

## *Programozás III*

### KOLLEKCIÓK 2. ISMÉTLÉS

### FELADATMEGOLDÁS (ismétlés)

Gondoljuk végig egy klinika `latogato()` metódusát:

A betegek listája tartalmazza a regisztrált betegek adatait, keresünk egy konkrét beteget.

A programrészlet:

### KONTÉNEREK – GYŰJTEMÉNYEK (ismétlés)

A konténer olyan objektum, amely objektumokat tárol, és alkalmas különböző karbantartási, keresési és bejárési funkciók megvalósítására.

Csomagja: `java.util`

A konténerek általánosak, azokba bármilyen objektumot betehetünk. (De csak objektumot!)

A gyűjtemények „változtatható méretű tömbök”, rendelkeznek karbantartási és keresési funkciókkal.

### FELADATMEGOLDÁS (ismétlés)

```
private void latogato() {
    Paciens beteg;
    System.out.print("\nKi keresi? 1: orvos - 2: barát ");
    int ki = scan.nextInt();

    System.out.print("A keresett beteg neve: ");
    String nev = scan.next();
    if (ki == 2) {
        beteg = new Paciens(nev, "");
    } else {
        System.out.print("TAJ száma: ");
        String taj = scan.next();
        beteg = new Paciens(nev, taj);
    }

    if (betegek.contains(beteg)) {
        System.out.println("\nMegvan");
    } else {
        System.out.println("\nNincs ilyen beteg");
    }
}
```

### FELADATMEGOLDÁS (ismétlés)

Biztos, hogy megtalálja?

Csak akkor, ha a Paciens osztályban van **equals()** és **hashCode()** metódus.

És ezek a metódusok honnan tudják, hogy orvos vagy barát keresi az illetőt?

Egyelőre sehonnan, fel kell rá készíteni őket.

### FELADATMEGOLDÁS (ismétlés)

```
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    // if (getClass() != obj.getClass()) {
    //     return false;
    // }
    final Paciens other = (Paciens) obj;
    if (!Objects.equals(this.nev, other.nev)) {
        return false;
    }
    if (hasonlitasSzempont == ORVOS) {
        if (!Objects.equals(this.tajSzam, other.tajSzam)) {
            return false;
        }
    }
    return true;
}
```

Mikor nem kell a kikommentezett rész?

### FELADATMEGOLDÁS (ismétlés)

```
public class Paciens {
    private String nev, tajSzam;
    private static int hasonlitasSzempont;
    public static final int ORVOS = 1;
    public static final int BARAT = 2;

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 97 * hash + Objects.hashCode(this.nev);
        if (hasonlitasSzempont == ORVOS) {
            hash = 97 * hash + Objects.hashCode(this.tajSzam);
        }
        return hash;
    }
}
```

```
private void latogato() {
    Paciens beteg;
    System.out.print("\nKi keresi? 1: orvos - 2: barát ");
    int ki = scan.nextInt();

    System.out.print("A keresett beteg neve: ");
    String nev = scan.next();
    if (ki == 2) {
        beteg = new Paciens(nev, "");
        Paciens.setHasonlitasSzempont(Paciens.BARAT);
    } else {
        System.out.print("TAJ száma: ");
        String taj = scan.next();
        beteg = new Paciens(nev, taj);
        Paciens.setHasonlitasSzempont(Paciens.ORVOS);
    }
    if (betegek.contains(beteg)) {
        System.out.println("\nMegvan");
    } else {
        System.out.println("\nNincs ilyen beteg");
    }
}
```

### FELADATMEGOLDÁS (ismétlés)

Mi történik, ha a Paciens osztálynak van egy utód osztálya, és az utódok „ugyanolyanságára” még egy további saját jogú feltétel is van?

A név-re, tajszámra vonatkozó feltétel öröklődik a Paciens osztályból, a saját feltételre vonatkozó megkötést viszont az utód osztályban kell generálni.

Mi a különbség az equals és a == között?

### RENDEZÉS (ismétlés)

Mikor használható a Collections.sort(adatok) hívás?

adatok: List<Adat> adatok;

Adat: class Adat implements Comparable

Adat-ban: compareTo() metódus

### RENDEZÉS (ismétlés)

A Collections osztály algoritmusai úgy működnek, hogy páronként összehasonlítják a gyűjtemény elemeit. Ezért ezek a metódusok megkövetelik, hogy **a konténerbe betett objektumok összehasonlíthatóak legyenek**, vagyis hogy

a/ maguk implementálják a Comparable interfészt, vagy

b/ létezzen hozzájuk a Comparator interfészt implementáló hasonlító osztály.

### RENDEZÉS (ismétlés)

Mikor használható a Collections.sort(adatok, new Hasonlitas()) hívás?

adatok: List<Adat> adatok;

Adat: class Adat

Hasonlitas: class Hasonlitas implements Comparator

Hasonlitas-ban: compare() metódus

## RENDEZÉS (ismétlés)

Rendezések:

Mindegy, hogy melyik fajtát használják, de egy projekten belül lehetőleg csak egyfajta legyen.

(Az AlapOszty implementa Comparable megoldás esetén is lehet többféle szempontú rendezés – hogyan?)

## PÉLDÁK

Hogyan lehetne rendezni a jubileumi feladat résztvevőit a PTE azonosító alapján?

```
public class AzonositoSzerint implements Comparator<Resztvevo>{  
  
    @Override  
    public int compare(Resztvevo o1, Resztvevo o2) {  
        String o1Azonosito, o2Azonosito;  
        o1Azonosito = o1 instanceof PTEsResztvevo ?  
            ((PTEsResztvevo)o1).getPteAzonosito() : "";  
        o2Azonosito = o2 instanceof PTEsResztvevo ?  
            ((PTEsResztvevo)o2).getPteAzonosito() : "";  
  
        return o1Azonosito.compareTo(o2Azonosito);  
    }  
}
```

## PÉLDÁK

Hogyan lehet rendezni Teglalap típusú objektumokat terület() szerint?

```
public class Teglalap implements Comparable<Teglalap>{  
    private double szelesseg, magassag;  
  
    public Teglalap(double szelesseg, double magassag) {  
        this.szelesseg = szelesseg;  
        this.magassag = magassag;  
    }  
  
    public double terület(){  
        return szelesseg*magassag;  
    }  
  
    @Override  
    public int compareTo(Teglalap t) {  
        return (int) Math.signum(this.terület() - t.terület());  
    }  
}
```

## PÉLDÁK

Egy listában Sportolo típusú példányok vannak, konkrétan Ugro és Futo objektumok. Mindegyiknek van egy

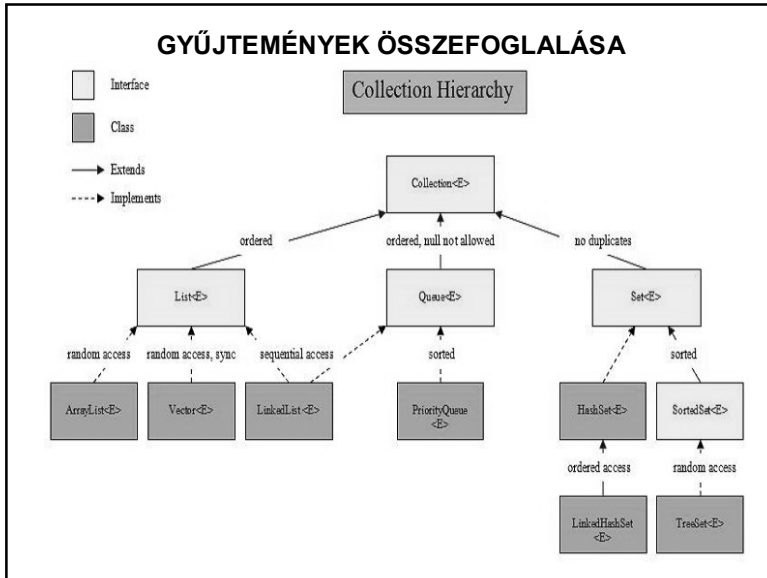
int teljesitmeny(){...} értéke.

Ez futók esetén az időeredmény (sec), ugróknál az ugrott magasság (cm).

Hogyan rendezhetjük teljesítmény szerint a sportolókat?

Együtt sehogy, nem összemérhető adatok.

Maga az öröklődés is hibás: csak formailag hasonló az utódok teljesitmeny() metódusa, logikailag egyáltalán nem, így nem is szabadna örökíteni.



### NÉHÁNY LINK

<http://java-latte.blogspot.hu/2013/11/object-equality-in-java.html>  
<http://java-latte.blogspot.hu/2014/02/comparable-and-comparator-interfaces-in.html>  
<http://java-latte.blogspot.hu/2013/06/dont-know-which-mapcollection-to-use.html>  
[https://www.tutorialspoint.com/java/java\\_treeset\\_class.htm](https://www.tutorialspoint.com/java/java_treeset_class.htm)  
<http://java-latte.blogspot.hu/2013/09/java-collection-arraylistvectorlinkedli.html>

Kicsit más jellegű ebből a blogból:  
<http://java-latte.blogspot.hu/2013/08/stack-and-heap-in-java.html>

