

Министерство науки и высшего образования Российской Федерации

Калужский филиал  
федерального государственного бюджетного образовательного  
учреждения высшего образования  
**«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»**  
(КФ МГТУ им. Н.Э. Баумана)

**С.А. Глебов, С.С. Гришунов**

**ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ И ПРИНЦИПЫ ООП**  
Методические указания к выполнению домашней работы  
по курсу «Объектно-ориентированное программирование»

Калуга - 2019

УДК 004.71  
ББК 32.972.1  
Г53

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/9 от «10» апреля 2019 г.

Зав. кафедрой ИУ4-КФ

 к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 11 от «19» апреля 2019 г.

Председатель методической  
комиссии факультета ИУ-КФ

 к.т.н., доцент М.Ю. Адкин

- Методической комиссией

КФ МГТУ им.Н.Э. Баумана протокол № 7 от «7» 05 2019 г.

Председатель методической комиссии  
КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:

к.т.н., доцент кафедры ИУ3-КФ


 А.В. Фиошин

Авторы

к.ф.-м.н., доцент кафедры ИУ4-КФ

асс. кафедры ИУ4-КФ

 С.А. Глебов

 С.С. Гришунов

#### Аннотация

Методические указания к выполнению домашней работы по курсу «Объектно-ориентированное программирование» содержат краткое описание принципов SOLID, а также слоистой архитектуры программных систем и задание на выполнение домашней работы.

Предназначены для студентов 2-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
ПРИНЦИПЫ SOLID .....	6
СЛОИСТАЯ АРХИТЕКТУРА.....	9
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ .....	11
ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ.....	11
ВАРИАНТЫ ЗАДАНИЙ.....	11
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ .....	21
ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ.....	21
ОСНОВНАЯ ЛИТЕРАТУРА.....	22
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА .....	22

## **ВВЕДЕНИЕ**

Настоящие методические указания составлены в соответствии с программой проведения курса «Объектно-ориентированное программирование» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 2-го курса направления подготовки 09.03.04 «Программная инженерия», содержат краткое описание принципов SOLID, а также слоистой архитектуры программных систем и задание на выполнение домашней работы.

Для выполнения домашней работы студенту необходимы базовые знания по программированию на высокоуровневом языке программирования C#.

Требования к программному обеспечению:

- Microsoft Windows 7/8/10
- Microsoft Visual Studio Community 2017

## **ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ**

Целью выполнения домашней работы является формирование практических навыков объектно-ориентированного проектирования в соответствии с принципами SOLID и использованием паттернов проектирования.

Основными задачами выполнения домашней работы являются:

- Закрепить навыки объектно-ориентированного проектирования в соответствии с принципами SOLID
- Научиться разрабатывать программные системы слоистой архитектуры.

Результатами работы являются:

- Библиотека классов, моделирующая предметную область согласно варианту задания
- Набор модульных тестов для демонстрации возможностей разработанной библиотеки классов
- Подготовленный отчет

## ПРИНЦИПЫ SOLID

SOLID это аббревиатура пяти основных принципов проектирования в объектно-ориентированном программировании — [Single responsibility](#), [Open-closed](#), [Liskov substitution](#), [Interface segregation](#) и [Dependency inversion](#) (принципы единственной ответственности, открытости / закрытости, подстановки Барбары Лисков, разделения интерфейса и инверсии зависимостей)

Аббревиатура SOLID была предложена Робертом Мартином, автором нескольких книг, широко известных в сообществе разработчиков. Эти принципы позволяют строить на базе ООП масштабируемые и сопровождаемые программные продукты с понятной бизнес-логикой.

### Принцип единственной обязанности (SRP)

Принцип единственной обязанности (Single-Responsibility Principle – SRP) У класса должна быть только одна причина для изменения.

Каждая обязанность – это ось изменения. Любое изменение требований проявляется в изменении распределения обязанностей между классами. Если класс берет на себя несколько обязанностей, то у него появляется несколько причин для изменения. Если класс отвечает за несколько действий, то его обязанности оказываются связанными. Изменение одной обязанности может привести к тому, что класс перестанет справляться с другими. Такого рода связанность – причина хрупкого дизайна, который неожиданным образом разрушается при изменении.

### Принцип открытости/закрытости (OCP)

Принцип открытости/закрытости (Open/Closed Principle – OCP) Программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для модификации.

Описание принципа OCP

У модулей, согласованных с принципом OCP, есть две основных характеристики.

1. Они открыты для расширения. Это означает, что поведение модуля можно расширить. Когда требования к приложению изменяются, мы добавляем в модуль новое поведение, отвечающее изменившимся требованиям. Иными словами, мы можем изменить состав функций модуля.

2. Они закрыты для модификации. Расширение поведения модуля не сопряжено с изменениями в исходном или двоичном коде модуля. Двоичное исполняемое представление модуля, будь то компоновка библиотеки, DLL или EXE-файл, остается неизменным.

### **Принцип подстановки Лискоу (Liskov Substitution Principle)**

Должна быть возможность вместо базового типа подставить любой его подтип.

Этот принцип был сформулирован Барбарой Лискоу в 1988 году. Она писала:

«Мы хотели бы иметь следующее свойство подстановки: если для каждого объекта  $o_1$  типа  $S$  существует объект  $o_2$  типа  $T$ , такой, что для любой программы  $P$ , определенной в терминах  $T$ , поведение  $P$  не изменяется при замене  $o_1$  на  $o_2$ , то  $S$  является подтипом  $T$ .»

Важность этого принципа становится очевидной, если рассмотреть последствия его нарушения. Предположим, что имеется функция  $f$ , принимающая в качестве аргумента ссылку на некоторый базовый класс  $B$ . Предположим также, что при передаче функции  $f$  ссылки на объект класса  $D$ , производного от  $B$ , она ведет себя неправильно. Тогда  $D$  нарушает принцип LSP. Понятно, что класс  $D$  оказывается хрупким в присутствии  $f$ . У авторов  $f$  может возникнуть искушение включить некоторый тест для  $D$ , проверяющий, что  $f$  правильно ведет себя при передаче ей  $D$ . Но такой тест нарушает принцип OCP, потому что  $f$  теперь не закрыта от других классов, производных от  $B$ . Подобные тесты демонстрируют неопытность разработчиков или, что еще хуже, чрезмерно поспешную реакцию на нарушение LSP.

### **Принцип разделения интерфейсов (ISP)**

Этот принцип относится к недостаткам «жирных» интерфейсов. Говорят, что класс имеет «жирный» интерфейс, если функции этого интерфейса недостаточно сцепленные. Иными словами, интерфейс класса можно разбить на группы методов. Каждая группа предназначена для обслуживания разнотипных клиентов. Одним клиентам нужна одна группа методов, другим – другая. Принцип ISP допускает, что могут существовать объекты, нуждающиеся в несцепленных интерфейсах, однако предполагает, что клиентам необязательно знать, что это единый класс. Клиенты должны лишь знать об абстрактных интерфейсах, обладающих свойством сцепленности.

### **Принцип инверсии зависимости (Dependency-Inversion Principle – DIP)**

- Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те и другие должны зависеть от абстракций
- Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций



## СЛОИСТАЯ АРХИТЕКТУРА

**Слоистая архитектура** — это такая архитектура системы, при которой система состоит из некоторой упорядоченной совокупности программных подсистем, называемых слоями, такой, что:

- на каждом слое ничего не известно о свойствах (и даже существовании) последующих (более высоких) слоев;
- каждый слой может взаимодействовать по управлению (обращаться к компонентам) с непосредственно предшествующим (более низким) слоем через заранее определенный интерфейс, ничего не зная о внутреннем строении всех предшествующих слоев;
- каждый слой располагает определенными ресурсами, которые он либо скрывает от других слоев, либо предоставляет непосредственно последующему слою (через указанный интерфейс) некоторые их абстракции.

Таким образом, в слоистой программной системе каждый слой может реализовать некоторую абстракцию данных. Связи между слоями ограничены передачей значений параметров обращения каждого слоя к смежному снизу слою и выдачей результатов этого обращения от нижнего слоя верхнему. Недопустимо использование глобальных данных несколькими слоями.

Данная архитектура позволяет вносить изменения в каждый из слоев, не затрагивая логику работы остальных слоев системы.

Основные слои представлены на рисунке 1.

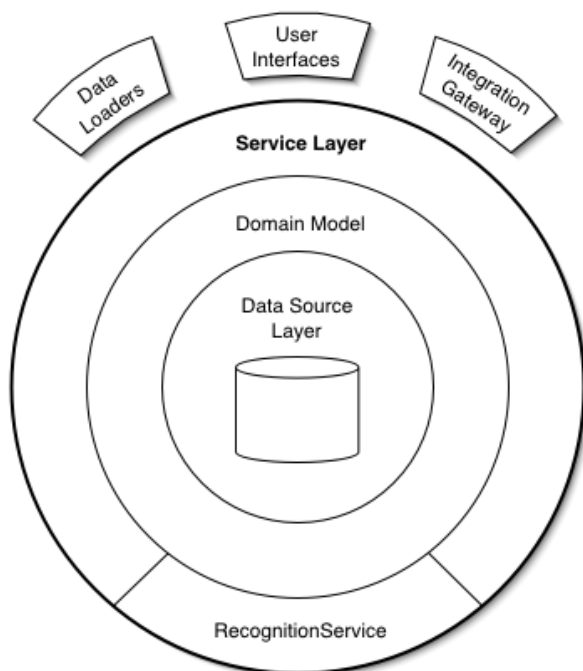


Рис. 1.

**Data Source Layer** – данный слой отвечает за доступ к данным (файловому хранилищу, базам данных и т.д.), предоставляет интерфейс для получения необходимых данных вышестоящим слоям.

**Domain Model** – отвечает за бизнес-логику модели. Данный уровень содержит сущности предметной модели, загружаемые и сохраняемые с использованием интерфейса DSL.

**Service Layer** – предоставляет безопасный интерфейс для взаимодействия с объектами предметной модели. Данный интерфейс может использоваться несколькими сторонними приложениями, реализующими взаимодействие с пользователями (**User Interfaces** или **Data Loader** для организации пакетного экспорта/импорта данных) и другими системами (**Integration Gateway**).

## **ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

Доработать предметную модель согласно выданному варианту задания.

### **ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ**

При выполнении лабораторной работы необходимо использовать стандарт кодирования языка C#. Для разработанных классов создать набор модульных тестов, проверяющих корректность кода, а также демонстрирующих полученный API.

### **ВАРИАНТЫ ЗАДАНИЙ**

#### **Вариант № 1. «Служба доставки»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, списки клиентов и товаров «жестко» заданы в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списков клиентов и товаров (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку списка клиентов и списка товаров из текстовых файлов «customers.dat» и «items.dat» соответственно.

В каждой строке текстового файла «customers.dat» содержится запись о клиенте в следующем формате:

Code|ContactPhone|FullName|Privileged

Т.е. перечислены значения атрибутов, разделенные символом «|». Если клиент привилегированный, то в поле «Privileged» указано значение «Y», в противном случае – значение «N». Пример файла, содержащего данные о 5 клиентах:

```
LDN0012|+79539008765|Воскресенский Никанор Онуфриевич|Y
LDN0027|+79605157707|Вознесенская Евпраксия Сергеевна|N
LDN0239|+79108765432|Троицкий Георгий Дмитриевич|N
NYK0003|+79208163490|Павловская Фекла Петровна|Y
NYK4011|+79037120354|Преображенский Федор Павлович|N
```

В каждой строке текстового файла «items.dat» содержится запись о товаре в следующем формате:

```
Article|Name|UnitPrice
```

Т.е. перечислены значения атрибутов, разделенные символом «|». Цена товара (поле «UnitPrice») приведено с точностью до 2 знаков после запятой, десятичным разделителем является точка. Пример файла, содержащего данные о 7 товарах:

```
MAR-25|Маргарита, 25 см|255.00
MAR-31|Маргарита, 31 см|355.00
CO-31|Цыпленок с чесноком, 31 см|415.00
CO-36|Цыпленок с чесноком, 36 см|545.00
SOS|Соус|20.00
JU|Сок 'Добрый', 1 л|70.00
PP|Картофельные дольки|195.00
```

## **Вариант № 2. «Управление задачами»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, списки клиентов, должностей и сотрудников «жестко» заданы в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списков клиентов, должностей и сотрудников (источником данных может быть список в памяти, файл, база данных и проч.);

- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку списка клиентов, списка должностей и списка сотрудников из текстовых файлов «customers.dat», «positions.dat» и «employees.dat» соответственно.

В каждой строке текстового файла «customers.dat» содержится запись о клиенте в следующем формате:

Name|ContactEmail|FullPhone|ContactPerson

Т. е. перечислены значения атрибутов, разделенные символом «|».

Пример файла, содержащего данные о 5 клиентах:

Analog Devices|andev@outlook.com|928-554-9897|Vickie E. Chaney  
 Traco Power|contact@tracopower.com|414-203-8348|Andrea R. Randall  
 Electronicon|adm@Electronicon.com|212-908-7687|Benny S. Treat  
 ATMEL|pub@atmel.org|610-910-4568|Karen K. Richardson  
 Fabrimex|pub@fabrimex.com|719-776-7470|Jacqueline G. Geter

В каждой строке текстового файла «positions.dat» содержится запись о должности в следующем формате:

Code|Name|BaseHourlyRate

Т. е. перечислены значения атрибутов, разделенные символом «|».

Часовая ставка (поле «BaseHourlyRate») приведено с точностью до 2 знаков после запятой, десятичным разделителем в является точка.

Пример файла, содержащего данные о 4 должностях:

DEV01|Junior Developer|5.00  
 DEV03|Senior Developer|11.50  
 TST04|Test Lead|9.00  
 AN01|Junior Analyst|6.50

В каждой строке текстового файла «employees.dat» содержится запись о должности в следующем формате:

Number|FullName|PositionCode|Rating

Т. е. перечислены значения атрибутов, разделенные символом «|».

Пример файла, содержащего данные о 5 сотрудниках:

M.01.074|Brice Lambson|DEV03|3

M.01.231|Andrew Peters|DEV01|1

M.01.264|Bill Wagner|DEV01|4

M.02.108|Robert Martin|TST04|5

M.05.220|Rowan Miller|AN01|3

M.05.217|James Gurtz|AN01|2

### **Вариант № 3. «Учебная программа»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, списки степеней и курсов «жестко» заданы в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списков степеней и курсов (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку списка клиентов, списка должностей и списка сотрудников из текстовых файлов «degrees.dat» и «courses.dat».

В каждой строке текстового файла «degree.dat» содержится запись о степени в следующем формате:

Code:Title:CreditsRequired:SpecialCoursesRequired

Т. е. перечислены значения атрибутов, разделенные символом «:».

Пример файла, содержащего данные о 2 степенях:

09.03.01:Бакалавр техники и технологии 'Информатика и выч.техника':100:3

09.03.04:Бакалавр техники и технологии 'Программная инженерия':130:5

В каждой строке текстового файла «courses.dat» содержится запись об учебном курсе в следующем формате:

Code:Title:LectureHours:PracticeHours:IsSpecial:HasExam:HasCourse  
Paper: Prerequisites

Т. е. перечислены значения атрибутов, разделенные символом «:». Значение «истина» для атрибутов-флагов «IsSpecial», «HasExam», «HasCoursePaper» записывается как «Y», значение «ложь» – как «N». Поле «Prerequisites» необязательное. Если оно присутствует, то в нем через запятую перечислены регистрационные коды дисциплин, которые должны быть изучены перед изучением текущей дисциплины.

Пример файла, содержащего данные о 4 курсах:

K1-004:Программирование:36:54:N:Y:N

K4-001:Математический анализ:54:54:N:Y:N

K1-042:Протоколы вычислительных сетей:18:18:N:N:Y

K1-017:Логика и теория алгоритмов:54:18:N:N:N:K4-001,K4-001

#### **Варианты № 4. «Начисление зарплаты»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, список должностей сотрудников «жестко» задан в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списка должностей (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку списка должностей из текстового файла «positions.dat» или из текстового файла «pos.csv». Реализованы должны быть оба механизма, но в приложении «жестко» должно задаваться использование одного из них.

В каждой строке текстового файла «positions.dat» содержится запись о должности в следующем формате:

Code|Name|BaseHourlyRate

Т. е. перечислены значения атрибутов, разделенные символом «|». Часовая ставка (поле «BaseHourlyRate») приведена с точностью до 2 знаков после запятой, десятичным разделителем в является точка. Пример файла, содержащего данные о 4 должностях:

```
DEV01|Junior Developer|5.00  
DEV03|Senior Developer|11.50  
TST04|Test Lead|9.00  
AN01|Junior Analyst|6.50
```

В каждой строке текстового файла «pos.csv» содержится запись о должности в следующем формате:

```
Department;Name;BaseHourlyRate;Currency
```

Т. е. перечислены значения атрибутов, разделенные символом «;». Часовая ставка (поле «BaseHourlyRate») приведена с точностью до 2 знаков после запятой, десятичным разделителем в является запятая. Пример файла, содержащего данные о 4 должностях:

```
M_Development;Junior Developer;5,00;RUR  
M_Development;Senior Developer;11,50;RUR  
M_Testing;Test Lead;9,00;RUR  
L_Analysis;Junior Analyst;6,50;RUR
```

### **Варианты № 5. «Качество работы»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, список номенклатур изделий «жестко» задан в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списка номенклатур (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;



- реализовать загрузку списка номенклатур из текстового файла «products.dat» или из текстового файла «catalog.txt». Реализованы должны быть оба механизма, но в приложении «жестко» должно задаваться использование одного из них.

В каждой строке текстового файла «products.dat» содержится запись о номенклатуре в следующем формате:

Article\Title\StandardTime

Т. е. перечислены значения атрибутов, разделенные символом «\». Норма рабочего времени (поле «StandardTime») приведена в виде целого числа – количества минут. Пример файла, содержащего данные о 2 номенклатурах:

ABC001\Lorem ipsum sid amet\252

ABC002\Consectetur adipisci velit\215

В каждой строке текстового файла «catalog.txt» содержится запись о номенклатуре в следующем формате:

Article;Title;StandardTime1;StandardTime2

Т. е. перечислены значения атрибутов, разделенные символом «;». Поля «StandardTime1» и «StandardTime2» заданы в формате «hh:mm:ss». Норма рабочего времени равна сумме этих двух компонент. Пример файла, содержащего данные о 2 номенклатурах:

ABC001;Lorem ipsum sid amet;03:12:00;01:00:00

ABC002;Consectetur adipisci velit;02:45:00;00:50:00

## **Вариант №6. «Лаборатория»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, список типовых анализов «жестко» задан в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- переработать приложение так, чтобы его «ядро» не зависело от способа получения списка типовых анализов (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;

- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку списка типовых анализов из текстового файла «tests.dat» или из текстового файла «reg.txt». Реализованы должны быть оба механизма, но в приложении «жестко» должно задаваться использование одного из них.

В каждой строке текстового файла «tests.dat» содержится запись о типовом тесте в следующем формате:

Name|Category|MinResult|MaxResult|UnitSymbol|UnitName

Т. е. перечислены значения атрибутов, разделенные символом «|». Категория представляет собой либо строку «HI» (важный тест), либо «NR» (обычный). Минимальное и максимальное значения допустимого диапазона значений (поля «MinResult» и «MaxResult») приведены с точностью до 2 знаков после запятой, десятичным разделителем в является точка. Пример файла, содержащего данные о 2 типовых тестах:

Кинематическая вязкость масла при 40°C|HI|  
28.80|35.20|мм<sup>2</sup>/с|мм<sup>2</sup>/с

Эмульгируемость с водой|NR|0.10|0.30|см<sup>3</sup>|см<sup>3</sup>

В каждой строке текстового файла «reg.txt» о типовом тесте в следующем формате:

Category;Name;MinResult;ResultRangeLength;UnitSymbol;UnitName;

Т. е. перечислены значения атрибутов, разделенные символом «;». Категория представляет собой либо строку «1» (важный тест), либо «0» (обычный). Допустимый диапазон значений задается начальным значением «MinResult» и шириной диапазона «ResultRangeLength». Поля «MinResult» и «ResultRangeLength» приведены с точностью до 1 знака после запятой, десятичным разделителем в является запятая. Пример файла, содержащего данные о 2 типовых тестах:

1;Кинематическая вязкость масла при 40°C;28,8;6,4;мм<sup>2</sup>/с;мм<sup>2</sup>/с;  
0;Эмульгируемость с водой;0,1;0,3;см<sup>3</sup>;см<sup>3</sup>;

## **Вариант №7. «Учебные сертификаты»**

В программе, разработанной в рамках выполнения лабораторной работы № 2, список студентов «жестко» задан в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

- в сущность «Студент (Student)» добавить атрибут «Дата рождения (BirthDate)»;
- переработать приложение так, чтобы его «ядро» не зависело от способа получения списка студентов (источником данных может быть список в памяти, файл, база данных и проч.);
- обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
- спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
- реализовать загрузку студентов из текстового файла «std-form-a.csv» или из текстового файла «std-form-b.dat». Реализованы должны быть оба механизма, но в приложении «жестко» должно задаваться использование одного из них.

В каждой строке текстового файла «std-form-a.csv» содержится запись о студенте в следующем формате:

RegNumber,FullName,BirthDate

Т. е. перечислены значения атрибутов, разделенные символом «,». Дата рождения (поле «BirthDate») приведена в формате «dd.MM.yyyy» (здесь dd – день месяца, MM – номер месяца, yyyy – год). Пример файла, содержащего данные о 3 студентах:

CS-00345/14,Brice Lambson,22.03.1990

MM-12007/13,Andrew Peters,17.07.1989

CS-09711/13,James Gurtz,05.11.1985

В каждой строке текстового файла «std-form-b.dat» содержится запись о студенте в следующем формате:

RegNumber|BirthDate|FirstName|MiddleName|LastName

Т. е. перечислены значения атрибутов, разделенные символом «|». При этом фамилия студента хранится в поле «LastName», имя – в поле

«FirstName», отчество – в поле «MiddleName». Дата рождения (поле «BirthDate») приведена в формате «уууу-ММ-уууу» (здесь dd – день месяца, ММ – номер месяца, уууу – год). В этом формате последние 3 символа представляют код учебного подразделения и должны игнорироваться при загрузке. Пример файла, содержащего данные о 3 студентах:

CS-00345/14.17|Brice|S|Lambson|1990-03-22

MM-12007/13.17|Andrew|B|Peters|1989-07-17

CS-09711/13.17|James|C|Gurtz|1985-11-05

## **КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ**

1. Перечислите паттерны SOLID.
2. Раскройте суть принципа единственной обязанности.
3. Раскройте суть принципа открытости/закрытости.
4. Раскройте суть принципа подстановки Лискоу.
5. Раскройте суть принципа разделения интерфейса.
6. Раскройте суть принципа инверсии зависимостей.
7. Дайте характеристику слоистой архитектуры приложений.

## **ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ**

На выполнение домашней работы отводится 12 часов: 11 часов на выполнение работы, 1 час на подготовку и защиту отчета по домашней работе.

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), UML-диаграмма классов модели, этапы выполнения работы (со скриншотами), результаты выполнения работы, выводы.

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Романенко, В.В. Объектно-ориентированное программирование: учебное пособие / В.В. Романенко; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР).– Томск : Томский Государственный Университет Систем Управления и Радиоэлектроники, 2014.– 475 с. : ил. – Библиогр.: с. 442. : То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=480517>
2. Николаев, Е.И. Объектно-ориентированное программирование: учебное пособие / Е.И. Николаев; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждения высшего профессионального образования «Северо-Кавказский федеральный университет».– Ставрополь: СКФУ, 2015.– 225 с.: ил.– Библиогр. В кн.: То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=458133>
3. Суханов, М.В. Основы Microsoft .Net Framework и языка программирования С#: учебное пособие / М.В. Суханов, И.В. Бачурин, И.С. Майоров; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждения высшего профессионального образования Северный (Арктический) федеральный университет им. М.В. Ломоносова.– Архангельск: ИД САФУ, 2014 – 97 с.: схем., табл., ил.– Библиогр. В кн.– ISBN 978-5-261-00934-4: То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=312313>

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Б.Албахари С# 6.0. Справочник. Полное описание языка: учеб. пособие / Б. Албахари, Дж. Албахари. – 6-е издание. М: O'reilly, 2016 г.– 1040 с.