# 11. Interface-ek 1.

## Az interface

Az **osztályok** a természetes objektumokat modellezik.

Az **adattagok** modellezik az objektumok tulajdonságait, állapotát, míg a **metódusok** az objektum viselkedését.

Például egy Kutya **osztály** modellezhet egy kutya objektumot,

az osztály **adatai** lehetnek pl. név, fajta, szín, stb.;

**viselkedése** pedig lehet, hogy elriasztja az idegeneket, játszik a gazdival, stb.

Az **interface**-k ezzel szemben **viselkedés** csoportokat modelleznek.

Például a valós kutya objektumról tudjuk, hogy házőrző. De mit is jelent az hogy házőrző? Ezt definiálhatjuk, modellezhetjük egy interface-el. Például a Házőrző interface modellezheti azt, ha van valaminek olyan két viselkedése, hogy képes elriasztani az idegeneket és hogy jelez ha idegen közeledik.

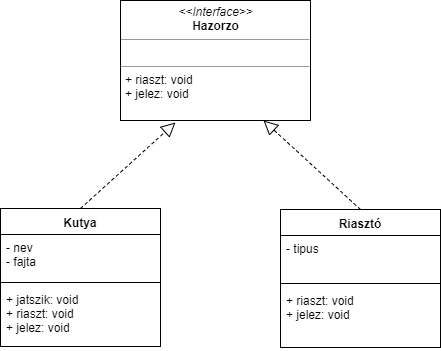
Egy osztály megvalósíthat egy (vagy több) interface-t, ha az interface-ben megadott viselkedésekkel rendelkezik.

A Kutya megvalósítja a Házőrző viselkedést, de természetesen nem csak egy kutya rendelkezhet ezekkel a viselkedésekkel, hanem például egy Riasztó is. Azaz ugyanazt az interface-t több osztály is megvalósíthatja.

Amikor tehát interface-t definiálunk, akkor megadjuk, hogy egy adott név alatt milyen viselkedéseket értünk.

A házőrző interface két funkciót tartalmaz, az idegenek elriasztását és az idegenek jelzését. Ha egy osztály rendelkezik ezekkel a funkciókkal, akkor jelezhetjük, hogy az osztály megvalósítja az interface-t.

Ha olyan osztályt akarunk készíteni, amely megvalósít egy interface-t, akkor kötelesek vagyunk az osztályban definiálni az interface-ben megadott viselkedéseket (metódusokkal).



Mire lehet jó, hogy definiálunk interface-eket és kijelentjük, hogy az osztály megvalósítja azt?

Ha nem neveztük volna el ezt a két viselkedést házőrzőnek, attól még rendelkezett volna vele a kutya és a riasztó is. A lényeg az, hogy ezek után képesek vagyunk egyes funkciókat, viselkedéseket egy interface meglétéhez kötni.

Például definiálhatunk egy funkciót, amely nem valamilyen konkrét objektummal képes működni, hanem bármivel, ami megvalósítja a házőrző interface-t. Ilyen funkció lehet például a riasztás bekapcsolása.

Egy másik osztályban írhatjuk ezt:

private void bekapcsol(Hazorzo valami) { valami.riaszt; }

Ez kutyával vagy riasztóval is működik.

## Az interface tagjai

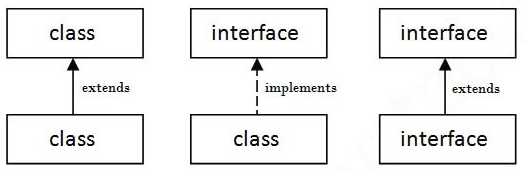
Az interface-k viselkedéseket tartalmaznak, amelyeket metódusok modelleznek. Viszont az interface számára lényegtelen a metódus törzse. Az interface-nek csak azt kell definiálnia, hogy mely viselkedésekből áll az interface, azt nem, hogy ezek a viselkedések hogyan működjenek.

A házőrző definiálja, hogy legyen jelzés funkció, de azt nem, hogy hogyan jelezzen. A kutya megvalósíthatja ugatással, a riasztó megvalósíthatja egy központi terminálra kiírt jelzéssel. Tehát az interface-ek tagjai absztrakt metódusok.

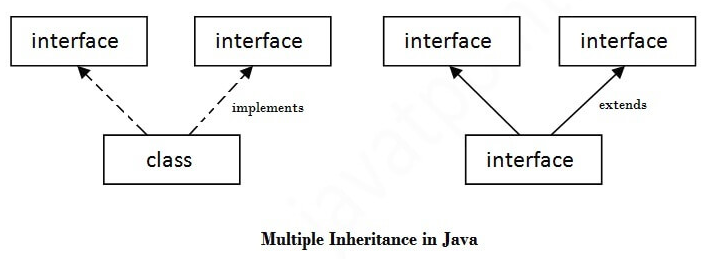
Ezen kívül állandók (konstansok) is szerepelhetnek az interface-ekben.

## Öröklődés

Interface-ekre is értelmezett az öröklődés, azaz lehet úgy definiálni egy interface-t, hogy az egy másik interface gyermeke.



Sőt, az interface-kre létezik a többszörös öröklődés is, azaz egy interface-nek létezhet több szülője is. Az interface-eknél ennek az a szerepe, hogy ha egy viselkedéscsoport több másikból tevődik össze, akkor lehetőségünk legyen ezt megadni.



Például ha az A interface az a1 és az a2 viselkedést tartalmazza, míg egy B interface a b1-et, és definiálunk egy C interface-t, amely gyermeke A-nak és B-nek, akkor a C interface tartalmazza az a1, az a2, és a b1 viselkedéseket, anélkül, hogy azokat még egyszer definiálni kellene.

## Az interface-ek hasznossága

A Java API készítői is sok helyen használnak interface-eket.

Például készítettek olyan metódust (Arrays.sort), amellyel egy tömb elemeit lehet rendezni. Ha a tömb elemi objektumok, akkor a metódus nem tudja, hogyan hasonlítsa össze őket.

Ezért készítettek egy Comparable nevű interface-t, amelyben egy compareTo nevű absztrakt metódus van, amely összehasonlít két objektumot.

Amelyik osztály megvalósítja a Comparable interface-t, annak tömbjeit már tudja rendezni az Arrays.sort() metódus.

## Az interface, mint típus

Az interface és az osztály is típusnak számít a Java nyelvben.

Egy forrás file-ban egy publikus típus (osztály vagy interface) definíció lehet, és tetszőleges számú nem publikus.

Az interface-ket is szokás külön fájlokban definiálni, csakúgy mint az osztályokat, és a fájl neve megegyezik a benne definiált interface nevével.

A forrásfájl szerkezetével kapcsolatban annyi a különbség az osztályokhoz képest, hogy a csomagmegadás és az import deklarációk után nem osztálydefiníciók következnek, hanem az interface definíciója.

## Az interface definíció szintaktikája

módosítók interface azonosító extends szülőinterface-ek {  
 tagok definíciói  
}

A módosítók a szokásos hozzáférési módosítók lehetnek (public, …).

A class kulcsszó helyett az interface kulcsszó szerepel.

Az extends részben adhatjuk meg a szülő interface-ket. Egy interface-nek tetszőleges számú szülője lehet, de csak interface. Az extends rész elmaradhat.

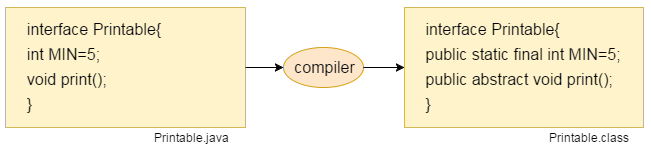
## A tagok definíciói

A Java nyelv 8-as verziója előtt csak kétféle tagja lehetett egy interface-nek. Most csak ezt a kettőt nézzük meg.

* absztrakt metódus
* konstans (állandó)

Az absztrakt metódus automatikusan public abstract módosítójú, anélkül, hogy kiírnánk, de akár ki is írhatjuk. Ezeket a metódusokat fogja az interface-t megvalósító osztály felüldefiniálni. Ezért nem lehet olyan módosítója, amely ezt kizárja, azaz nem lehet final vagy static.

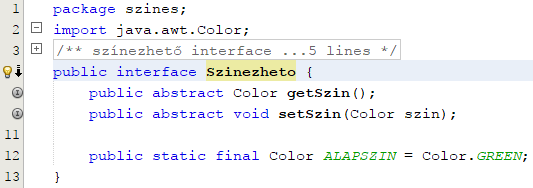
A konstans automatikusan public static final módosítójú, más módosítója nem lehet. Ezeket kiírhatjuk, de nem kötelező.



## Feladat

A mai feladatban egy interface-t készítünk Szinezheto néven, amely két metódust és egy állandót definiál.

## Szines interface

1. Indítsd el a NetBeans programot, és hozz létre egy új projektet szines néven!
2. Hozz létre egy interface-t a szines csomagon belül, Szinezheto néven! (New, Java Interface)
3. Készítsd el a Szinezheto interface kódját az alábbiak szerint:  
     
   Importálni kell a java.awt.Color osztályt. A getSzin() metódus egy objektum színének lekérdezésére, a setSzin() metódus pedig a beállítására való. Az ALAPSZIN értéke a zöld szín.  
   A 9-10. sorban az abstract szó elhagyható.

A következő leckében folytatjuk.