

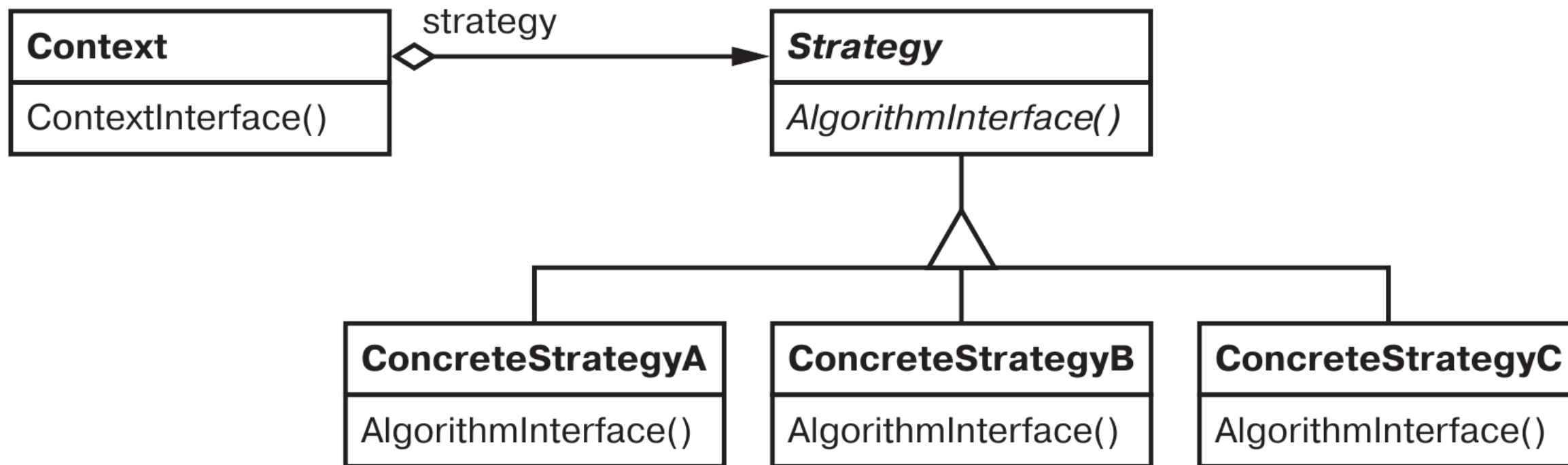


# Lesson #3

## Pattern “Strategy” – Example

# Паттерн «Strategy» («Стратегия»)

## Структура



# Паттерн «Strategy» («Стратегия»)

## Участники

*Strategy* – стратегия

объявляет общий для всех поддерживаемых алгоритмов интерфейс (абстрактный класс). Класс `Context` пользуется этим интерфейсом для вызова конкретного алгоритма, определенного в классе `ConcreteStrategy`;

*ConcreteStrategy* – конкретная стратегия

реализует алгоритм, использующий интерфейс, объявленный в классе `Strategy`;

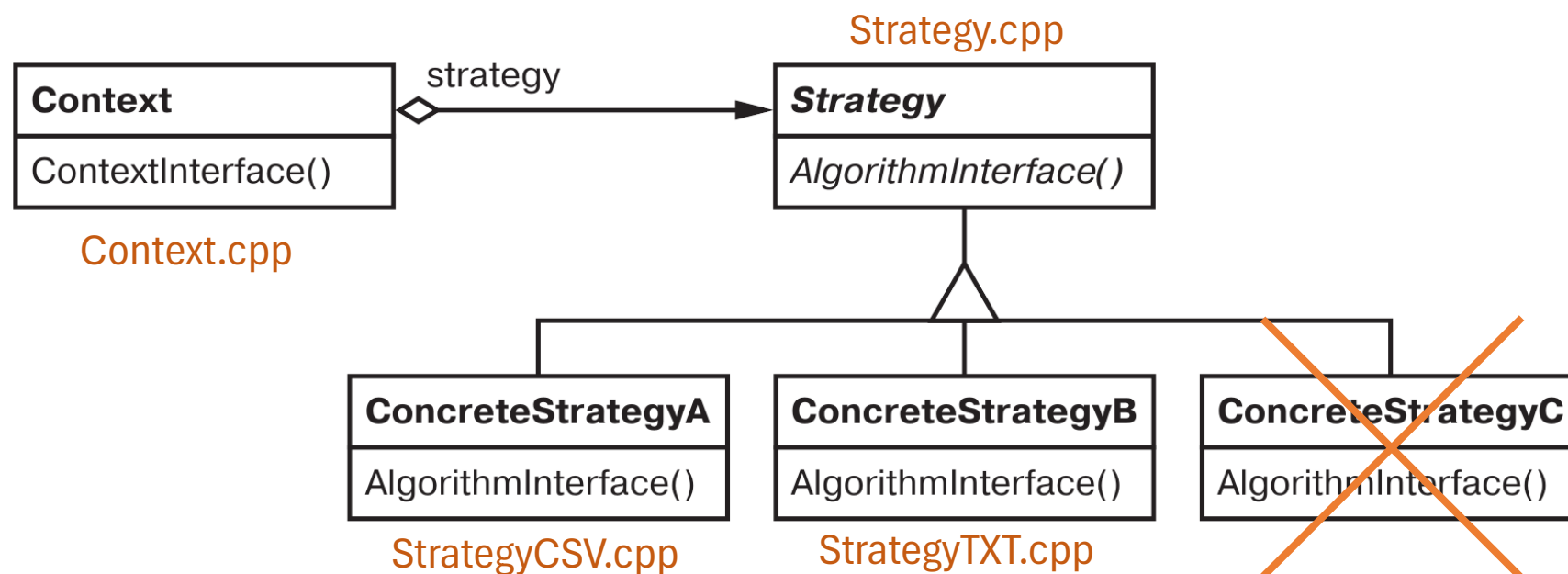
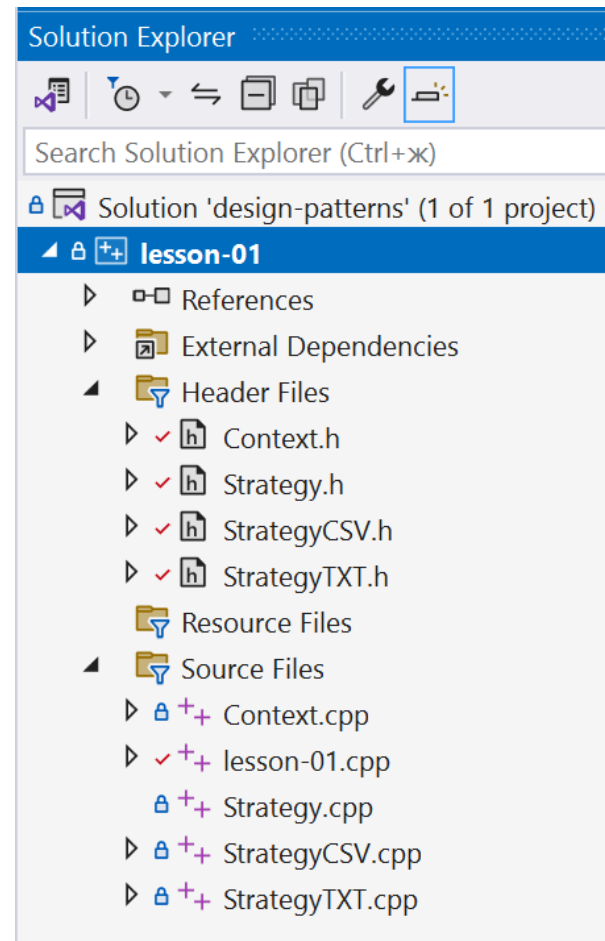
*Context* – контекст

настраивается объектом класса `ConcreteStrategy`;  
хранит ссылку на объект класса `Strategy`;  
может определять интерфейс, который позволяет объекту `Strategy` обращаться к данным контекста.

# Паттерн «Strategy» («Стратегия»)

## Пример кода

<https://github.com/VASoftLab/design-patterns.git>



# Паттерн «Strategy» («Стратегия»)

## Strategy.h

```
1      #pragma once
2      √ #include <string>
3      | #include <iostream>
4      | // Абстрактный класс стратегии
5      √ class Strategy
6      | {
7      |     public:
8      |         virtual ~Strategy() = default;
9      |         virtual std::string exportData() = 0;
10     | };

```

PATTERN "STRATEGY"

# Паттерн «Strategy» («Стратегия»)

## StrategyCSV.h

PATTERN "STRATEGY"

```
1  #pragma once
2  #include "Strategy.h"
3  // Класс стратегии для экспорта данных в CSV формате
4  class StrategyCSV : public Strategy
5  {
6  public:
7      StrategyCSV( );
8      ~StrategyCSV( ) = default;
9  private:
10     std::string exportData( ) override;
11 }
```

# Паттерн «Strategy» («Стратегия»)

## StrategyTXT.h

PATTERN "STRATEGY"

```
1  #pragma once
2  #include "Strategy.h"
3  // Класс стратегии для экспорта данных в TXT формате
4  class StrategyTXT : public Strategy
5  {
6  public:
7      StrategyTXT();
8      ~StrategyTXT() = default;
9  private:
10     std::string exportData() override;
11 }
```

# Паттерн «Strategy» («Стратегия»)

## Context.h

```
1  #pragma once
2  √ #include <memory>
3  | #include "Strategy.h"
4  | // Класс контекста
5  √ class Context
6  | {
7  | private:
8  |     // Указатель на объект стратегии
9  |     std::unique_ptr<Strategy> _strategy;
10 | public:
11 |     Context();
12 |     // Метод для задания стратегии
13 |     void setStrategy(std::unique_ptr<Strategy>&& strategy);
14 |     // Полиморфный метод для вывода данных
15 |     std::string exportData();
16 | };
```

PATTERN "STRATEGY"



# Паттерн «Strategy» («Стратегия»)

## StrategyCSV.cpp

PATTERN "STRATEGY"

```
1  #include "StrategyCSV.h"
2
3  StrategyCSV::StrategyCSV( )
4  {
5      std::cout << "StrategyCSV constructor" << std::endl;
6  }
7  std::string StrategyCSV::exportData( )
8  {
9      std::cout << "Export data with CSV strategy" << std::endl;
10     return "CSV";
11 }
```

# Паттерн «Strategy» («Стратегия»)

## StrategyTXT.cpp

PATTERN «STRATEGY»

```
1  #include "StrategyTXT.h"
2  [
3  StrategyTXT::StrategyTXT( )
4  {
5      std::cout << "StrategyTXT constructor" << std::endl;
6  }
7  std::string StrategyTXT::exportData( )
8  {
9      std::cout << "Export data with TXT strategy" << std::endl;
10     return "TXT";
11 }
```

# Паттерн «Strategy» («Стратегия»)

## Context.cpp

```
1  #include "Context.h"
2  Context::Context()
3  {
4      _strategy = NULL;
5  }
6  void Context::setStrategy(std::unique_ptr<Strategy>&& strategy)
7  {
8      if (strategy)
9          _strategy = std::move(strategy);
10 }
11 std::string Context::exportData()
12 {
13     if (_strategy)
14         return _strategy->exportData();
15     else
16         return "ERROR";
17 }
18
```

# Паттерн «Strategy» («Стратегия»)

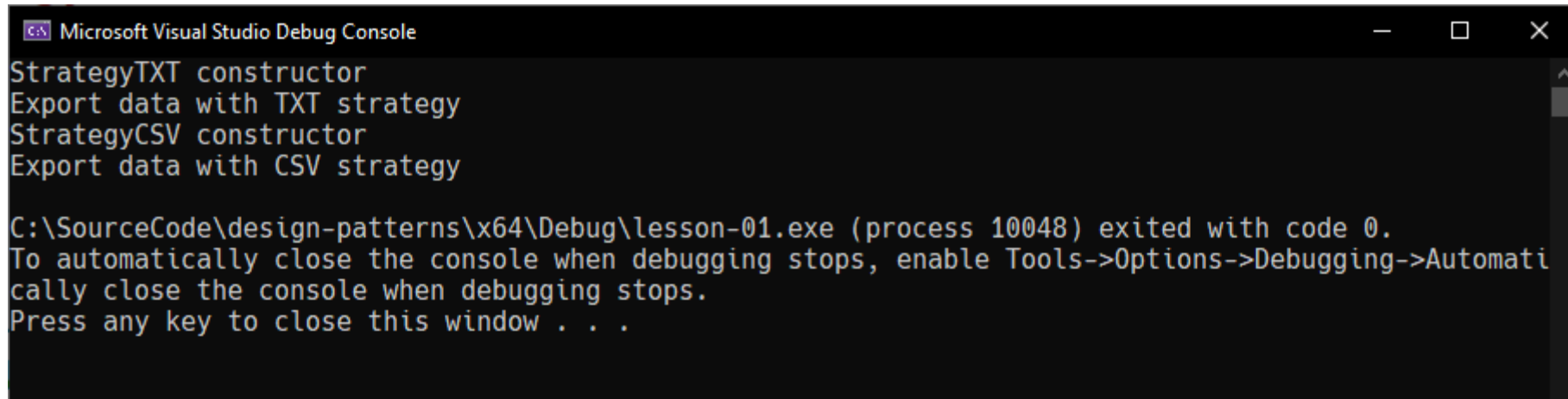
## lesson-01.cpp

PATTERN "STRATEGY"

```
1  ✓ #include <iostream>
2  | #include "Context.h"
3  | #include "StrategyCSV.h"
4  | #include "StrategyTXT.h"
5
6  ✓ int main()
7  | {
8  |     // Создание объекта контекста
9  |     std::unique_ptr<Context> context = std::make_unique<Context>( );
10 |     // Назначение стратегии A (TXT)
11 |     context->setStrategy(std::make_unique<StrategyTXT>( ));
12 |     context->exportData( );
13 |     // Назначение стратегии B (CSV)
14 |     context->setStrategy(std::make_unique<StrategyCSV>( ));
15 |     context->exportData( );
16 | }
```

# Паттерн «Strategy» («Стратегия»)

## Результаты работы



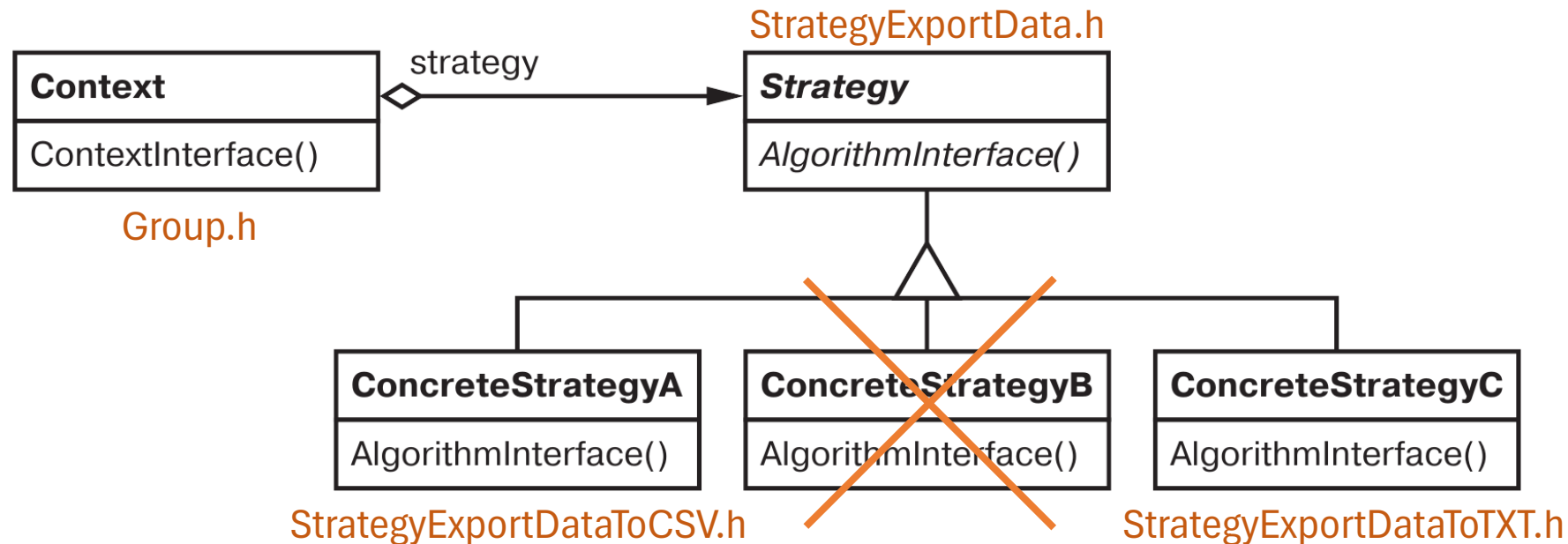
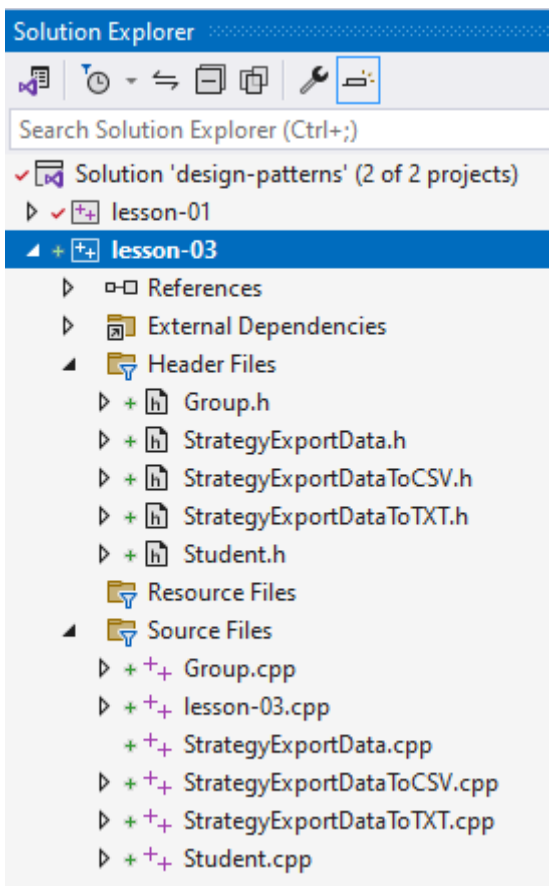
```
Microsoft Visual Studio Debug Console
StrategyTXT constructor
Export data with TXT strategy
StrategyCSV constructor
Export data with CSV strategy

C:\SourceCode\design-patterns\x64\Debug\lesson-01.exe (process 10048) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automati
cally close the console when debugging stops.
Press any key to close this window . . .
```

to be continued...

# Структура решения

## lesson-03.cpp



# Класс «Student»

PATTERN "STRATEGY"

```

1  #pragma once
2  #include <string>
3  #include <iostream>
4  class Student
5  {
6  private:
7      std::string firstName; // Имя
8      std::string lastName; // Фамилия
9      std::string indInsAccNumber; // СНИЛС
10     unsigned short yearOfBirth; // Год рождения
11     float avgGrade;
12 public:
13     Student();
14     Student(
15         std::string firstname,
16         std::string lastname,
17         std::string indinsaccnumber,
18         unsigned short yearofbirth,
19         float avggrade);
20     ~Student() = default;
21
22     std::string getFullName();
23     std::string getIndInsAccNumber();
24     unsigned short getYearOfBirth();
25     float getAvgGrade();
26 };

```

```

1  #include "Student.h"
2
3  Student::Student()
4  {
5      firstName = "";
6      lastName = "";
7      indInsAccNumber = "000-000-000 00";
8      yearOfBirth = 1900;
9      avgGrade = 0.0;
10
11     std::cout << "Student (default constructor)" << std::endl;
12 }
13
14 Student::Student(std::string firstname,
15                 std::string lastname,
16                 std::string indinsaccnumber,
17                 unsigned short yearofbirth,
18                 float avggrade)
19 {
20     firstName = firstname;
21     lastName = lastname;
22     indInsAccNumber = indinsaccnumber;
23     yearOfBirth = yearofbirth;
24     avgGrade = avggrade;
25
26     std::cout << "Student (parameterized constructor)" << std::endl;
27 }
28
29 std::string Student::getFullName()
30 {
31     return lastName + " " + firstName;
32 }
33
34 std::string Student::getIndInsAccNumber()
35 {
36     return indInsAccNumber;
37 }
38 unsigned short Student::getYearOfBirth()
39 {
40     return yearOfBirth;
41 }
42 float Student::getAvgGrade()
43 {
44     return avgGrade;
45 }

```

# Класс «Group»

## PATTERN "STRATEGY"

```

1  #pragma once
2
3  #include <string>
4  #include <vector>
5  #include <memory>
6
7  #include "StrategyExportData.h"
8  #include "Student.h"
9
10 class Group
11 {
12 public:
13     std::string name;
14     std::vector<std::unique_ptr<Student>> group;
15 private:
16     // Указатель на объект стратегии
17     std::unique_ptr<StrategyExportData> _strategy;
18 public:
19     Group();
20     ~Group();
21     // Метод для задания стратегии
22     void setStrategy(std::unique_ptr<StrategyExportData>&& strategy);
23     // Полиморфный метод для вывода данных
24     std::string exportData();
25 };

```

```

1  #include "Group.h"
2
3  Group::Group()
4  {
5      _strategy = nullptr;
6  }
7  Group::~Group()
8  {
9      _strategy = nullptr;
10     for (auto& student : group)
11         student = nullptr;
12 }
13 void Group::setStrategy(std::unique_ptr<StrategyExportData>&& strategy)
14 {
15     if (strategy)
16         _strategy = std::move(strategy);
17 }
18 std::string Group::exportData()
19 {
20     if (_strategy)
21     {
22         return _strategy->exportData(group);
23     }
24     else
25         return "ERROR";
26 }

```



# Класс «StrategyExportData»

```
1  #pragma once
2  #include <string>
3  #include <iostream>
4  #include <vector>
5
6  #include "Student.h"
7
8  // Абстрактный класс стратегии
9  class StrategyExportData
10 {
11 public:
12     virtual ~StrategyExportData() = default;
13     virtual std::string exportData(std::vector<std::unique_ptr<Student>>& students) = 0;
14 };
```

# «StrategyExportDataToCSV» «StrategyExportDataToTXT»

```
1  #pragma once
2  #include "StrategyExportData.h"
3  #include "Group.h"
4
5  // Класс стратегии для экспорта данных в CSV формате
6  class StrategyExportDataToCSV : public StrategyExportData
7  {
8  public:
9      StrategyExportDataToCSV();
10     ~StrategyExportDataToCSV() = default;
11 private:
12     std::string exportData(std::vector<std::unique_ptr<Student>>& students) override;
13 };
```

```
1  #pragma once
2  #include "StrategyExportData.h"
3  // Класс стратегии для экспорта данных в TXT формате
4  class StrategyExportDataToTXT : public StrategyExportData
5  {
6  public:
7      StrategyExportDataToTXT();
8      ~StrategyExportDataToTXT() = default;
9 private:
10     std::string exportData(std::vector<std::unique_ptr<Student>>& students) override;
11 };
```

# «StrategyExportDataToCSV»

```

1  #include "StrategyExportDataToCSV.h"
2  #include <sstream>
3
4  StrategyExportDataToCSV::StrategyExportDataToCSV()
5  {
6      std::cout << std::endl << "StrategyCSV constructor" << std::endl;
7  }
8  std::string StrategyExportDataToCSV::exportData(std::vector<std::unique_ptr<Student>>& students)
9  {
10     std::cout << "Export data with CSV strategy" << std::endl;
11
12     std::stringstream output;
13     std::string sep = ",";
14
15     int counter = 0;
16
17     try
18     {
19         for (auto& student : students)
20         {
21             counter++;
22             std::cout << counter << sep << student->getFullName()
23                 << sep << student->getYearOfBirth() |
24                 << sep << student->getIndInsAccNumber() << std::endl;
25             output << counter << sep << student->getFullName()
26                 << sep << student->getYearOfBirth()
27                 << sep << student->getIndInsAccNumber() << std::endl;
28         }
29         return output.str();
30     }
31     catch (...)
32     {
33         return "CSV ERROR";
34     }
35 }

```

# «StrategyExportDataToTXT»

```

1  #include "StrategyExportDataToTXT.h"
2  #include <sstream>
3
4  StrategyExportDataToTXT::StrategyExportDataToTXT()
5  {
6      std::cout << std::endl << "StrategyTXT constructor" << std::endl;
7  }
8  std::string StrategyExportDataToTXT::exportData(std::vector<std::unique_ptr<Student>>& students)
9  {
10     std::cout << "Export data with TXT strategy" << std::endl;
11
12     std::stringstream output;
13     int counter = 0;
14
15     try
16     {
17         for (auto& student : students)
18         {
19             counter++;
20             std::cout << counter << "\t" << student->getYearOfBirth()
21                 << "\t" << student->getIndInsAccNumber()
22                 << "\t" << student->getAvgGrade() << "\t"
23                 << student->getFullName() << std::endl;
24             output << counter << "\t" << student->getYearOfBirth()
25                 << "\t" << student->getIndInsAccNumber()
26                 << "\t" << student->getAvgGrade() << "\t"
27                 << student->getFullName() << std::endl;
28         }
29         return output.str();
30     }
31     catch (...)
32     {
33         return "TXT ERROR";
34     }
35 }

```

# Вызывающий модуль

```

1  #include <iostream>
2  #include <fstream>
3
4  #include "StrategyExportDataToCSV.h"
5  #include "StrategyExportDataToTXT.h"
6
7  #include "Group.h"
8
9  int main()
10 {
11     setlocale(LC_ALL, "Russian");
12
13     // Создание объекта контекста
14     std::unique_ptr<Group> uts21 = std::make_unique<Group>();
15     // Ввод данных о группе
16     uts21->name = "UTS-21";
17     uts21->group.push_back(std::move(std::make_unique<Student>( "Александра", "Алексеева", "123-456-789 01", 2010, 4.5)));
18     uts21->group.push_back(std::move(std::make_unique<Student>( "Анастасия", "Борщёва", "123-456-789 02", 2010, 4.25)));
19     uts21->group.push_back(std::move(std::make_unique<Student>( "Евгений", "Зайцев", "123-456-789 03", 2009, 3.0)));
20     uts21->group.push_back(std::move(std::make_unique<Student>( "Вадим", "Згурский", "123-456-789 04", 2009, 5.0)));
21     uts21->group.push_back(std::move(std::make_unique<Student>( "Алексей", "Катков", "123-456-789 05", 2009, 5.0)));
22
23     // Назначение стратегии - экспорт в CSV
24     uts21->setStrategy(std::make_unique<StrategyExportDataToCSV>());
25     std::string outCSV = uts21->exportData();
26
27     // Назначение стратегии - экспорт в TXT
28     uts21->setStrategy(std::make_unique<StrategyExportDataToTXT>());
29     std::string outTXT = uts21->exportData();
30
31     std::ofstream fileCSV("output.csv");
32     fileCSV << outCSV;
33     fileCSV.close();
34
35     std::ofstream fileTXT("output.txt");
36     fileTXT << outTXT;
37     fileTXT.close();
38
39 }

```

# Результаты работы

```
Microsoft Visual Studio Debug Console

StrategyCSV constructor
Export data with CSV strategy
1,Алексеева Александра,2010,123-456-789 01
2,Борщёва Анастасия,2010,123-456-789 02
3,Зайцев Евгений,2009,123-456-789 03
4,Згурский Вадим,2009,123-456-789 04
5,Катков Алексей,2009,123-456-789 05

StrategyTXT constructor
Export data with TXT strategy
1      2010      123-456-789 01  4.5      Алексеева Александра
2      2010      123-456-789 02  4.25     Борщёва Анастасия
3      2009      123-456-789 03   3        Зайцев Евгений
4      2009      123-456-789 04   5        Згурский Вадим
5      2009      123-456-789 05   5        Катков Алексей

D:\SourceCode\design-patterns\x64\Debug\lesson-03.exe (process 27980) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

output	csv	199	02/24/2024 11:48	-a--
output	txt	214	02/24/2024 11:48	-a--

```
output.csv - Notepad
File Edit Format View Help
1,Алексеева Александра,2010,123-456-789 01
2,Борщёва Анастасия,2010,123-456-789 02
3,Зайцев Евгений,2009,123-456-789 03
4,Згурский Вадим,2009,123-456-789 04
5,Катков Алексей,2009,123-456-789 05
```

```
output.txt - Notepad
File Edit Format View Help
1      2010      123-456-789 01  4.5      Алексеева Александра
2      2010      123-456-789 02  4.25     Борщёва Анастасия
3      2009      123-456-789 03   3        Зайцев Евгений
4      2009      123-456-789 04   5        Згурский Вадим
5      2009      123-456-789 05   5        Катков Алексей
```

	A	B	C	D
1	1	Алексеева Александра	2010	123-456-789 01
2	2	Борщёва Анастасия	2010	123-456-789 02
3	3	Зайцев Евгений	2009	123-456-789 03
4	4	Згурский Вадим	2009	123-456-789 04
5	5	Катков Алексей	2009	123-456-789 05

# Задания на работу

УТС/6-21-о ООППРО		
Вариант	№	
1	1	Алексеева Александра Владимировна
2	2	Борщёва Анастасия Сергеевна
3	3	Зайцев Евгений Дмитриевич
4	4	Згурский Вадим Сергеевич
5	5	Катков Алексей Сергеевич
6	6	Кузнецов Матвей Анатольевич
7	7	Кучерук Дмитрий Олегович
8	8	Леонтьев Данил Сергеевич
9	9	Лозко Виктор Витальевич
10	10	Мохначев Максим Александрович
1	11	Мутин Кирилл Дмитриевич
2	12	Мутин Константин Алексеевич
3	13	Пронин Кирилл Андреевич
4	14	Рябый Андрей Игоревич
5	15	Самойлова Вероника Андреевна
6	16	Шапошников Данила Павлович
7	17	Шатунов Александр Вячеславович
8	18	Яцкевич Руслан Игоревич

## Задания на работу

- Вариант 1** – Кодировщик систем счисления
- Вариант 2** – Словарь русский/английский/французский
- Вариант 3** – Конвертер списка в HTML/XML
- Вариант 4** – Сортировщик массива пузырьковая/выбором
- Вариант 5** – Конвертер единиц измерения
- Вариант 6** – Расчет среднего балла
- Вариант 7** – Расчет стоимости проживания в отеле
- Вариант 8** – Расчет размера скидки
- Вариант 9** – Кодировщик пароля (система аутентификации)
- Вариант 10** – Расчет стоимости техобслуживания для авто