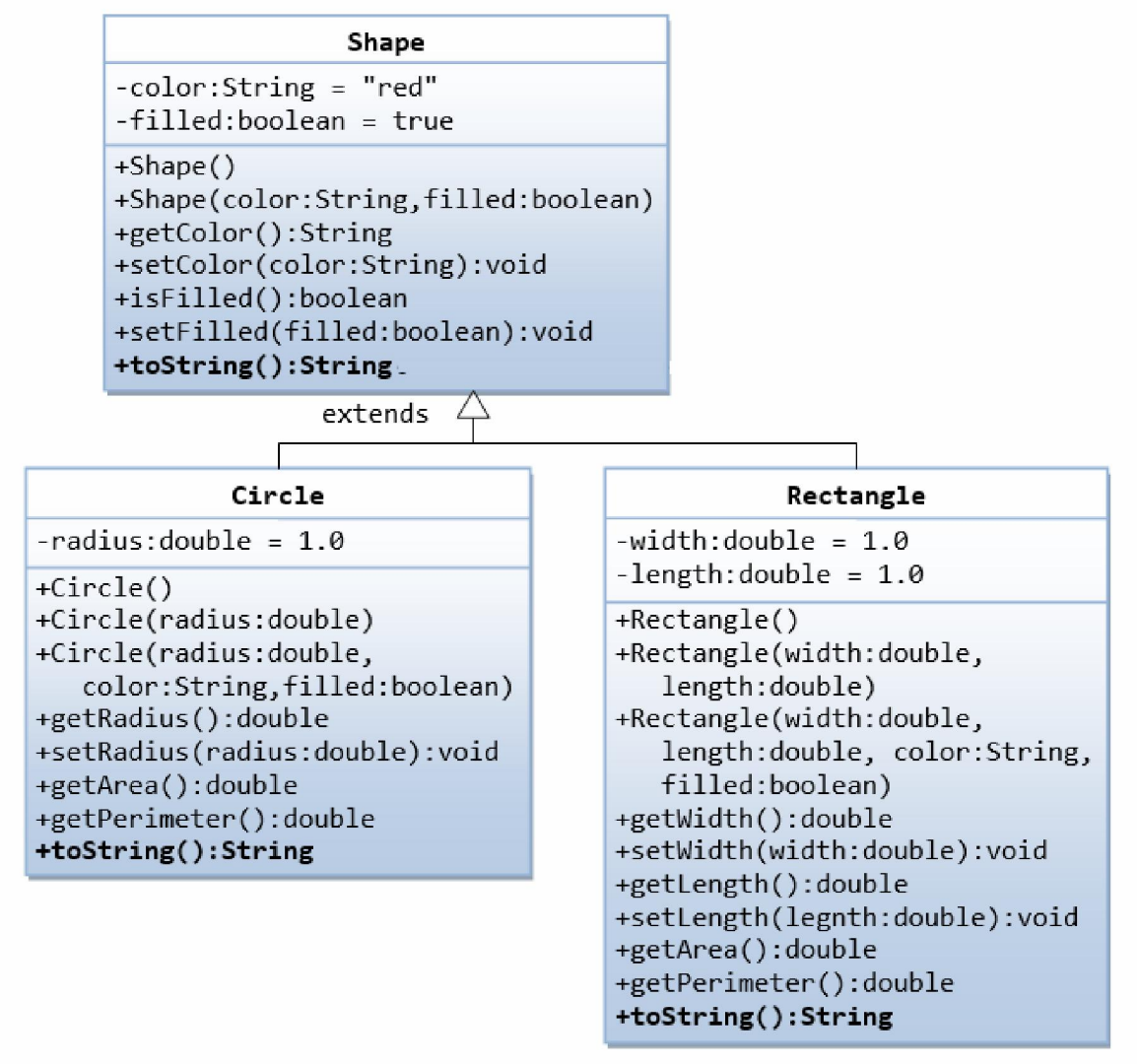
Exercițiul 12

Să se implementeze clasele `Shape`, `Circle` şi `Rectangle`, unde `Circle` şi `Rectangle` sunt derivate din `Shape` (a se vedea diagrama de clase UML).



Referințe:

Clasele `Circle` şi `Sphere`.

To do

1. Să se declare şi să se definească metoda `double getPerimeter()`[1] în clasele `Circle` şi `Rectangle`. Este nevoie ca aceasta să fie declarată şi în clasa `Shape`? Explicați.
2. Să se declare clasa `Shape` *clasa abstractă*. Ce se întâmplă?
3. Să se declare metodele `getArea()` şi `getPerimeter()` virtuale în clasa `Shape`. Observați ce se întâmplă.

[1] Metoda calculează şi returnează perimetrul figurii geometrice în a cărei clasă este definită.

Conținutul fișierului `test-shape.cpp` ar putea fi:

```
#include <iostream>
#include "circle.h"
#include "rectangle.h"

int main() {
  Circle c1(5, "blue");

  cout << c1.toString() << endl;
  cout << "aria=" << c1.getArea() << endl;

  Rectangle r1(5, 6, "green");

  cout << r1.toString() << endl;
  cout << "aria=" << r1.getArea() << endl;

  Shape *ps1, *ps2;  // pointeri la Shape

  ps1 = new Circle(6);

  cout << ps1->toString() << endl;
  cout << "aria=" << ps1->getArea() << endl;

  ps2 = new Rectangle(7, 8);

  cout << ps2->toString() << endl;
  cout << "aria=" << ps2->getArea() << endl;

  delete ps1;
  delete ps2;

  Shape s5 = Circle(6);

  cout << s5.toString() << endl;
  cout << "aria=" << s5.getArea() << endl;

  Circle c3(8);
  Shape& rs3 = c3;    // referinta

  cout << rs3.toString() << endl;
  cout << "aria=" << rs3.getArea() << endl;

  Circle c4(9);
  Shape* ps4 = &c4;

  cout << ps4->toString() << endl;
  cout << "aria=" << ps4->getArea() << endl;

  return 0;
}
```