## Comparable

A Javás rendezések egyik alapja `Comparable` interface. A `Comparable` interface-t megvalósító osztályok nyilatkozni tudnak arról, hogy milyen szempont alapján hasonlíthatók össze egymással. Másképpen mondva: a `Comparable` által deklarált `int compareTo(Object masik)` metódus visszatérési értéke alapján az adott objektum megmondja, hogy ő van-e a sorrendezésben előbb (negatív visszatérési érték), ő van később a sorrendezésben (pozitív visszatérési érték), vagy egyformák (0 visszatérési érték).

Nézzünk egy példát! A `Diak` objektumok születési év szerinti sorrendjét adjuk meg `compareTo` segítségével.

```
public class Diak implements Comparable {
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Object m) {
        Diak masik = (Diak)m;
        if (szulEv < masik.szulEv) {
            return -1;
        } else if (szulEv == masik.szulEv) {
            return 0;
        } else {
            return +1;
        }
    }
}
```

A fenti `compareTo` esetén nem lehetünk biztosak abban, hogy a `compareTo` `Diak` objektumot kap paraméterül. Lehet, hogy valami mást... Ekkor tutira futás idejű hibát kapunk, mégpedig `ClassCastException`-t.

Javítsuk meg, érjük el, hogy a `compareTo` csak másik `Diak`-ra működjön:

```
public class Diak implements Comparable<Diak> { // <--- itt módosult (A kacsacsőrök között egy osz
tálynév szerepel, amit a compareTo paramétertípusaként fogunk használni. Bővebben a Generics témáb
an találhatsz erről infót).
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Diak masik) { // <--- meg itt módosult
        if (szulEv < masik.szulEv) {
            return -1;
        } else if (szulEv == masik.szulEv) {
            return 0;
        } else {
            return +1;
        }
    }
}
```

Tipp. Vizsgáljuk meg, hogy a fenti specifikációnak megfelel-e a `szulEv - masik.szulEv` képlet:

- ha `szulEv < masik.szulEv`, akkor negatív számot kell visszaadni, `szulEv - masik.szulEv` pont negatív.
- ha `szulEv == masik.szulEv`, akkor 0-t kell visszaadni, `szulEv - masik.szulEv` pont nulla.
- ha `szulEv > masik.szulEv`, akkor pozitív számot kell visszaadni, `szulEv - masik.szulEv` pont pozitív.

Magyarul ez a trükk pont jó:

```
public class Diak implements Comparable<Diak> {
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Diak masik) {
        return szulEv - masik.szulEv;
    }
}
```

## String összehasonlítása

Ha úgy döntünk, hogy inkább név szerinti sorrendet szeretnénk, akkor:

```
public class Diak implements Comparable<Diak> {
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Diak masik) {
        return nev.compareTo(masik.nev);
    }
}
```

... azaz visszavezettük a `String` objektumok összehasonlítására a `Diak` objektumok összehasonlítását.

## Double számok összehasonlítása

A tanfolyam elején tanultuk, hogy a double számok értékei kis mértékben módosulhatnak a műveletek hatására, eltérhetnek a matematikai értéktől. A megoldás az, hogy bevezetünk egy minimális különbséget, aminek megléte esetén még egyenlőséget állapítunk meg:

```
public class Diak implements Comparable<Diak> {
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Diak masik) {
        double eps = 1e-10; // 1..5 közötti számoknál teljesen jó.
        if (Math.abs(this.atlag - masik.atlag) < eps) {
            return 0;
        } else if (this.atlag < masik.atlag) {
            return -1;
        } else {
            return +1;
        }
    }
}
```

...vagy használhatjuk a `Double` osztályt, ami egyetlen `double` értéket tartalmaz, de objektumként, és vannak metódusai, mint a `compareTo`:

```
public class Diak implements Comparable<Diak> {
    private String nev;
    private int szulEv;
    private double atlag;

    ...

    @Override
    public int compareTo(Diak masik) {
        Double a = new Double(atlag); // rövidíthető: Double a = atlag;
        Double b = new Double(masik.atlag); // rövidíthető: Double b = masik.atlag;
        return a.compareTo(b);
    }
}
```

Hogy mire használhatók a `compareTo` metódusok, azt a következő alfejezetben nézzük meg.