Kutyamenhely

A feladatod az, hogy modellezz egy **kutyamenhelyet**. Készíts két osztályt: egy `Dog` és egy `Shelter` osztályt. A cél, hogy a program objektumorientált módon kezelje a kutyákat: lehessen őket hozzáadni, törölni, örökbe adni és listázni.

Dog osztály

A `Dog` osztály feladata egy kutya adatainak tárolása.
Minden kutyának van: egy egyedi azonosítója (ID), neve, kora és fajtája.
Az ID-t nem kell a konstruktorban megadni, az automatikusan és inkrementálisan jön létre (minden új kutya egy új ID-t kap).

Két kutya akkor tekinthető **azonosnak**, ha az ID-juk megegyezik.
Legyen lehetőség a kutya adatainak kiírására a képernyőre szép, áttekinthető formában (ehhez majd a << operátor túlterhelését kell használni).

Shelter osztály

A `Shelter` osztály képes több kutyát (`Dog` objektumot) tárolni.
A következő funkciókat kell megvalósítani:

- Új kutya hozzáadása a menhelyhez:
  - A + operátor segítségével lehessen új kutyát hozzáadni a menhelyhezpéldául: `shelter + d1;`.
  - Ha már létezik ugyanazzal az ID-val rendelkező kutya, akkor az új példányt **ne lehessen hozzáadni**, hanem dobj kivételt ebben az esetben.
- Kutyák törlése **adott ID** alapján:
  - A − operátor segítségével lehessen kitörölni egy kutyát a menhelyről az ID-ja alapján: `shelter - 1;`, vagyis törli az 1-es ID-jú kutyát
  - Ha nincs ilyen ID-jú kutya, szintén kivételt kell dobni.
- **Véletlenszerű kutya örökbeadása**:
  - Azaz a menhelyből egy random kutya kiválasztása és visszaadása.
  - A véletlen szám generáláshoz használhatod a következő kódrészletet:

```cpp
std::random_device rd;
std::mt19937 gen(rd());
std::uniform_int_distribution<int> dist(IDE_JON_AZ_ALSO_KORLAT,
    IDE_JON_A_FELSO_KORLAT);
int random_number = dist(gen);
```

- A menhelyen lévő kutyák **listázása**, vagyis a menhely kiírása:
  - A menhely teljes tartalma is kiírható legyen a képernyőre a << operátorral.
- **Indexelés (`operator[]`)**:
  - Legyen lehetőség az adott sorszámú kutya elérésére index alapján: például: `shelter[0]` az első kutyát adja vissza a menhelyről.
  - Biztosítsd, hogy ha az index érvénytelen (pl. nagyobb, mint a tárolt kutyák száma), akkor a program megfelelően kezelje a hibát (pl. `std::out_of_range` kivételt dobjon).

main.cpp teszteléshez

```cpp
#include <iostream>
#include <stdexcept>
#include "Dog.h"
#include "Shelter.h"

int main()
{
    Shelter shelter;
    // Uj kutyak letrehozasa
    Dog d1("Claus", "Bulldog", 3);
    Dog d2("Bejgli", "Beagle", 4);
    Dog d3("Rex", "Nemet juhasz", 2);
    // Hozzaadas
    std::cout << "Kutyak hozzaadasa a menhelyhez...\n";
    shelter + d1;
    shelter + d2;
    shelter + d3;
    std::cout << shelter << "\n";
    // Duplikalt hozzaadas - kivetelt kell dobjon
    std::cout << "Duplikált kutya hozzaadasa...\n";
    try
    {
        shelter + d2;
    }
    catch (const std::exception &ex)
    {
        std::cout << "Kivetel elkapva: " << ex.what() << "\n";
    }
```

```
29      // Indexeles
30      std::cout << "Elso kutya indexelessel: " << shelter[0] << "\n";
31      // Torles ID alapjan
32      std::cout << "Kutya torlése (ID=1)...\n";
33      shelter - 1;
34      std::cout << shelter << "\n";
35      // Veletlenszeru orokbeadas
36      std::cout << "Veletlenszeru orokbeadas...\n";
37      Dog adopted;
38      adopted = shelter.adoptRandom();
39      std::cout << "Orokbe adott kutya: " << adopted << "\n";
40      std::cout << "\nMenhely aktualis allapota:\n"
41                << shelter << "\n";
42      // Hibas indexeles
43      std::cout << "Hibas indexeles kiprobalasa...\n";
44      try
45      {
46          std::cout << shelter[10] << "\n";
47      }
48      catch (const std::out_of_range &ex)
49      {
50          std::cout << "Kivetel elkapva: " << ex.what() << "\n";
51      }
52      return 0;
53  }
```