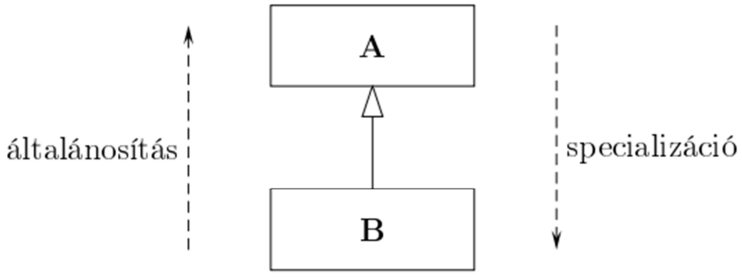
Programozási nyelvek Java

5. gyakorlat

Általánosítás és specializáció (öröklődés)

- Az általános tulajdonságokkal rendelkező dolgok (superclass, ősosztály) és a kevésbé általános, speciálisabb dolgok (subclass, származtatott osztály) között fennálló reláció
- Először létrehozzuk az általános tulajdonságokkal felruházott osztályt, majd annak tulajdonságait átvéve származtatjuk a speciális tulajdonságokkal rendelkező osztályt
- Osztályok hierarchiába rendezhetőek a fenti elv alapján
- A kód újrafelhasználását segíti elő (jó, mert nem szeretjük ismételni a kódot)
- Java a C++-szal ellentétben csak az úgynevezett egyszeres öröklődést engedélyezi, azaz minden osztálynak csak egyetlen közvetlen szülője lehet.



Specializáció

- Az altípusképzés támogatja a nyelvben (programtervezési tervezési eszköz)
- Java-ban altípusos polimorfizmus van
    - a leszármazott rendelkezik a szülő metódusaival, adattagjaival
    - a leszármazott típusú referencia mutathat szülő típusú objektumra
    - statikus típus: a deklarált típus
    - dinamikus típus: a referált objektum aktuális típusa (amit odaadtunk neki valójában)
- Ha egy osztályt **final** kulcsszóval látunk el, akkor belőle nem lehet örökölteti (megakadályozható, hogy kiegészítsék, megváltoztathassák az adott osztály tulajdonságait)
- @Override annotáció: explicit jelzés a metódus felüldefiniálására
  Főként a következőket szokás felüldefiniálni:
    - Object.equals()
    - Object.hashCode()

- Object.toString()

```
//A Child osztályt származtatom a Parent osztályból
public class Child extends Parent {

        /* Child törzse */

}
```

Túlterhelés (overloading), felüldefiniálás (overriding)

A felüldefiniálás során elfedjük a leszármazott osztályban az ősosztály azonos nevű metódusát, melynek azonos a paraméter-típuslistája és azonos a visszatérési értéke is. Ha ez a három kritérium egyszerre nem teljesül, akkor csupán bővítjük az ősosztályt egy újabb (túlterhelt) metódussal.

A felüldefiniálás nem lesz sikeres, ha a leszármazott osztályban a metódus más ellenőrzött kivételt vált ki, illetve ha a hozzáférési kategóriája szűkebb, mint az ősosztály azonos metódusa. Ez utóbbi esetekben a fordító hibát fog jelezni.

A felüldefiniálás a gyerek osztályban letiltható, ha a függvényt **final** kulcsszóval látjuk el.

Példa (Programozzuk is le közösen!!!)

Készítsünk programot, amellyel a testek térfogatát határozhatjuk meg, és emellett tároljuk el, hogy az egyes testfajtákból hány darab objektum létezik. A feladatot öröklődés felhasználásával oldjuk meg.

Fajták

- Szabályos sokszögek: gömb, kocka, tetraéder, oktaéder
- Hasábjellegű testek: henger, négyzet alapú hasáb, szabályos háromszög alapú hasáb
- Gúla jellegű testek (speciális hasáb jellegűek): kúp, négyzet alapú gúla

Az absztrakt **Test** osztály adja meg a testek közös jellemzőit. Ebből származtassuk az absztrakt **Szabalyos**, az absztrakt **Hasab** osztályokat.

A **Szabalyos** osztály konkrét esetei: **Gomb**, **Kocka**, **Tetraeder**, **Oktaeder**.

A **Hasab** osztály konkrét esete: **Henger**, **Negyzetes** (négyzet alapú hasábok), **Haromszoges** (szabáylos háromszög alapú hasáb). Ezen kívül tartalmaz egy speciális esetet: absztrakt **Gula** osztályt (A területet 1/3-dal kell megszorozni).

A **Gula** osztály konkrét esete: **Kup**, **NGula** (négyzet alapú gúla).

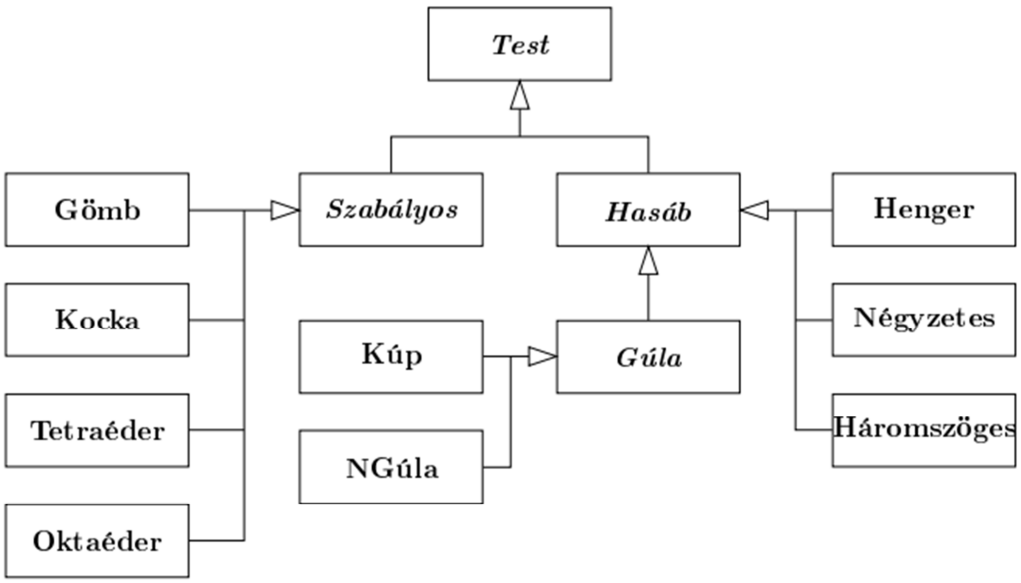
Minden test jellemzője a *méret*, ami szabályos sokszögek esetén a sugarat illetve az oldal hosszát jelenti. Hasábok esetében az alap egy szabályos sokszög (aminek az adatai adottak), így csak a magasságot kell eltárolni.

Továbbá minden osztályban nyílván kell tartani az osztály objektumainak a számát.

A közös művelet a térfogat kiszámítására szolgáló *terfogat* függvény és darabszámot visszaadó *darab* függvény.

Az objektumok számát az egyes konstruktorokban növelni kell. Egy származtatott osztályban explicit meg kell hívni az ősosztály konstruktorát is (ez növeli az ősosztály darabszámát is).

Osztálydiagram



Feladatok

1. Valósítsátok meg a következő feladatot.
   Adottak személyek (**Person**), akik vagy fiúk (**Boy**) vagy lányok (**Girl**) lehetnek. Minden személynek van neve (**name**), születési ideje (**bornDate**), születési helye (**bornPlace**). A lányoknak ezeken a tulajdonságokon kívül van még leánykori neve (**maidenName**) és mellmérete (**boobsSize**), a fiúknak pedig kopaszodási tulajdonságuk (0-1 közötti valós szám) (**baldAttr**).
   Készítsétek el minden osztály minden adattagjához a getter, setter metódusokat.
   Ezek után készítsetek hozzá egy tesztfájlt.
2. Készítsetek az **accessories** csomagba egy **Shoes** és egy **Dress** osztályt. A Shoes osztálynak 2 adattagja legyen egy méret és egy ár. A Dress osztálynak csak ára legyen.
3. Írjátok át a honlapon levő **Person** osztályt (4. gyakorlat – 3. feladat), hogy az embernek legyen pénze, cipőmérete és képes legyen cipőt és ruhát is venni. Ehhez szükség van néhány új adattagra és 2 új függvényre is (buyShoes, buyDress). Csak akkor tudjon bármit is venni, ha még nincs neki és van rá elég pénze. Cipő esetében a méretnek is meg kell egyeznie.
4. Írjatok a Person osztályba egy **invite** függvényt, aminek paraméterében át kell adni egy másik személyt. A függvény térjen vissza egy logikai változóval, hogy sikeres volt-e a felkérés. Az azonos nemű partnereket nem engedélyezzük. Egy fiúnak csak akkor mond a lány igent, ha gazdag, azaz több pénze van. Továbbá tároljátok el a partnert is az osztályban. Akinek már van partnere, ő már ne senkit sem elhívni és senkivel sem elmenni.
5. Végül írjatok egy **goToParty** függvényt. Egy személy csak akkor tud elmenni egy party-ra, ha van ruhája, cipője és partnere, továbbá a partnerének is van ruhája és cipője.