Challenge 1: Rearranging a Class

In this challenge, you'll be splitting up a class into .h and .cpp files.

WE'LL COVER THE FOLLOWING

Problem Statement

We have provided you the implementation for the `Planet` class. Your task is to ensure abstraction by splitting up the code into **header** and **.cpp** files.

In the code below, `Planet.h` and `Planet.cpp` are empty, whereas `main.cpp` contains the full implementation. The class contains three variables:

1. `radius` - The distance from the center of the planet.
2. `mass` - The mass of the planet.
3. `G` - The gravitational constant.

The `getMass()` function simply returns the mass of the planet and `calculateGravity()` calculates the gravitational strength of the planet.

The end goal is to remove all the implementation of the `Planet` class from `main.cpp`.

Good luck!

main.cpp

main.cpp

Planet.h

Planet.cpp

```
#include <iostream>
using namespace std;

class Planet {
  double radius;
  double mass;
  const double G = 6.673e-11;

  public:
  Planet(double r, double m){
    radius = r;
    mass = m;
  }

  double getMass(){
    return mass;
  }

  void setMass(double m){
    mass = m;
  }

  double getRadius(){
    return radius;
  }

  void setRadius(double r){
    radius = r;
  }

  double calculateGravity(){
    double gravity = (G * mass)/(radius * radius);
    return gravity;
  }
};

int main() {
  Planet earth(6.38e6, 5.98e24);
  double m = earth.getMass();
  cout << "Mass: " << m << endl;
  double gravStrength = earth.calculateGravity();
  cout << "Gravitational strength: "<< gravStrength << endl;
}
```

Solution Review

This problem is very similar to what we implemented in the Abstraction in Header Files. Let's take a look at the final form of the three files and explain them one by one:

main.cpp

Planet.h

Planet.cpp

```
#ifndef PLANET_H
#define PLANET_H

class Planet {
  double radius;
  double mass;
  const double G = 6.673e-11;

  public:
  Planet(double r, double m);
  double getMass();
  void setMass(double m);
  double getRadius();
  void setRadius(double r);
  double calculateGravity();
};

#endif
```

Planet.h

The header file only contains the declaration of the class and all its members (variables and methods). Only the `G` variable is defined since its a `const double` and its value will not change.

Planet.cpp

All of the implementations go in this file. We simply write the class name followed by the scope resolution operator and the function name. After this comes the code for that particular function. Be sure to include `Planet.h` in this file.

main.cpp

As we can see, `main.cpp` is a much smaller file now, containing only the main function where the program is executed. Be sure to include `Planet.h` in this file.