AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő: 20 **perc**

Programozás alapja 2. 5. ellenőrző dolgozat. 2015.05.06. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

1. Az *Adat* osztály egy fizikai kísérlet adatait tárolja. Az osztály belső szerkezetét nem ismerjük, de minden tárolt adata beállítható a konstruktorával, valamint lekérdezhető és beállítható a *getter/setter* függvényekkel, melyek interfészét pontosan ismerjük:

Függvény	**Funkció**
`void Adat::setAzon(long);`	azonosító beállítása
`long Adat::getAzon();`	azonosító lekérdezése
`void Adat::setE(double);`	energia beállítása
`double Adat::getE()`	energia lekérdezése
`Adat::Adat(long = 0, double = 0)`	konstruktor

Az osztálynak ezen kívül számos publikus tagfüggvénye is van, melyek pontos funkcióját/leírását a fizikus kollegák ismerik, mi nem. Azt tudjuk, hogy az osztály nem perzisztens és nincs read/write tagfüggvénye. **Hozzon** létre az *Adat* osztályból egy perzisztens viselkedésű *PAdat* osztályt! A perzisztens változatnak minden funkciójában meg kell felelnie az *Adat* osztálynak. **Deklarálja** és **valósítsa** meg a *PAdat* osztályt úgy, hogy az alábbi programrészlet az elvárásoknak megfelelően működjön.

```
PAdat a1, a2(2), a3(3, 4.5);
std::cout << a3.getE();
std::stringstream ss;
a2.write(ss);      // adat kiírása a stream-re
a1.read(ss);       // adat visszaolvasása a stream-ről
```

Egy lehetséges megoldás

```
class PAdat : public Adat {
public:
    PAdat(long a = 0, double e = 0) :Adat(a, e) {}
    void read(std::istream& is) {
        long a;
        double e;
        (is >> a >> e).ignore(1);
        setAzon(a);
        setE(e);
    }
    void write(std::ostream& os) {
        os << getAzon() << ' ' << getE() << std::endl;
    }
};
```

BBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő: 20 **perc**

Programozás alapja 2. 5. ellenőrző dolgozat. 2015.05.06. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

1. A *Data* osztály egy fizikai kísérlet adatait tárolja. Az osztály belső szerkezetét nem ismerjük, de minden tárolt adata beállítható a konstruktorával, valamint lekérdezhető és beállítható a *getter/setter* függvényekkel, melyek interfészét pontosan ismerjük:

Függvény	**Funkció**
`void Data::setId(long);`	azonosító beállítása
`long Data::getId();`	azonosító lekérdezése
`void Data::setG(double);`	gyorsulás beállítása
`double Data::getG()`	gyorsulás lekérdezése
`Adat::Data(long = 0, double = 0)`	konstruktor

Az osztálynak ezen kívül számos publikus tagfüggvénye is van, melyek pontos funkcióját/leírását a fizikus kollegák ismerik, mi nem. Azt tudjuk, hogy az osztály nem perzisztens és nincs read/write tagfüggvénye. **Hozzon** létre a *Data* osztályból egy perzisztens viselkedésű *PData* osztályt! A perzisztens változatnak minden funkciójában meg kell felelnie a *Data* osztálynak. **Deklarálja** és **valósítsa** meg a *PData* osztályt úgy, hogy az alábbi programrészlet az elvárásoknak megfelelően működjön.

```
PData a1, a2(2), a3(3, 4.5);
std::cout << a3.getId();
std::stringstream ss;
a2.write(ss);     // adat kiírása a stream-re
a1.read(ss);      // adat visszaolvasása a stream-ről
```

Egy lehetséges megoldás

```
class PData : public Data {
public:
    PData(long id = 0, double g = 0) :Data(id, g) {}
    void read(std::istream& is) {
        long a;
        double g;
        (is >> a >> g).ignore(1);
        setId(a);
        setG(g);
    }
    void write(std::ostream& os) {
        os << getId() << ' ' << getG() << std::endl;
    }
};
```