A rendelkezésre álló idő:18 **perc**

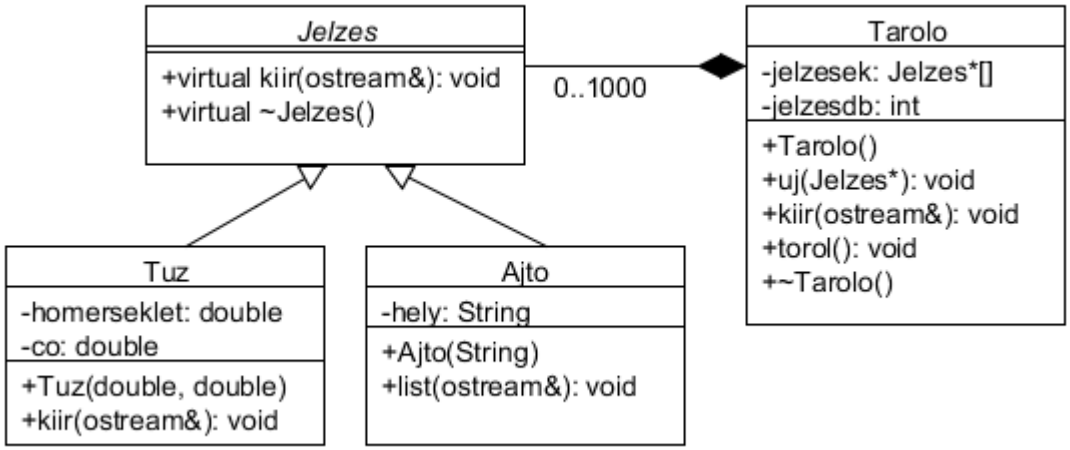
Programozás alapja 2. 3. ellenőrző dolgozat. 2016.04.22. Kurz/**Terem**: G5/	Elért pontszám:
Név: **Neptun:**	

Egy biztonsági rendszerben különféle érzékelők jelzéseit (*Jelzes*) kell fogadni és eltárolni. Az érzékelők teljesen eltérő adatokat szolgáltatnak. A tűzjelző (*Tuz*) pl. hőfokot és CO koncentrációt (2 db *double*), az ajtónyitás érzékelő (*Ajto*) pedig egy szöveget (*String*) ad, ami az érzékelő helyét azonosítja. A rendszert később bővíteni fogjuk újabb, jelenleg még nem ismert érzékelőkkel. A beérkező jelzéseket a beérkezés sorrendjében, egyetlen tárolóban (*Tarolo*) kell eltárolni. A letárolt jelzéseket naponta egy *std::ostream* típusú objektumra kiírják (*kiir*), majd törlik (*torol*). Egy nap maximum 1000 jelzés keletkezik. A rendszerben megvalósítandó műveleteket egy kódrészlettel mutatjuk be:

```
Tarolo naplo;                    // Ez lesz a tároló
naplo.uj(new Tuz(198, 0.55));    // Tűzjelző jelzését tárba tesszük
naplo.uj(new Ajto("IB28"));      // Ajtónyitás érzékelő jelzését tárba tesszük
naplo.kiir(std::cout);           // Tárolóban levő jelzések és azok adatainak kiírása
naplo.torol();                   // Összes jelzés törlése a nap végén
naplo.uj(new Ajto("IB26"));      // Ajtónyitás érzékelő jelzését tárba tesszük
```

1. feladat 3 p

Tervezzen meg és osztálydiagramon ábrázoljon egy olyan objektummodellt, ami alkalmas a biztonsági rendszer adatainak keletkezési sorrendben való tárolására és könnyen bővíthető!



2. feladat 7 p

Deklarálja a *Tarolo*, *Jelzes*, és *Ajto* osztályokat! Az adattagok privát láthatóságúak legyenek! Feltételezheti, hogy a ***Tarolo*** osztályból példányosított objektumot nem akarjuk paraméterként átadni és értékadás jobb, ill. bal oldalán sem szerepeltetjük. A laborgyakorlaton elkészített *String* osztály rendelkezésére áll, azt nem kell deklarálnia!
Valósítsa meg a *Tarolo* osztály tagfüggvényeit! Ne használjon STL elemeket!

```
class Jelzes {
public:
      virtual void kiir(std::ostream&);
      virtual ~Jelzes();
};

class Ajto : public Jelzes {
      String hely;
public:
      Ajto(String);
      void kiir(std::ostream&);
};
```

```
class Tarolo {
      Jelzes* jelzesek[1000];
      int db;
public:
      Tarolo() : db(0) {}
      void uj(Jelzes* a) {
            jelzesek[db++] = a;
      }
      void kiir(std::ostream& os) {
            for (int i = 0; i < db; i++)
                  jelzesek[i]->kiir(os);
      }
      void torol() {
            for (int i = 0; i < db; i++)
                  delete jelzesek[i];
            db = 0;
      }
      ~Tarolo() { torol(); }
};
```

BBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 3. ellenőrző dolgozat. 2016.04.22. Kurz/**Terem**: G5/	Elért pontszám:
Név: **Neptun:**	

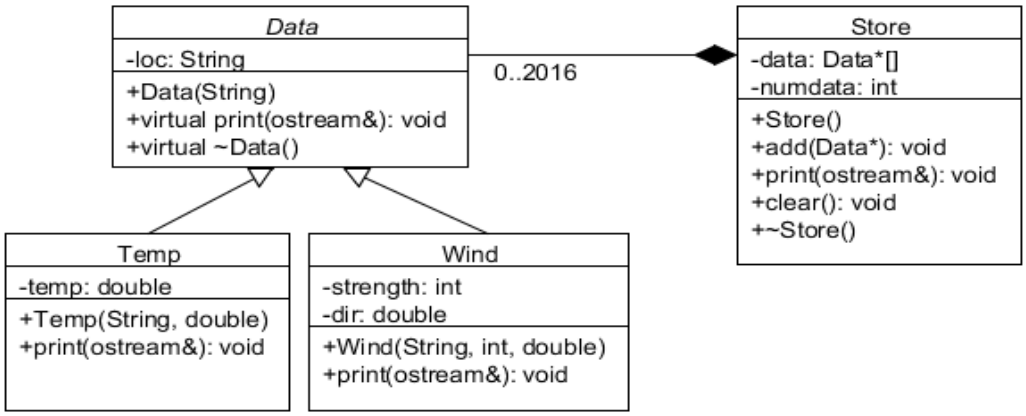
1. feladat

Egy meteorológiai rendszerben különféle érzékelők adatait (*Data*) kell fogadni és eltárolni. Az érzékelőkben közös, hogy a címüket tárolják (*String*), de ezen kívül teljesen eltérő adatokat adnak. A hőmérő (*Temp*) pl. hőfokot (*double*), az Szélmérő (*Wind*) pedig egy szélerősséget (int) és a szélirányt (*double*) szolgáltatja. A rendszert később bővíteni fogjuk újabb, jelenleg még nem működő érzékelőkkel (pl. légnyomásmérő). A beérkező adatokat a beérkezés sorrendjében, egyetlen tárolóban (*Store*) kell eltárolni. A tárolóban levő adatokat (azok attribútumait is) naponta egy *std::ostream* típusú objektumra kiírják (*print*), majd törlik (*clear*). Egy nap maximum 2016 adat keletkezik. A rendszerben megvalósítandó műveleteket egy kódrészlettel mutatjuk be:

```
Store st;                                  // Ez lesz a tároló
st.add(new Temp("BME I", 23.5));           // Hőmérsékleti adat tárolása
st.add(new Wind("BME St", 6, 127.3));      // Szélerősség és irány tárolása
st.print(std::cout);                       // Tárolóban levő adatok kiírása
st.clear();                                // Összes adat törlése a nap végén
st.add(new Temp("BME Ch", 19.2));          // Hőmérsékleti adat hozzáadása
```

1. feladat 3 p

Tervezzen meg és rajzoljon le egy olyan objektummodellt, ami alkalmas a biztonsági rendszer adatainak keletkezési sorrendben való tárolására és könnyen bővíthető! Ne használjon STL elemeket!



2. feladat 7 p

Deklarálja a *Store*, *Data*, és *Temp* osztályokat! Az adattagok privát láthatóságúak legyenek! Feltételezheti, hogy a ***Store*** osztályból példányosított objektumot nem akarjuk paraméterként átadni és értékadás jobb, ill. bal oldalán sem szerepeltetjük. A laborgyakorlaton elkészített String osztály rendelkezésére áll, azt nem kell deklarálnia!

Valósítsa meg a *Store* osztály tagfüggvényeit! Ne használjon STL elemeket!

```
class Data {
      String loc;
public:
      Data(String l):loc(l){}
      virtual void print(std::ostream&);
      virtual ~Data();
};

class Temp : public Data {
      double temp;
public:
      Temp(String l, double t):
            Data(l), temp(t){};
      void print(std::ostream&);
};
```

```
class Store {
      Data* data[2016];
      int numdata;
public:
      Store() : numdata(0) {}
      void add(Data* a) {
            data[numdata++] = a;
      }
      void print(std::ostream& os) {
            for (int i = 0; i < numdata; i++)
                  data[i]->print(os);
      }
      void clear() {
            for (int i = 0; i < numdata; i++)
                  delete data[i];
            numdata = 0;
      }
      ~Store() { clear(); }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 3. ellenőrző dolgozat. 2016.04.22. Kurz/**Terem**: G5/	Elért pontszám:
Név: **Neptun:**	