Le type point en C++

Interface: point.hpp

```hpp
#include <ostream>
#ifndef POINT_H
#define POINT_H

class Point {
  static int count_; // Total amount of points
  const int rank_;
  double x_, y_;

  friend std::ostream&
  operator<<(std::ostream & out, Point const& p);

public:
  Point(double x, double y);
  ~Point() = default;

  double x() const;
  double y() const;
  void move(double dx, double dy);

  Point operator +(Point const& p2) const;
  bool operator == (Point const& other) const;
};

std::ostream&
operator<<(std::ostream & out, Point const& p);

#endif /* POINT_H */
```

Implémentation: point.cpp

```cpp
#include "point.hpp"

int Point::count_ = 0;

Point::Point(double x, double y)
  : x_(x), y_(y), rank_(++count_) {
}
double Point::x() const {
  return this->x_;
}
double Point::y() const {
  return this->y_;
}
void Point::move(double dx, double dy) {
  x_ += dx;
  y_ += dy;
}
Point Point::operator +(Point const& p2) const{
  return Point(x_ + p2.x_, y_+p2.y_);
}
bool Point::operator== (Point const& p2) const{
  return x_==p2.x_ && y_==p2.y_;
}

std::ostream&
operator<<(std::ostream & out, Point const& p) {
  out << p.rank_<<":(" << p.x_ << ", "
      << p.y_ << ")";
  return out;
}
```

Usage: main.cpp

```cpp
#include "point.hpp"
#include <iostream>

int main(int argc, char** argv) {
  Point p1(10, 20);
  std::cout << "    p1: " << p1 << std::endl;
  p1.move(42, 42);
  std::cout << "new p1: " << p1 << std::endl;
  Point p2 = Point(20, 40);
  std::cout << "    p2: " << p2 << std::endl;
  Point total = p1 + p2;
  std::cout << " total: " << total << std::endl;
  Point* courbe =
     new Point [3] {{10,20}, {11,21}, {12,22}};
  delete[] courbe;
}
```

Test unitaire: pointTest.cpp

```cpp
#define CATCH_CONFIG_MAIN
#include "catch.hpp"

#include "point.hpp"

TEST_CASE( "ctor et getters", "[point]" ) {
  Point c1(1,4);
  REQUIRE( c1.x() == 1);
  REQUIRE( c1.y() == 4);
}
```

Pour compiler: CMakeLists.txt

```
project(MyProject)
set(CMAKE_CXX_FLAGS  "${CMAKE_C_FLAGS} -Wall")
add_executable(prog main.cpp point.cpp)
add_executable(PointTest point.cpp pointTest.cpp)

enable_testing()
add_test(NAME point COMMAND PointTest)
```

Exemple de session

```
$ cmake . && make
(tests automatique du système + compilation)
$ ./prog
    p1: 1:(10, 20)
new p1: 1:(52, 62)
    p2: 2:(20, 40)
 total: 3:(72, 102)
$ ./PointTest
===============================================
All tests passed (2 assertions in 1 test case)
$ ctest
Test project
    Start 1: point
1/1 Test #1: point .........   Passed   0.00 sec

100% tests passed, 0 tests failed out of 1
$
```

Objet comme champ d'une classe

```cpp
class PointCouleur {
  Point p;
  int couleur;
  PointCouleur(double x, double y, int c)
     : p(x,y), couleur(c) {}
};
```