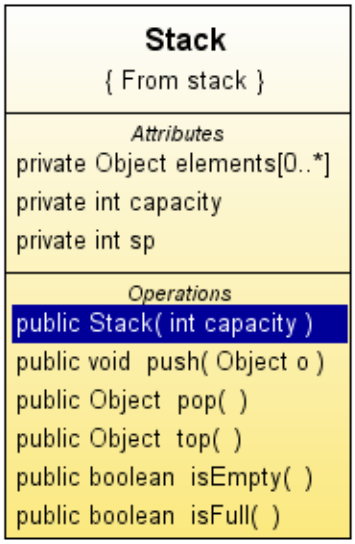
OBJEKTUMORIENTÁLT PROGRAMOZÁS 2008.
6. GYAKORLAT

CÉL:

- adatstruktúrák típusfüggetlen implementációja
- kód újrafelhasználása - Dinamikus tömb használata - **java.util.Vector**
- *származtatás* és *tartalmazási kapcsolat* közötti hasonlóságok és különbségek

1. Feladat – típusfüggetlen tároló

Készítse a *verem* adatstruktúra típusfüggetlen implementációját. Használja az alábbi diagramot.



Tesztelje az osztályt:

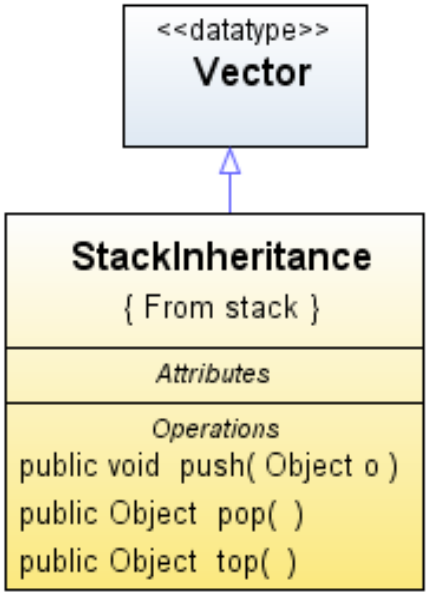
- Hozzon létre egy 11 kapacitású vermet.
- Töltse fel 0 és 10 közötti egész számokkal.
- Ürítse ki a vermet, minden lépésben kiíratva az éppen kivett elemet.
- Töltse fel ugyanazt a vermet karakterláncokkal. Ezek legyenek a következők: *STR0, STR1, ..., STR10*
- Ürítse ki a vermet, minden lépésben kiíratva az éppen kivett elemet.

OBJEKTUMORIENTÁLT PROGRAMOZÁS 2008.
6. GYAKORLAT

---

2. Feladat - Származtatás

Implementálja a verem adatstruktúrát származtatást használva. Ebben a változatban nem használunk kapacitást. Ősosztályként használja a `java.util.Vector` dinamikus tömb osztályt.



A Vector osztályból a következő metódusokat használhatja:

```
void add( Object o )
Object elementAt( int index )
Object remove( int index)
int size()
boolean isEmpty()
```

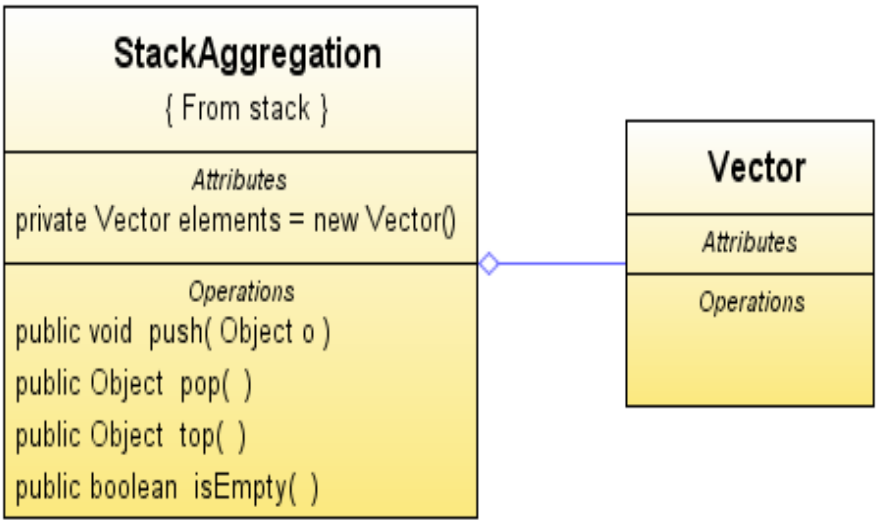
Az osztály tesztelésére használja az előző feladatban megadott adatokat.

OBJEKTUMORIENTÁLT PROGRAMOZÁS 2008.
6. GYAKORLAT

---

3. Feladat – Tartalmazás

Implementálja a verem adatstruktúrát tartalmazási kapcsolatot használva. Az elemek tárolására használjon dinamikus tömböt (`java.util.Vector`).



Az osztály tesztelésére használja az előző feladatban megadott adatokat.

Ellenőrizze a következő kódrészletek helyességét:

1. kódrészlet

```
StackInheritance s1 = new StackInheritance();
for( int i =0; i<15; ++i )
    s1.push("Name"+(i+1));
System.out.println( s1.get(9));
```

2. kódrészlet

```
StackAggregation s1 = new StackAggregation();
for( int i =0; i<15; ++i )
    s1.push("Name"+(i+1));
System.out.println( s1.get(9));
```

Melyik megoldás garantálja a CSAK veremszerű működést?