- Exercise

Let's solve an exercise in this lesson related to example 2 of the previous lesson.

WE'LL COVER THE FOLLOWING

- Problem statement

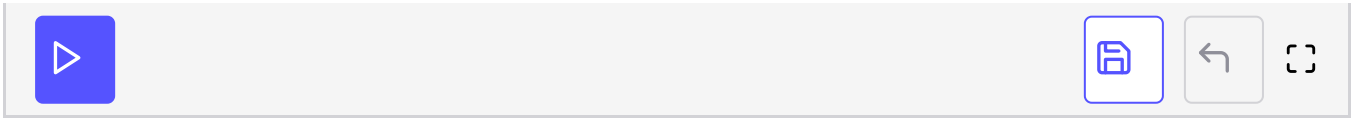
Problem statement

The class hierarchy in the file was built as we have seen in the previous example, following the famous template method design pattern. The typical implementation of the template method pattern is to use NVI. NVI stands for non-virtual interface, meaning the interface should not be virtual. NVI requires non-virtual inheritance, which is not possible in Java or Python. Click here for more details.

Refactor it with the help of `override` and `final`.

```
#include <iostream>

class Sort{
public:
  virtual void processData(){
    readData();
    sortData();
    writeData();
  }
private:
  virtual void readData(){}
  virtual void sortData() = 0;
  virtual void writeData(){}
};


// Try using final and override

class QuickSort: public Sort{
private:
  void readData(){
    std::cout << "readData" << std::endl;
```

```
  }
  void sortData(){
    std::cout <<  "sortData" << std::endl;

  }
  void writeData(){
    std::cout << "writeData" << std::endl;
  }
};


int main(){

  std::cout << std::endl;

  //Sort* sort = new QuickSort;
  //sort->processData();

  std::cout << std::endl;

}
```



In the next lesson, we'll discuss the solution to this exercise.