Objektumorientált programozás

Objektumok kapcsolatai és kivételkezelés

Goldschmidt Balázs

balage@iit.bme.hu

Objektumok kapcsolatai

Egyetemi nyilvántartás

- Legyen a feladat:
  - egyetemi nyilvántartás
  - hallgatók
    - név, neptunkód, szül. év, átlag, megsz. kreditek
  - oktatók
    - név, neptunkód, szül. év, beosztás

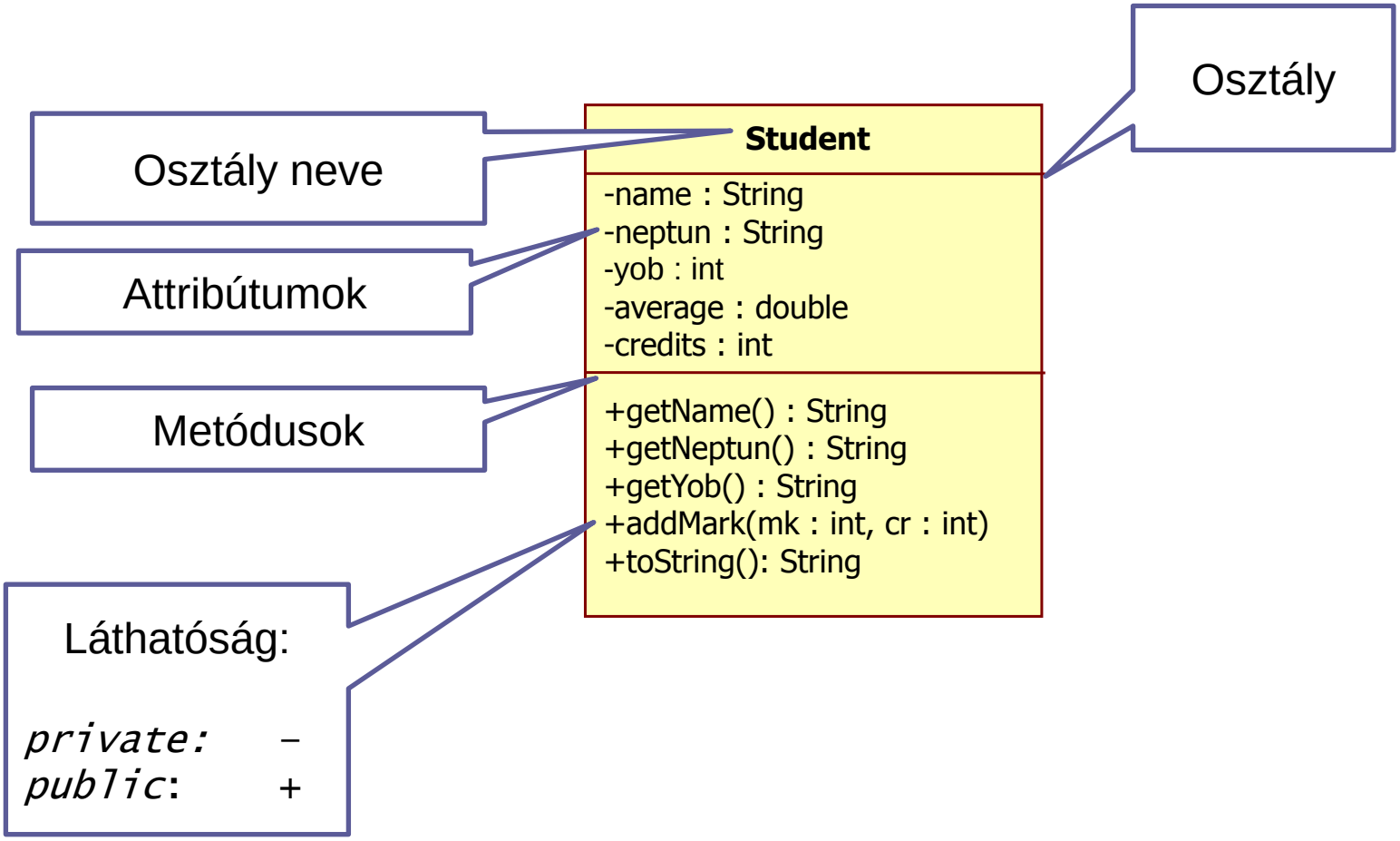
Hallgató

```
public class Student {
  private String name; private String neptun;
  private int yob; // Year Of Birth
  private double average;
  private int credits;

  public Student(String na, String ne, int y) {
    name = na; neptun = ne; yob = y;
    average = 0.0; credits = 0;
  }

  public String getName() { return name; }
  public String getNeptun() { return neptun; }
  ...
```

Hallgató

```
  ...

  public void addMark(int mark, int credit) {
    average = (average*credits + mark*credit) /
              (credits+credit);
    credits += credit;
  }
  public String toString() {
    return name+" ("+neptun+") "
           +yob+", "+average+", "+credits;
  }
}
```

UML jelölés (osztálydiagram)



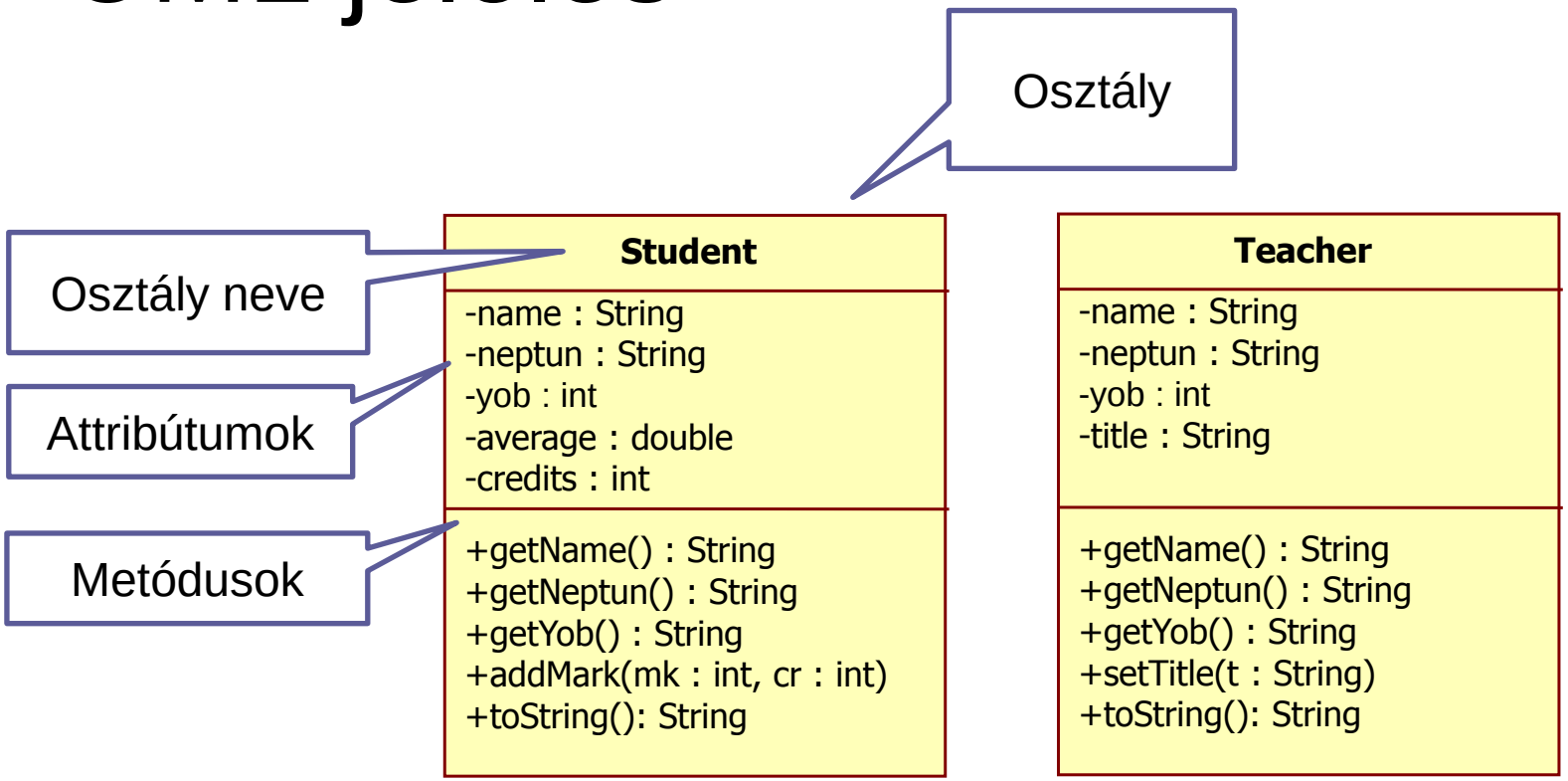
Oktató

```
public class Teacher {
  private String name; private String neptun;
  private int yob; // Year Of Birth
  private String title;

  public Teacher(String na, String ne, int y) {
    name = na; neptun = ne; yob = y;
    title = "assistant teacher";
  }

  public String getName() { return name; }
  public String getNeptun() { return neptun; }
  ...
```

Oktató

```
  ...

  public void setTitle(String s) {
       title = s;
  }
  public String toString() {
    return name+" ("+neptun+") "
           +yob+", "+title;
  }
}
```

UML jelölés


Osztály
Osztály neve
Attribútumok
Metódusok
Student
-name : String
-neptun : String
-yob : int
-average : double
-credits : int
+getName() : String
+getNeptun() : String
+getYob() : String
+addMark(mk : int, cr : int)
+toString(): String
Teacher
-name : String
-neptun : String
-yob : int
-title : String
+getName() : String
+getNeptun() : String
+getYob() : String
+setTitle(t : String)
+toString(): String

Példa

```
Student s1 = new Student("Gipsz Jakab", "1A2B3C", 1996);
Student s2 = new Student("Nagy Károly", "XXX111", 1998);
Student s3 = new Student("Kis Pippin", "111XXX", 1999);

s1.addMark(4, 2);

Teacher t1 = new Teacher("Rend Elek", "Q1W2E3", 1973);
Teacher t2 = new Teacher("Csirke Béla", "OKTAT6", 1980);

System.out.println(s1);
System.out.println(s2);
System.out.println(s3);

System.out.println(t1);
System.out.println(t2);
```

Egyetemi nyilvántartás

- Ami megvan:
  - hallgatók
    - név, neptunkód, szül. év, átlag, megsz. kreditek
  - oktatók
    - név, neptunkód, szül. év, beosztás
- Bővítsük:
  - kurzusok
    - név, neptun, ki oktatja, ki hallgatja

Kurzustól elvárt viselkedés

- Törzsadatok kezelése
  - getter-setter a névhez, neptunkódhoz, stb
- Oktató hozzárendelése
  - maximáljuk a számukat
- Hallgató hozzárendelése
  - konstruktorban állítható számosság
- Hallgató törlése
- Hallgatói lista nyomtatása

Kurzus

```
public class Course {
  private String name;
  private String neptun;
  private Student students[];
  private Teacher teacher;

  public Course(String na, String ne, int sn) {
    name = na; neptun = ne;
    students = new Student[sn]; // no Student is created!!!
  }

  public String getName() { return name; }
  public String getNeptun() { return neptun; }
  ...
```

Kurzus

```
...
  public void setTeacher(Teacher t) { teacher = t; }
  public void addStudent(Student s) {
    for (int i = 0; i < students.length; i++) {
      if (students[i] == null) {
        students[i] = s;
        return;
      }
    }
  }
  public void listStudents() {
    for (int i = 0; i < students.length; i++) {
      if (students[i] != null) {
        System.out.println(students[i]);
      }
    }
  }
...
```

Ha a ciklus után lenne teendő, *break* kellene

Kurzus

```
...
  public void removeByNeptun(String neptun) {
    for (int i = 0; i < students.length; i++) {
      if (students[i] == null) continue;
      String actNeptun = students[i].getNeptun();
      if (actNeptun.equals(neptun)) {
        students[i] = null;
        return;
      }
    }
  }
  public String toString() {
    return name+" ("+neptun+")";
  }
}
```

Ha a pozíció üres, jöjjön a következő

Csak kivesszük a tömbből, a helyén "lyuk" marad

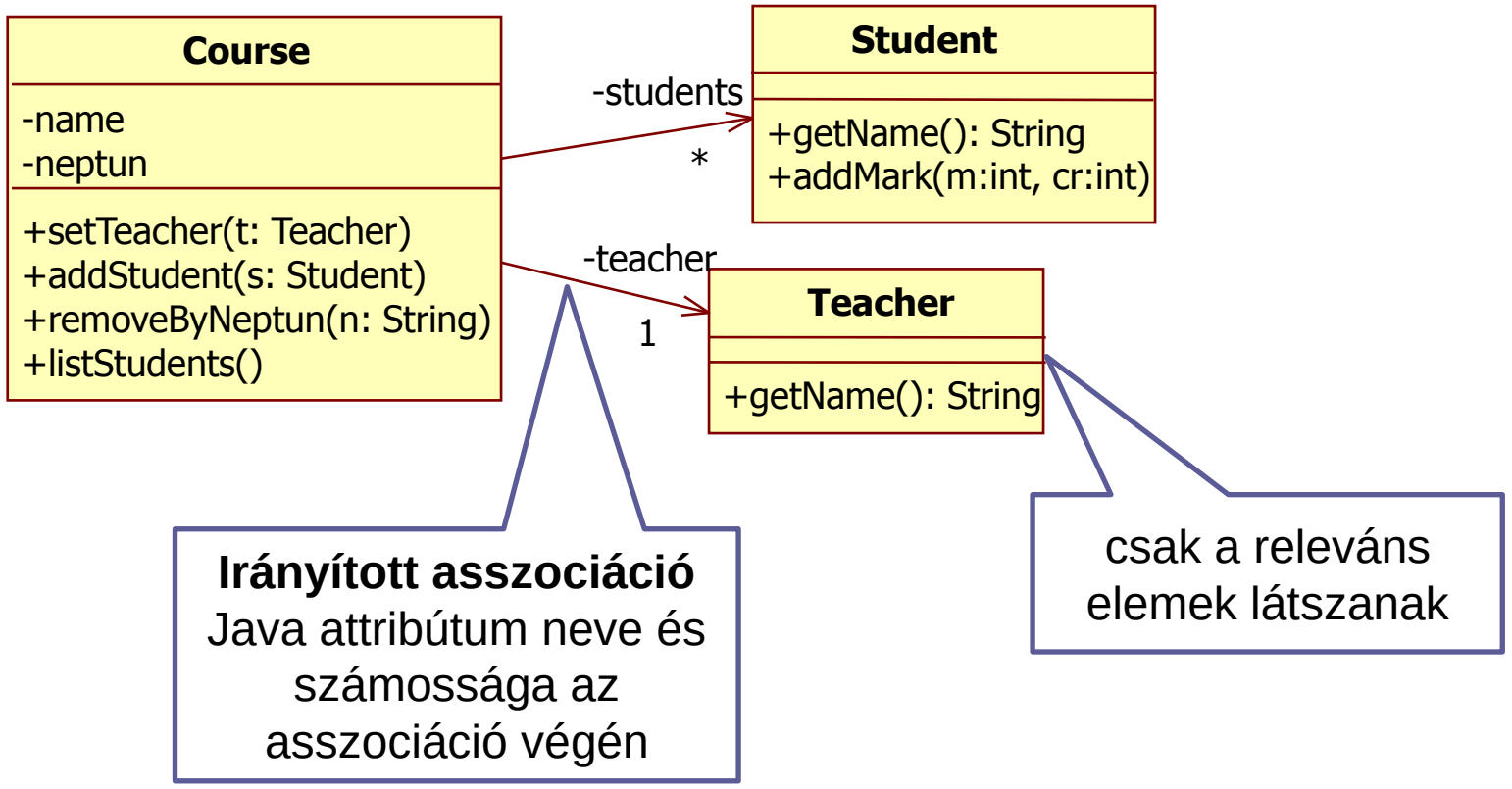
Ha a ciklus után lenne teendő, *break* kellene

equals a Stringek tartalmát hasonlítja

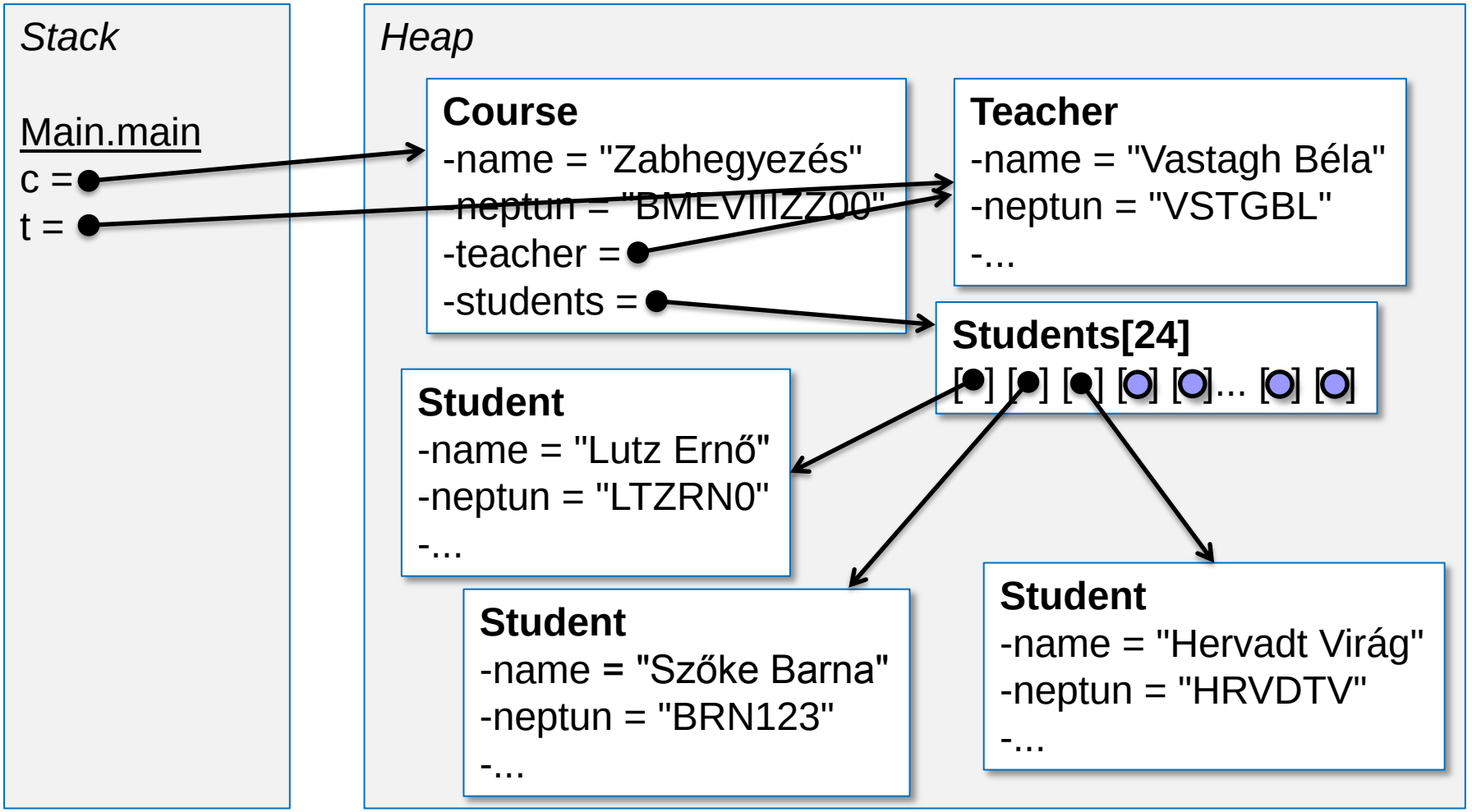
Példaprogram

```
public class Main {
  public static void main(String[] args) {
    Course c = new Course("Zabhegyezés", "BMEVIIIZZ00", 24);
    Teacher t = new Teacher("Vastagh Béla", "VSTGBL", 1975);
    c.setTeacher(t);

    c.addStudent(new Student("Lutz Ernő", "LTZRN0",1997));
    c.addStudent(new Student("Szőke Barna", "BRN123",1997));
    c.addStudent(new Student("Hervadt Virág", "HRVDTV",1998));
    c.listStudents();

    c.removeByNeptun("BRN123");
    c.listStudents();
  }
}
```

UML jelölés


Course
-name
-neptun
+setTeacher(t: Teacher)
+addStudent(s: Student)
+removeByNeptun(n: String)
+listStudents()
-students
*
Student
+getName(): String
+addMark(m:int, cr:int)
-teacher
1
Teacher
+getName(): String
Irányított asszociáció
Java attribútum neve és számossága az asszociáció végén
csak a releváns elemek látszanak

Memóriakép


Stack
Main.main
c =
t =
Heap
Course
-name = "Zabhegyezés"
-neptun = "BMEVIIIZZ00"
-teacher =
-students =
Teacher
-name = "Vastagh Béla"
-neptun = "VSTGBL"
-...
Students[24]
[] [] [] [] []... [] []
Student
-name = "Lutz Ernő"
-neptun = "LTZRN0"
-...
Student
-name = "Szőke Barna"
-neptun = "BRN123"
-...
Student
-name = "Hervadt Virág"
-neptun = "HRVDTV"
-...

Kivételkezelés

Probléma kezelése

- Mi történik, ha betelik a tárgy?
- Jelen megoldás
  - a hallgató nem kerül be ✓
  - nincs visszajelzés ☹
- Elvárt viselkedés
  - a hallgató nem kerül be ✓
  - valami visszajelzést kapunk *?*

Probléma kezelése 2.

- 1. ötlet: visszatérési érték
  - boolean
    - *true*: siker
    - *false*: nem fért be
  - int
    - az új méret
    - gondok: ehhez tudni kellene a régit is...

Hallgató hozzáadása boolean

```
...
  public boolean addStudent(Student s) {
    for (int i = 0; i < students.length; i++) {
      if (students[i] == null) {
        students[i] = s;
        return true;
      }
    }
    return false;
  }

  public boolean removeByNeptun(String neptun) {
    // fentihez hasonlóan
...
```

Hibakezelés bemutatása

```
...
Course c = new Course("Zabhegyezés", "BMEVIIIZZ00", 24);
Teacher t = new Teacher("Vastagh Béla", "VSTGBL");
c.setTeacher(t);

if (!c.addStudent(new Student("Lutz Ernő", "LTZRN0"))) {
  System.out.println("Hallgató hozzáadása sikertelen!");
}
c.addStudent(new Student("Szőke Barna", "BRN123"));
...
```

Itt nem ellenőrzünk

Itt ellenőrzünk

Probléma kezelése 3.

- 2. ötlet: valami jobb kellene
  - különüljön el a hibakezelés és a normál működés
    - ne keveredjen a visszatérési érték és a hibajel
  - legyen szabványos
  - legyen szintaktikailag (fordítás során) ellenőrzött

Kivételkezelés

- Pythonhoz hasonló
- Ha hiba történik
  - jelezzük a hibát a megtörténés helyén
    - függetlenül a visszatérési értéktől

    → kivétel eldobása (*throw*)
  - dolgozzuk fel ott, ahol használni akartuk a funkciót

    → kivétel elkapása (*try-catch-finally*)
  - ehhez használjunk objektumokat

    → kivétel osztály és leszármazottai (*Exception*)

Hallgató hozzáadása exceptionnel

```
...
public void addStudent(Student s) throws Exception {
  for (int i = 0; i < students.length; i++) {
    if (students[i] == null) {
      students[i] = s;
      return;
    }
  }
  throw new Exception("Tárgy megtelt");
}

public void removeByNeptun(String neptun) throws Exception {
  // fentihez hasonlóan, csak a szöveg más
...
```

Új kivétel létrehozása és eldobása

jelezni kell, hogy kivétel várható

Kivétel elkapása

```
...
Course c = new Course("Zabhegyezés", "BMEVIIIZZ00", 24);
Teacher t = new Teacher("Vastagh Béla", "VSTGBL");
c.setTeacher(t);
try {
  c.addStudent(new Student("Lutz Ernő", "LTZRN0"));
  c.addStudent(new Student("Szőke Barna", "BRN123"));
  c.addStudent(new Student("Hervadt Virág", "HRVDTV"));
} catch (Exception e) {
  System.err.println(e.getMessage());
  // e.printStackTrace();
}
...
```

Itt jelezzük, hogy hiba volt

Kivételkezelés szabályai

- El nem kapott kivétel típusát jelezni kell a metódus fejlécében (*throws*)
  - lehet a kivétel őstípusa is
- Metódus dobásakor megszakad a metódus végrehajtása
- A dobott kivételt a hívási láncban első, a dobottal kompatibilis típust váró *catch* ág kapja el

Csomagoló osztályok

Primitív típusok objektumként

- Néha szükség van primitív típusra referenciával
  - pl. ArrayList eleme
  - angolul *wrapper* (csomagoló) osztályok)
- Minden primitívhez nagybetűs csomagoló
  - double – Double
  - int – Integer
  - boolean – Boolean
  - stb.

Csomagoló osztályok

- Automatikusan konvertálódik
  - ún. dobozolás, boxing
  - nem hatékony!
- Tartalmaz hasznos konstansokat és metódusokat
  - pl. *Double.NaN*, *Double.POSITIVE_INFINITY*,
  - pl. *double Double.parseDouble(String s), boolean Double.isNaN(double d)*

Csomagoló példák

```
public static void main(String args[]) {
    int a = 2;
    Integer b = 3;
    a = a + b;
    Integer d = 3+3;       // int -> Integer
    System.out.println(d*3); // Integer -> int

    Integer i1 = Integer.parseInt(args[0]);// boxing
    Integer i2 = Integer.parseInt(args[0]);// boxing

    i1.equals(i2);  // igaz, nincs boxing
    i1 == i2;       // csak ha -128<i1≤127
    i1 <= i2;       // igaz, van boxing
    i1++;                   // inkr, van boxing
    i1 > 120;       // van boxing
}
```

Segédosztályok

Segédosztályok: *Math*

- Konstansok
  - E, PI
- Függvények (mind static)
  - 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

Segédosztályok: *Random*

- Véletlenszám előállítása
  - konstruktorok
    - *default*: automatikus seed-elés
    - *seeded* (param *long*), determinisztikus *(setSeed)*
  - *nextXXX()*
    - egyenletes eloszlás
    - *boolean*, *bytes*, *int*, *long*: eredmény az adott típus tartományán
    - *double*, *float*: eredmény a [0.0, 1.0) intervallumban
    - *nextInt(int n)*: eredmény *0* és *n* között (jobbról nyílt)
  - *nextGaussian()*
    - normális eloszlás (várható érték: *0*, szórás: *1*)

Segédosztályok: *Arrays*

- Tömbök kényelmes kezelése
  - *sort*: rendezés
  - *binarySearch*: keresés rendezett tömbben
  - *fill*: feltöltés adott értékkel
  - *equals*: azonosság
  - *copyOf*: nyújtás-zsugorítás
  - *copyOfRange*: kivágás
  - *toString*: tartalom Stringgé konvertálása