Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем

и технологий»

**Отчёт**

по лабораторной работе №21

по дисциплине ***«Объектно-ориентированные технологии***

***программирования и стандарты проектирования»***

тема: «Объектно-Ориентированного Программирования на языке Java. Абстракция»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701119  Сташкевич Александр Сергеевич |
| Преподаватель: |  | Иванченко Виктор Викторович |

2020-2021 учебный год

**ЛАБОРАТОРНАЯ РАБОТА №21**

**Объектно-Ориентированного Программирования на языке Java. Абстракция**

Цель работы:

Научиться грамотно анализировать предметную область и с помощью абстракции выделять существенные детали, на базе которых в дальнейшем проектируются классы и объекты будущей программной системы согласно методологии ООП, а так же практически закрепить данные навыки при решении соответствующих задач (бизнес проблем).

Основное задание:

Необходимо решить задачу с использованием методологии ООП. Для чего необходимо подобрать самостоятельно соответствующую проблемную (предметную/доменную) область, которая базируется на объектах и событиях реального мира (примеры соответствующих предметных областей приведены ниже). Спроектировать классы (собственные пользовательские типы данных) в языке Java для программного представления данных объектов и основной логики будущей программной системы. Система должна решать, как минимум, два полезных действия и иметь дополнительно следующие опции:

−не менее 3 разнообразных классов предметной области;

−не менее 5 атрибутов (состояния)и методов (поведения)в классе-сущности;

−не менее 3 методов, которые реализуют бизнес-логику программы, в со-ответствующих функциональных классах;

−хранить глобальные характеристики системы или характеристики уровня отдельных классов.

На базе спроектированной программной системы реализовать программу и продемонстрировать её работоспособность.

Требования к выполнению:

1)Необходимо спроектировать и реализовать UML-диаграмму взаимодействия классов и объектов разрабатываемой программной системы с отображением всех связей (отношений) между классами и объектами.

2)При проектировании и разработке системы необходимо полностью использовать своё объектно-ориентированное воображение и по максимум использовать возможности, которые предоставляет язык программирования Java для реализации ООП-методологии.

3)Основные классы системы должны быть самодостаточными, т.е. не зависеть, к примеру, от консоли! Любые типы отношений между классами должны при-меняться обосновано и лишь тогда, когда это имеет смысл.

4)При выполнении задания необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый, легко поддерживаемый и читаемый код.

5)Также рекомендуется придерживаться Single Responsibility Principle, SRP(принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.

6)Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: by.bntu.fitr.poisit.nameofstudent.nameofpro-ject.

7)В соответствующих компонентах бизнес-логики необходимо предусмотреть «защиту от дурака».

8)На базе спроектированной программной системы реализуйте простейшее ин-терактивное консольное приложение. Используйте при реализации архитектурный шаблон проектирования Model-View-Controller,MVC.

9)Программа должна обязательно быть снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчиков, название бригады (если есть), номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.

10)Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем.

11)Интерфейс программы и комментарии должны быть на английском языке.

12)Необходимо проверить все тестовые случаи работы основной бизнес-логики программы.

13)При разработке программ придерживайтесь соглашений по написанию кода на Java(JavaCode-Convention)!!!

**Решение**

Выделим основные сущности нашей предметной области: классы-сущности Компьютер(Computer), общий класс Магазин(Store), класс бизнес логики Менеджер(Manager), утилитный класс для вывода данных Принтер (Printer), Конвертация(Convert), Создания компьютеров (ComputerCreator), Инициализации компьютеров (ComputerInitialization) и класс-контроллер MainClass. Согласно предметной области спроектируем UML-диаграмму взаимодействия классов и объектов будущего приложения:

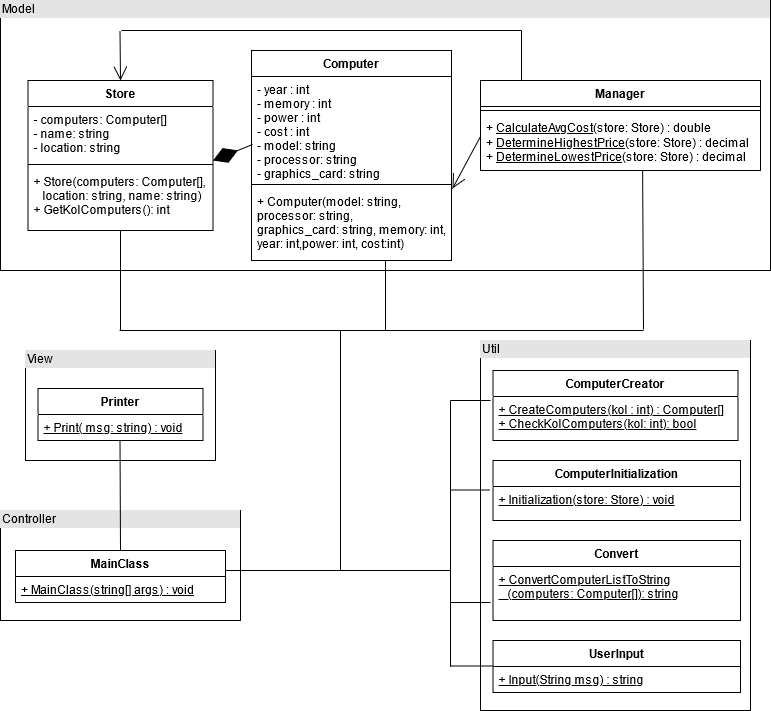


Рисунок 1 – UML- диаграмма приложения

**Результат выполнения основного задания:**

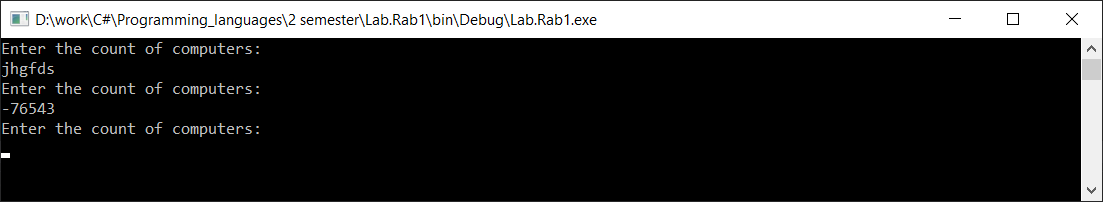


Рисунок 2 – Результат проверки программы на некорректность данных

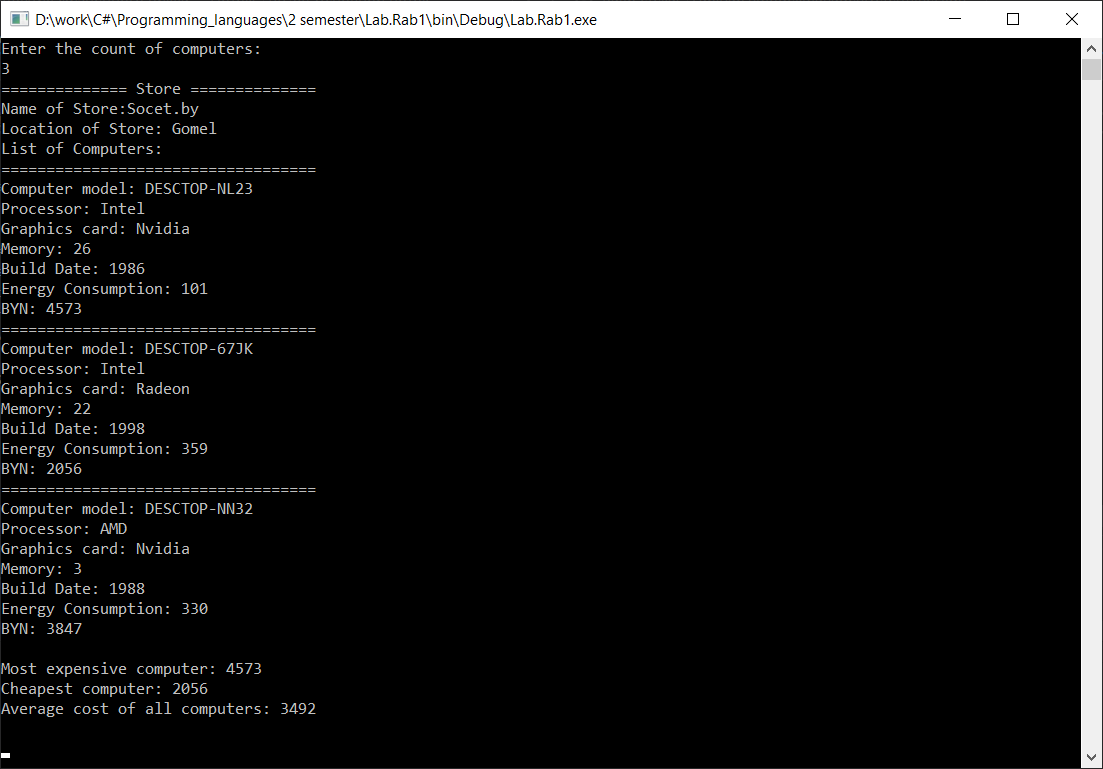


Рисунок 3 – Результат выполнения основного задания

Ответы на контрольные вопросы:

1)Какова была главная идея создания процедурных языков программирования? На что эти языки программирования ставили акцент и что являлось исходным понятием методологии?

*Процедурное программирование описывает структуры данных и алгоритмы.*

*Процедурное (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 40-х гг.*

*Процедурный язык программирования предоставляет возможность программисту определять каждый шаг в процессе решения задачи. Особенность таких языков программирования состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом.*

*Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.*

2)К чему не были готовы процедурные языки программирования в начале 2000 годов? Какой был кризис в сфере разработки программного обеспечения?

*После появления достаточно доступных персональных компьютеров произошел кризис программного обеспечения. Заключался он в том, что чем мощнее становились компьютеры, тем мощнее становились программы и они становились более неуправляемыми из-за много тысячных строк кода. Что вело за собой сложность поддержания кода, масштабирования его и т.д. Если раньше компьютеры использовались раньше для вычислительных задач, то сейчас все поменялось в сторону манипуляций данными.*

3)Суть объектно-ориентированного подхода?

*Главная идея ООП–всё состоит из объектов! Программа, написанная с использованием ООП, состоит из множества объектов, и все эти объекты взаимодействуют между собой посредством посылке (передачи) сообщений друг другу. ООП и реальный мир не могут существовать раздельно!*

*Основная суть объектно-ориентированного программирования состоит в том, что он нацелен на повторное использование кода. Для этого программа представляется как совокупность объектов.*

4)Опишите базовые концепции, которые лежат в основе методологии ООП.

*Основные* [*принципы*](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF) *структурирования в случае ООП:*

1. [*абстракция*](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
2. [*инкапсуляция*](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))
3. [*наследование*](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))
4. [*полиморфизм*](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))

5)Что такое объект в реальном мире? Чем он характеризуется в реальном мире?(тип, состояние, поведение, уникальность, эквивалентность)

*Объект –это мыслимая или реальная сущность, обладающая характерным поведением, отличительными характеристиками и являющая важной в предметной области. Каждый объект имеет состояние, обладает некоторым хорошо определённым поведением и уникальной идентичностью.*

*Состояние (state)(синонимы: параметры, аспекты, характеристики, свойства, атрибуты, ...) –совокупный результат поведения объекта: одно из стабильных условий, в которых объект может существовать, охарактеризованных количественно; в любой конкретный момент времени состояние объекта включает в себя перечень параметров объекта и текущее значение этих параметров.*

*Поведение (behavior)–действия и реакции объекта, выраженные в терминах передачи сообщений и изменения состояния; видимая извне и воспроизводимая активность объекта.*

*Уникальность (identity)–эта природа объекта; то, что отличает один объект от других. В машинном представлении уникальность объекта –это адрес размещения объекта в памяти. Следовательно, уникальность объекта состоит в том, что всегда можно определить, указывают две ссылки на один и тот же объект, или на разные объекты.*

*Эквивалентность (equivalence)объектов –это когда состояние сравниваемых объектов совпадает, т.е. соответствующие атрибуты состояния объектов равны.*

6)Что такое объект в программировании и для чего он вводится при моделировании будущей программной системы?

*ООП использует в качестве базовых элементов объекты, а не алгоритмы.*

*Использование объектного подхода –это естественный путь решать задачи реального мира с помощью компьютера. Это просто, понятно и проверено временем.*

*Методология ООП переводит программирование в моделирование. Другими словами, ООП позволяет упрощать сложные вещи через моделирование.*

*Программные объекты могут представлять собой объекты реального мира или быть полностью абстрактными объектами, которые могут существовать только в рамках программы.*

7)Чем характеризуется объект в программном(виртуальном)мире? Как реализуются данные вещи в объектно-ориентированных языках программирования?

*1) Object type*

*2) Object states*

*3) Object behaviors*

*4) Uniqueness*

*5) Equivalence*

8)Какая разница между уникальностью и эквивалентностью?

*Если две ссылочные переменные указывают на один и тот же объект – то они не являются уникальными.*

*Если у нас физически разные объекты, но их содержимое одинаково – то это эквивалентность.*

9)Какую роль выполняет класс в ООП?

*Чтобы создать программный объект или группу объектов необходимо вначале описать где-то его(их) характеристики и шаблон поведения. Для этих целей в ООП существуют классы (classes). Формально, класс–это шаблон поведения объектов определённого типа с определёнными параметрам, которые описывают состояние объекта.*

10)Как и что можно описать в Java-классе?

*Любой объект может обладать двумя основными характеристиками:*

*состояние - некоторые данные, которые хранит объект.*

*поведение - действия, которые может совершать объект.*

11)Что представляет собой абстракция в ООП?

*Абстракция (абстрагирование)в программировании – процесс выделения(отделения от всего остального)из предметной(проблемной)области существенных деталей и их реализация в программной среде.*

12)Простейшая форма абстракции в ООП?

*Объект –простейший вид абстракции в ООП.*

13)Какие разновидности классов существуют в ООП?

*В Java есть 4 вида классов внутри другого класса:*

1. *Вложенные внутренние классы – нестатические классы внутри внешнего класса.*
2. *Вложенные статические классы – статические классы внутри внешнего класса.*
3. *Локальные классы Java – классы внутри методов.*
4. *Анонимные Java классы – классы, которые создаются на ходу.*

14)Каким способом можно создать объект (экземпляр класса) в языке Java?

Car merin = new Car();

15)Как можно обратиться к полям и методам объекта в языке Java?

*Для обращения к методу или полю класса нужно воспользоваться точечной нотацией.*

16)Какова разница между классом и объектом в языке Java?

*Понятие класса в ООП – это тип данных (такой же как, например, Real или String), а объект – конкретный экземпляр класса (его копия), хранящийся в памяти компьютера как переменная соответствующего типа.*

17)Перечислите преимущества и недостатки объектно-ориентированного программирования.

*Использование парадигмы ООП в программировании помогает:*

1. *Уменьшить сложность программного обеспечения (ПО);*
2. *увеличить производительность труда программистов и скорость разработки ПО;*
3. *повысить надёжность ПО*
4. *обеспечить возможность лёгкой модификации отдельных компонентов ПО без изменения остальных его частей;*
5. *обеспечить возможность повторного использования отдельных компонентов ПО;*

*У парадигмы ООП также есть некоторые минусы:*

1. *много времени и сил тратится на размышления об абстракциях и шаблонах проектирования вместо решения реальных проблем;*
2. *на чтение объектно-ориентированного кода тратится больше времени, чем на его написание;*
3. *увеличивается объём кода;*
4. *увеличиваются накладные расходы на ресурсы;*
5. *снижается скорость выполнения программ;*
6. *требует много труда, которое окупается на больших проектах.*

18)Что такое UML?

*UML–унифицированный язык моделирования (Unified Modeling Language) –это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.*

19)Что отображает и зачем нужна UML-диаграмма классов?

*Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Язык UML применяется не только для проектирования, но и с целью документирования, а также эскизирования проекта.*

20)Как с помощью UML можно описать классы и их взаимосвязь друг с другом (т.е. описать)?

*Словарь UML включает три вида строительных блоков: диаграммы, сущности и связи. Сущности–это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей*

*ПРИЛОЖЕНИЕ А*

**Листинг файлов:**

**Файл Computer.cs**

namespace Lab.Rab1.Model

{

public class Computer

{

public const int MAX\_YEAR = 2021;

public const int MIN\_YEAR = 1981;

public const int MIN\_MEMORY = 2;

public const int MAX\_MEMORY = 64;

public const int MIN\_POWER = 100;

public const int MAX\_POWER = 500;

public const int MIN\_COST = 500;

public const int MAX\_COST = 5000;

private int year;

private int memory;

private int power;

private int cost;

public string Model { get; set; }

public string Processor { get; set; }

public string Graphics\_card { get; set; }

public Computer(string model = "DESKTOP-USER", string processor = "Intel", string graphics\_card = "Nvidia",

int memory = 4, int year = 2018, int power = 120, int cost = 890)

{

Model = model;

Processor = processor;

Graphics\_card = graphics\_card;

Memory = memory;

Year = year;

Power = power;

Cost = cost;

}

public int Memory

{

get => memory;

set

{

if (value >= MIN\_MEMORY && value <= MAX\_MEMORY)

{

memory = value;

}

}

}

public int Year

{

get => year;

set

{

if (value >= MIN\_YEAR && value <= MAX\_YEAR)

{

year = value;

}

}

}

public int Power

{

get => power;

set

{

if(value>=MIN\_POWER && value <= MAX\_POWER)

{

power = value;

}

}

}

public int Cost

{

get => cost;

set

{

if(value > MIN\_COST && value < MAX\_COST)

{

cost = value;

}

}

}

public override string ToString()

{

return $"Computer model: {Model}\nProcessor: {Processor}\n" +

$"Graphics card: {Graphics\_card}\nMemory: {Memory}\n" +

$"Build Date: {Year}\nEnergy Consumption: {Power}\n" +

$"BYN: {Cost}\n";

}

}

}

**Файл Manager.cs**

namespace Lab.Rab1.Model

{

public class Manager

{

public static double CalculateAvgCost(Store store)

{

double avgCost = 0;

foreach(Computer computer in store.Computers)

{

avgCost += computer.Cost;

}

return avgCost / store.Computers.Length;

}

public static decimal DetermineHighestPrice(Store store)

{

decimal highestPrice = store.Computers[0].Cost;

foreach(Computer computer in store.Computers)

{

if(computer.Cost>highestPrice)

{

highestPrice = computer.Cost;

}

}

return highestPrice;

}

public static decimal DetermineLowestPrice(Store store)

{

decimal lowestPrice = store.Computers[0].Cost;

foreach (Computer computer in store.Computers)

{

if (computer.Cost < lowestPrice)

{

lowestPrice = computer.Cost;

}

}

return lowestPrice;

}

}

}

**Файл Store.cs**

using System.Text;

namespace Lab.Rab1.Model

{

public class Store

{

public const string DEFAULT\_NAME\_STORE = "Socet.by";

public const string DEFAULT\_LOCATION = "Gomel";

public Computer[] Computers { get; set; }

public string Name { get; set; }

public string Location { get; set; }

public Store(Computer[] computers, string location = DEFAULT\_LOCATION, string name = DEFAULT\_NAME\_STORE)

{

Computers = computers;

Location = location;

Name = name;

}

public int GetKolComputers()

{

return Computers.Length;

}

public override string ToString()

{

StringBuilder listComputers = new StringBuilder();

listComputers.Append("Name of Store:").Append(Name).Append("\n");

listComputers.Append("Location of Store: ").Append(Location).Append("\n");

listComputers.Append("List of Computers: \n");

if (GetKolComputers() == 0)

{

listComputers.Append("No computers in store!\n");

}

else

{

listComputers.Append(Util.Convert.ConvertComputerListToString(Computers));

}

return listComputers + "";

}

}

}

**Файл MainClass.cs**

using Lab.Rab1.Model;

using Lab.Rab1.Util;

using Lab.Rab1.View;

using System;

namespace Lab.Rab1.Controller

{

class MainClass

{

public static void Main(string[] args)

{

int size;

bool checkSize;

do

{

checkSize = Int32.TryParse(UserInput.Input("Enter the count of computers: "), out size);

}

while (checkSize != true || size < 1);

Store store = new Store(ComputerCreator.CreateComputers(size));

ComputerInitialization.Initialization(store);

Printer.Print($"============== Store ==============\n" +

$"{store.ToString()}\n" +

$"Most expensive computer: {Manager.DetermineHighestPrice(store)}\n" +

$"Сheapest computer: {Manager.DetermineLowestPrice(store)}\n" +

$"Average cost of all computers: {Manager.CalculateAvgCost(store)}\n");

Console.ReadKey();

}

}

}

**Файл ComputerCreator.cs**

using Lab.Rab1.Model;

namespace Lab.Rab1.Util

{

class ComputerCreator

{

public const int MIN\_KOL\_COMPUTERS = 1;

public const int MAX\_KOL\_COMPUTERS = 25;

public static Computer[] CreateComputers(int kol)

{

if (!CheckKolComputers(kol))

{

return new Computer[0];

}

Computer[] computers = new Computer[kol];

for (int i = 0; i < kol; i++)

{

computers[i] = new Computer();

}

return computers;

}

public static bool CheckKolComputers(int kol)

{

return kol >= MIN\_KOL\_COMPUTERS & kol <= MAX\_KOL\_COMPUTERS;

}

}

}

**Файл ComputerInitialization.cs**

using Lab.Rab1.Model;

using System;

namespace Lab.Rab1.Util

{

class ComputerInitialization

{

public static string[] namesOfModel = {"DESCTOP-NN32","DESCTOP-NL23","DESCTOP-GH56","DESCTOP-99LS","DESCTOP-67JK"};

public static string[] namesOfProcessor = { "Intel", "AMD" };

public static string[] namesOfGraphicsCard = { "Nvidia", "Radeon" };

public const int MAX\_YEAR = 2021;

public const int MIN\_YEAR = 1981;

public const int MIN\_MEMORY = 2;

public const int MAX\_MEMORY = 64;

public const int MIN\_POWER = 100;

public const int MAX\_POWER = 500;

public const int MIN\_COST = 500;

public const int MAX\_COST = 5000;

public static void Initialization(Store store)

{

Random random = new Random();

foreach (Computer computer in store.Computers)

{

computer.Model = namesOfModel[random.Next(namesOfModel.Length)];

computer.Processor = namesOfProcessor[random.Next(namesOfProcessor.Length)];

computer.Graphics\_card = namesOfGraphicsCard[random.Next(namesOfGraphicsCard.Length)];

computer.Power = random.Next(MIN\_POWER, MAX\_POWER);

computer.Cost = random.Next(MIN\_COST,MAX\_COST);

computer.Year = random.Next(MIN\_YEAR, MAX\_YEAR);

computer.Memory = random.Next(MIN\_MEMORY,MAX\_MEMORY);

}

}

}

}

**Файл Convert.cs**

using Lab.Rab1.Model;

using System;

namespace Lab.Rab1.Util

{

public class Convert

{

public static string ConvertComputerListToString(Computer[] computers)

{

string msg = "";

foreach(Computer computer in computers)

{

msg += "\n" +computer + String.Empty.PadLeft(35, '=');

}

return msg;

}

}

}

**Файл UserInput.cs**

using System;

namespace Lab.Rab1.Util

{

class UserInput

{

public static string Input(String msg)

{

Console.WriteLine(msg);

return Console.ReadLine();

}

}

}

**Файл Printer.cs**

using System;

namespace Lab.Rab1.View

{

public class Printer

{

public static void Print(string msg)

{

Console.WriteLine(msg);

}

}

}