address a;
  // ctr, set, get
}
```

Szerializálás folyamata 3

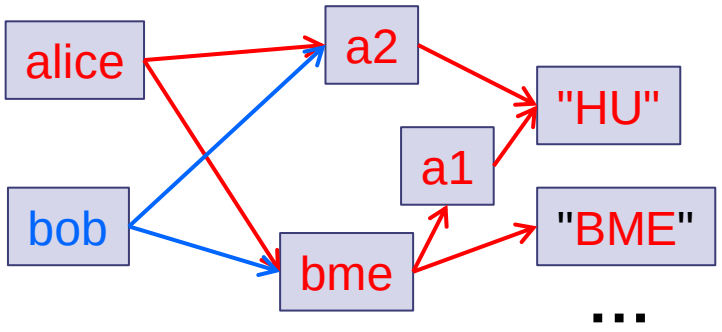
- Legyenek a következő objektumaink!

```
Address a1 = new Address("HU", "Bp", "Műegyetem rkp 3");
Address a2 = new Address("HU", "Bp", "Alkotás u. 10");

University bme = new University("BME", a1);

Student alice = new Student("Alice", bme,  a2);
Student bob   = new Student("Bob", bme, a2);
```

```
ObjectOutputStream oos = ...;
oos.writeObject(alice);
oos.writeObject(bob);
oos.close();
```



Egyéb adatmentési eljárások

- Szerializálás nagyon alacsony szintű
  - csak Java-ban kezelhető, adatok monolitban
- JSON (**J**ava**S**cript **o**bject **n**otation)
  - org.json, javax.json, gson (Google)
- XML (e**X**tensible **M**arkup **L**anguage)
  - JAXB: objektum-xml leképezés
  - JDOM, SAX: közvetlen XML feldolgozás
- Adatbázis
  - JDBC: SQL elérés
  - JPA: objektum-relációs támogatás

Gyakori segédosztályok (utility classes)

Properties

- Konfigurációs fájlok kezeléséhez
  - név-érték párok szöveges formátumban
  - hogyan olvassuk be helyesen?
    - saját beolvasó algoritmus megvalósítása: overkill
    - szabványos segédosztály: *Properties*
- *Properties* osztály
  - Map interfész (java.util.Map)
  - kényelmes beolvasás és kiíratás
    - sokféle formátumot támogat

Properties

- *java.util.Properties* osztály
  - Map interfész *(String* kulcs*, String* érték*)*
  - *getProperty* (default értékkel is), *setProperty*
    - property lekérdezése és beállítása
  - *Set<String> stringPropertyNames()*
    - kulcsok halmaza
  - *load(InputStream* vagy *Reader)*
    - beolvasás bármilyen forrásból (fájl, hálózat, stb.)
    - sokfajta formátum (kettőspont, egyenlőségjel, stb.)
  - *store(OutputStream* vagy *Writer, String comments)*
    - tartalom kiíratása bárhova (fájl, hálózat, stb., megadott kommenttel)
  - XML formátum szintén támogatva

StringBuffer és StringBuilder

- Változtatható tartalmú szöveg tárolása
- Sztring-intenzív műveletekhez
  - összefűzés (hozzáfűzés, beszúrás, stb.)
  - karaktermódosítás, -törlés, stb.
- StringBuffer
  - szálbiztos, lassabb (szerver oldalon)
- StringBuilder
  - egyszálú esetre, gyorsabb (kliens oldalon)

StringBuilder példa

- Olvassunk be sorokat, és fűzzük őket egybe!

```
InputStreamReader isr =
      new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
StringBuilder sb = new StringBuilder();
while (true) {
      String line = br.readLine();
      if (line == null) break;
      sb.append(line);
}
br.close();
System.out.println(sb.toString());
```

Idő és dátum támogatása

- Date
  - egy időpillanat reprezentálására
    - millisec pontosság
  - kifutóban...
- Calendar
  - abstract
  - dátumrendszerek egységes kezeléséhez
- GregorianCalendar *extends Calendar*
- LocalTime, LocalDate, stb.
- DateFormat
  - dátum feldolgozásához és kiíratásához

Date

- Date() és Date(long d)
  - ctr (aktuális idő vagy *d* (ms 1970-01-01UTC00:00:00 óta)
- boolean after/before(Date d)
- int compareTo(Date d)
- int equals(Object o)
  - triviális összehasonlítások
- long getTime()
  - ms a fenti időpont (epoch) óta
- String toString()
  - visszaadja “*dow mon dd hh:mm:ss zzz yyyy*” formátumban

Calendar

- Dátum ábrázolására
  - absztrakt osztály
  - a statikus *getInstance()* visszaadja az aktuális időt
- Dátum és időkezelés generikus metódusokkal
  - *add(f, delta)*
  - *set(f, delta), get(f), clear(f)*
  - *roll(f, delta)*
    - *f* mezőn *delta* növelés, állítás, forgatás, lekérés, nullázás
    - ha kell, a többi mezőt is igazítja (pl. 31-e plusz 2 nap)
    - *f* konstans értékekkel adható meg

Calendar

- Mezők konstansai
  - DATE, DAY_OF_MONTH, DAY_OF_WEEK, DAY_OF_YEAR
  - WEEK_OF_MONTH, WEEK_OF_YEAR
  - ERA, YEAR, MONTH, MINUTE, SECOND, MILLISECOND
  - AM_PM, HOUR_OF_DAY (0-24), HOUR (0-12)
  - ZONE_OFFSET
- Mezők értékeihez konstansok
  - JANUARY, FEBRUARY, MARCH, stb.
    - JANUARY==0 **!!!**
  - MONDAY, TUESDAY, stb.
    - SUNDAY==0, MONDAY==1, stb.
  - AM, PM
  - (GregorianCalendar: AD, BC)

Calendar

- Metódust biztosít mindenhez ☺
  - boolean after/before(Object when)
  - int clear()
  - int get[Actual]Maximum(f)
  - int get[Actual]Minimum(f)
  - int getGreatestMinimum(f)
  - int getLeastMaximum(f)
  - get/setFirstDayOfWeek
  - get/setMinimalDaysInFirstWeek
  - …

GregorianCalendar

- Calendar leszármazottja
  - Konstruktorok
    - év, hó, nap *[,óra, perc [, másodperc]]*
  - Konstansok
    - *AD, BC*
  - Metódusok
    - setGregorianChange(Date date)
    - Date getGregorianChange()
    - boolean isLeapYear()
    - …

Calendar példa

```
Calendar c = new GregorianCalendar(1996,0,23);//96-01-23
c.set(Calendar.MONTH, Calendar.MAY); // 1996-05-23
c.set(Calendar.DATE, 31);            // 1996-05-31

c.add(Calendar.MONTH, 15);           // 1997-08-31
c.roll(Calendar.DATE, 10);           // 1997-08-10

DateFormat df = DateFormat.getDateTimeInstance();
System.out.println(df.format(c.getTime()));
   // Aug 10, 1997 12:00:00 AM
```

LocalTime, LocalDate, stb.

- Egyszerű, szép idő- és dátumkezelés
  - package *java.time*
  - sokkal OO-bb, mint *Date* vagy *Calendar*
    - getter/setter metódusok, isAfter, isBefore, isEqual, until, stb.
    - statikus metódusok objektumlétrehozásra (now, of, stb.)
  - nem támogat időzónát (Local...!)
  - értéke nem módosítható (immutable)
- *LocalTime*: csak idő (óra, perc, mp)
- *LocalDate*: csak dátum (év, hó, nap)
- *LocalDateTime*: dátum és idő egyben

LocalDate, LocalTime példa

```
LocalDate d1 = LocalDate.of(1996,1,23); //96-01-23
LocalDate d2 = d1.plusDays(100);        //96-05-02
// van még plusWeeks, plusMonths, plusYears, stb
// ugyanezek minus...-szal
```

```
LocalTime t1 = LocalTime.of(14,32,56); // 14h 32m 56s
LocalTime t2 = LocalTime.of(15,10);    // 15h 10m 00s
boolean b = t1.isBefore(t2);           // true
// minusSeconds, minusMinutes, minusHours, minusNanos,
// plusSeconds, plusMinutes, plusHours, plusNanos, stb.
```

```
LocalDateTime dt1 = t1.atDate(d1);  // 96-01-23 14:32:56
LocalDateTime dt2 = d2.atTime(t2);  // 96-05-02 15:10:00
long hours = dt1.until(dt2, HOURS); // 2400 óra különbség
```

Random

- Véletlenszámok generálására
  - konstruktorok
    - *default*: automatikus inicializálás
    - dedikált inicializálás (*long* paraméter), determinisztikus működés *(setSeed* is van*)*
  - *nextXXX()*
    - egyenletes eloszlás
    - *boolean*, *bytes*, *int*, *long*: eredmény a típus értékkészletén
    - *double*, *float*: eredmény a [0.0, 1.0) halmazon
    - *nextInt(int n)*: 0 and *n* közötti érték (jobbról nyílt)
  - *nextGaussian()*
    - normális eloszlás (átlag: 0, szórás: 1)

Math/StrictMath

- Matematikai segédosztályok
  - java.lang.Math, java.lang.StrictMath
- Konstansokat definálnak
  - E, PI
- Függvényeket valósítanak meg
  - abs, signum, sqrt, *cbrt, ceil, floor, round, rint,*
  - sin, cos, tan, *sinh, cosh, tanh*
  - asin, acos, atan, atan2
  - pow, exp, *expm1,* log, log10, *log1p, scalb,*
  - max, min, *nextAfter, nextUp,* toDegrees, toRadians

Nagy egészek ábrázolása

- BigInteger
  - tetszőlegesen nagy egész szám
  - statikus konstansok
    - *0 (ZERO)*, *1 (ONE)*, *10 (TEN)*
  - konstruktor *byte[], random, long, int* vagy *String* paraméterrel
  - műveletek metódusokkal implementálva
    - *add*, *abs*, *and*, *pow*, stb.
  - módosító metódusok
    - *setBit*, *testBit*, stb.
  - lekérdező metódusok
    - *toByteArray*, *toString*, *longValue*, stb.

Nagy valósok ábrázolása

- BigDecimal
  - tetszőleges méretű lebegőpontos számok
  - statikus konstansok
    - *0 (ZERO)*, *1 (ONE)*, *10 (TEN)*
    - kerekítés *(ROUND_DOWN, ROUND_UP, ROUND_HALF_UP, stb.)*
  - konstruktorok *char[], String, BigInt., double, long, int* paraméterrel
  - műveletek metódusokkal implementálva
    - *add*, *abs*, *round*, *pow*, stb
  - módosító metódusok
    - *setScale*, *testBit*, stb.
  - lekérdező metódusok
    - *scale, precision, toBigInteger*, *toString*, *doubleValue*, stb.

Scanner

- Konstruktor
  - paraméter: *File*, *InputStream*, *Path*, *Readable*, *String*
- Metódusok
  - *hasNext[XXX]*
  - *next[XXX]*
    - BigDecimal, BigInteger, boolean, byte, double, …
  - *useDelimiter(Pattern|String), delimiter()*
  - *useRadix(int radix), radix()*
    - for setting/getting delimiter and radix
  - *findInLine, skip, match, etc*

Felsorolás típus

Enum: objektumtípus

- Felsorolás típus: saját értékkészlet
  - attribútumokkal
  - metódusokkal
- Enum-ok...
  - szerializálhatók
  - kiírathatók
  - for-each-elhetők
  - switch-case-ben is használhatók

```
public enum Planet {
    Mercury, Venus, Earth, Mars,
    Jupiter, Saturn, Uranus, Neptune
}
```

Enum példa

```
public enum Planet { // complex enum
    Mercury(3.3e23, 2.44e6), Venus(4.868e24, 6.052e6),
      Earth(5.972e24, 6.371e6), Mars(6.417e23, 3.39e6),
      Jupiter(1.899e27, 6.991e7), Saturn(5.684e26, 5.823e7),
      Uranus(8.681e25, 2.536e7), Neptune(1.024e26, 2.462e7);

    private final double mass, radius; // kg, m

    Planet(double m, double r) { mass = m; radius = r;}

    public double mass() { return mass; }
    public double radius() { return radius; }
    public double sGrav() { return 6.674e-11*mass/radius/radius; }
}
```

```
for (Planet p : Planet.values()) {
        System.out.println(p+": "+p.sGrav());
}
```

Enum példa 2

```
enum Letter {A, B, C}
```

```
Letter e = Letter.A;
switch (e) {
      case A: System.out.println("A!"); break;
      case B: System.out.println("B!"); break;
      case C: System.out.println("C!"); break;
}
```

Static import of members

```
import static mypackage.Letter.*;
```

```
if (e == B) { ... }
```

Enum metódusok

- **`String name()`**
  - enum érték neve (megegyezik a forráskódbelivel)
- **`int ordinal()`**
  - enum érték sorszáma
- **`static <T extends Enum<T>> T valueOf([Class<T> enumType,] String name)`**
  - visszaadja az adott nevű enum értéket
- **`static <T extends Enum<T>> T[] values()`**
  - visszaadja a típus összes értékét egy tömbben

Egyebek

Változó hosszú paraméterlista

- Pre 1.5: tömböt adtunk át
  - kényelmetlen

```
void foo(String s, double[]  da) { for (double d : da) …; }
```

- 1.5: *varargs*
  - akár tömb, akár vesszővel elválasztott értékek
  - csak a paraméterlista végén

```
void foo(String s, double... da) { for (double d : da) …; }
```

```
double[] dd = {1.1, 2.5, 5.1};
foo("X", dd);
foo("X", 1.2, 3.4, 5.6, 7.8, 8.9, 9.11);
```

Listainicializálás

- *Arrays.asList*
  - `List<T> asList(T... t)`
  - visszaad egy fix méretű listát, ami
    - *t* elemeket tartalmazza
    - lista módosítása visszahat a *t* tömbre

```
List<String> friends =
    Arrays.asList("Andi", "Béci", "Cili", "Dani");
```

```
void foo(List<String> sl, List<Double> dl) { … }
```

```
foo(Arrays.asList("A", "B", "C"),
    Arrays.asList(1.1, 2.2, 3.3));
```

Annotációk

- Extra információ a forráskódban

```
public @interface Copyright {
    String value() default "2008 Me";
}
@Copyright("2008 Bytemongers Limited")
public class ÁrvíztűrőTükörfúrógép { ... }
```

  - *Deklaráció: interface egy kezdő @ jellel*
  - *Metódusai az annotáció tartalmát adják meg*
    - *return érték lehet primitív, String, Class, enum*
  - *Lehet default értéke a tartalomnak*

Annotációk használata

- Általában a metaadatokat adják meg
  - fordító számára
  - kódgenerátor számára
  - stb.
- Példa: metódus felüldefiniálását jelezzük

```
@Override
public int read() throws IOException {
      return super.read()-1;
}
```