Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 13

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Классы и объекты»

Выполнил: ст. гр. ТУУ-111

Кругликов Е.А.

Вариант №11

Проверил: к.т.н., доц. Сафронов А.И.

1. **Цель работы**

В среде *Microsoft* *Visual Studio* на языке *Visual C#* в консольном режиме составить программное обеспечение для решения типовых задач программирования по тематике «Классы и объекты» («*Classes and Objects*»), заданных по варианту.

1. **Формулировка задачи**

**Вариант №11**

Создайте классы «Принтер» (возможное имя «clsPrinter») и «Картридж» (возможное имя «clsCartridge»).

**Поля (fields):**

* Для принтера:
  + идентификатор (int);
  + производитель (enum);
  + тип (enum);
  + модель (string);
  + картридж (clsCartridge);
  + дата приобретения (long);
  + гарантийный срок службы в годах (byte).
* Для картриджа:
  + идентификатор (int);
  + производитель (enum);
  + принтер (clsPtinter);
  + модель (string);
  + дата приобретения (long);
  + дата заправки (long);
  + количество доступных для печати страниц (int).

**Конструктор (instance constructor):**

* Для принтера:
  + конструктор с параметрами соответствующих типов в нужном количестве (разбор входной информации перед конструированием экземпляра класса).
* Для картриджа:
  + только пустой конструктор.

**Методы (methods):**

* Для принтера:
  + Печать указанного количества страниц (void).
  + Подсчёт количества дней до истечения гарантийного срока принтера (int).
  + Проверка наличия картриджа (bool).
  + Проверка наличия непустого картриджа (bool).
* Для картриджа:
  + Подсчёт оставшегося количества страниц, доступных для печати (int).
  + Отклик об оригинальности картриджа (один и тот же производитель, что и у принтера; не перезаправленный, bool).

**Операторы (operators):**

* Для принтера:
  + напечатать страницу (++);
  + напечатать указанное количество страниц (+);
  + извлечь картридж (--);
  + установить картридж (!).
* Для картриджа:
  + очистить печатающий картридж (!);
  + заправить печатающий картридж (~).

**Хранение элементов:**

* принтеры хранятся в файле с возможностью записи (prn.txt);
* картриджи хранятся в файле с возможностью записи (crt.txt).

**Сопоставление (Equals()):**

* Принтеры считать равными, если они от одного и того же производителя, а также одного и того же типа.

**Вывод (ToString()):**

* Для принтера:
  + Лазерный Hewlett-Packard LaserJet M111a (19.10.2017)
* Для картриджа:
  + $$ 1. Hewlett-Packard CE310A (1000 стр.) [19.10.2017] $$

**Хэширование (GetHashCode()):**

* Для принтера:
  + Сумма идентификатора, всех цифр даты приобретения, идентификаторов типа и производителя.
* Для картриджа:
  + Сумма количества доступных для печати страниц, идентификатора, порядковых номеров в алфавите всех букв из модели, а также цифр из модели.

1. **Блок-схема алгоритма**

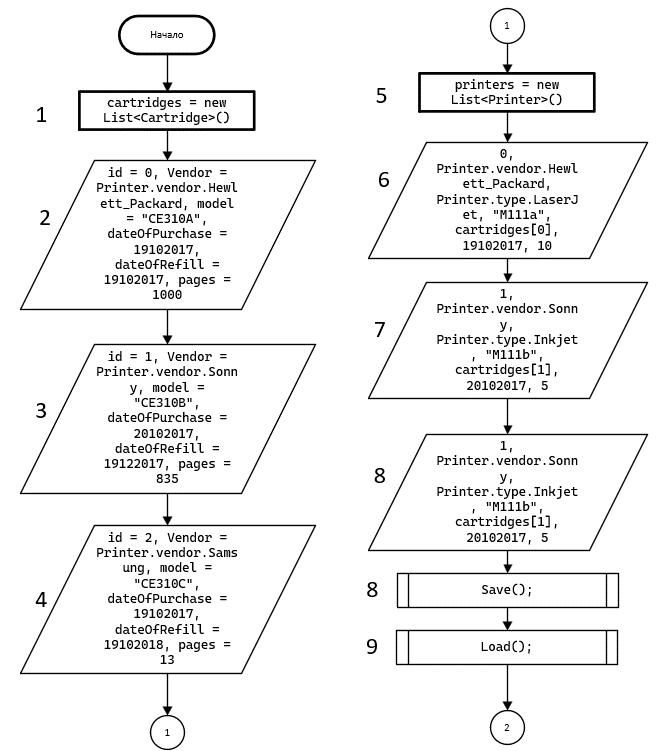
****

Рисунок 1 – Метод Main класса Program (часть 1)

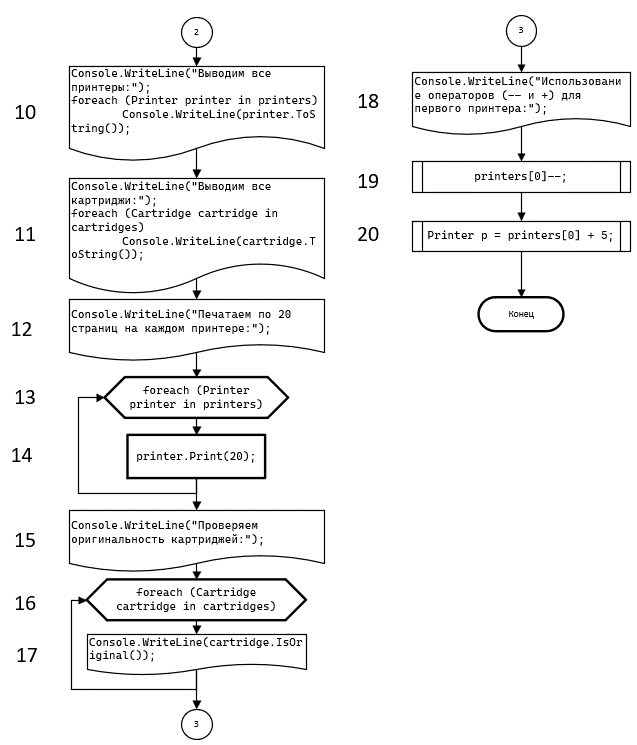
****

Рисунок 2 – Метод Main класса Program (часть 2)

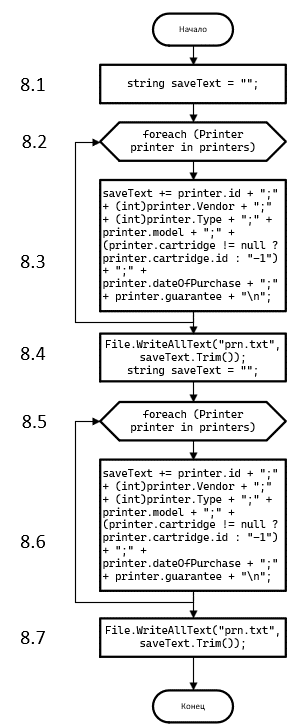
****

Рисунок 3 – Метод Save класса Program

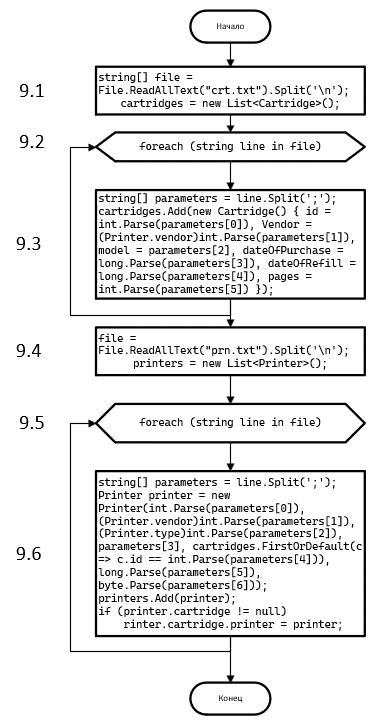
****

Рисунок 4 – Метод Load класса Program

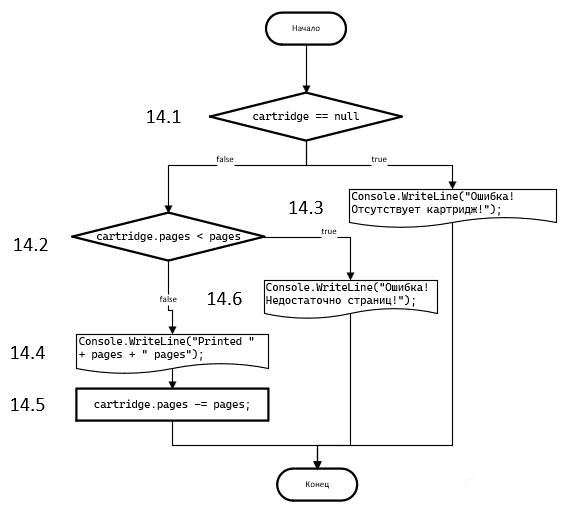


Рисунок 4 – Метод Print класса Printer

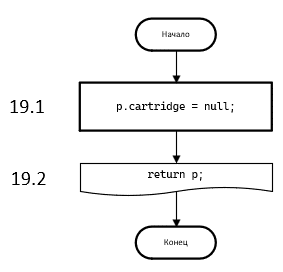


Рисунок 5 – Оператор -- класса Printer

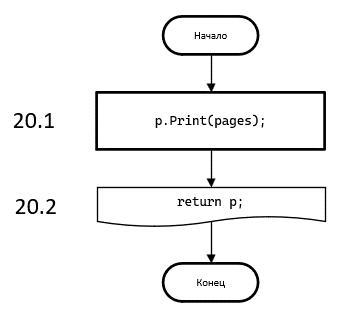


Рисунок 6 – Оператор + класса Program

1. **Подбор тестовых примеров**

Создаём три экземпляра класса Cartridge со следующими данными:

1) Идентификатор – 0;

Производитель – Hewlett\_Packard;

Модель – CE310A;

Дата покупки – 19.10.2017;

Дата заправки – 19.10.2017;

Страницы – 1000.

2) Идентификатор – 1;

Производитель – Sonny;

Модель – CE310B;

Дата покупки – 20.10.2017;

Дата заправки – 19.12.2017;

Страницы – 835.

3) Идентификатор – 0;

Производитель – Samsung;

Модель – CE310C;

Дата покупки – 19.10.2017;

Дата заправки – 19.10.2018;

Страницы – 13.

Создаём три экземпляра класса Printer со следующими данными:

1) Идентификатор – 0;

Производитель – Hewlett\_Packard;

Тип – LaserJet;

Модель - M111a;

Картридж – первый созданный картридж;

Дата покупки – 19.10.2017;

Гарантия – 10.

2) Идентификатор – 1;

Производитель – Sonny;

Тип – Inkjet;

Модель - M111b;

Картридж – второй созданный картридж;

Дата покупки – 20.10.2017;

Гарантия – 5.

3) Идентификатор – 0;

Производитель – Samsung;

Тип – \_3D;

Модель - M111c;

Картридж – третий созданный картридж;

Дата покупки – 19.10.2017;

Гарантия – 3.

* 1. **Метод Save:**

Сериализация экземпляров класса, чтобы предусмотреть возможность записи данных при использовании текстового файла.

* 1. **Метод Load:**

Десериализация экземпляров класса, чтобы предусмотреть возможность чтения данных при использовании текстового файла.

* 1. **Метод Print:**

Печатаем страницы при наличии заправленного картриджа.

* 1. **Оператор --:**

Извлекаем картридж из принтера.

* 1. **Оператор +**

Печатаем указанное количество страниц (в примере – 5).

1. **Листинг**

namespace Printers

{

internal class Program

{

public static List<Printer> printers;

public static List<Cartridge> cartridges;

static void Main(string[] args)

{

cartridges = new List<Cartridge>()

{

new Cartridge() { id = 0, Vendor = Printer.vendor.Hewlett\_Packard, model = "CE310A", dateOfPurchase = 19102017, dateOfRefill = 19102017, pages = 1000 },

new Cartridge() { id = 1, Vendor = Printer.vendor.Sonny, model = "CE310B", dateOfPurchase = 20102017, dateOfRefill = 19122017, pages = 835 },

new Cartridge() { id = 2, Vendor = Printer.vendor.Samsung, model = "CE310C", dateOfPurchase = 19102017, dateOfRefill = 19102018, pages = 13 },

};

printers = new List<Printer>()

{

new Printer(0, Printer.vendor.Hewlett\_Packard, Printer.type.LaserJet, "M111a", cartridges[0], 19102017, 10),

new Printer(1, Printer.vendor.Sonny, Printer.type.Inkjet, "M111b", cartridges[1], 20102017, 5),

new Printer(2, Printer.vendor.Samsung, Printer.type.\_3D, "M111c", cartridges[2], 19102017, 3),

};

Save();

Load();

Console.WriteLine("Выводим все принтеры:");

foreach (Printer printer in printers)

Console.WriteLine(printer.ToString());

Console.WriteLine("Выводим все картриджи:");

foreach (Cartridge cartridge in cartridges)

Console.WriteLine(cartridge.ToString());

Console.WriteLine("Печатаем по 20 страниц на каждом принтере:");

foreach (Printer printer in printers)

printer.Print(20);

Console.WriteLine("Проверяем оригинальность картриджей:");

foreach (Cartridge cartridge in cartridges)

Console.WriteLine(cartridge.IsOriginal());

Console.WriteLine("Использование операторов (-- и +) для первого принтера:");

printers[0]--;

Printer p = printers[0] + 5;

}

public static void Save()

{

string saveText = "";

foreach (Printer printer in printers)

{

saveText += printer.id + ";" + (int)printer.Vendor + ";" + (int)printer.Type + ";" + printer.model + ";" + (printer.cartridge != null ? printer.cartridge.id : "-1") + ";" + printer.dateOfPurchase + ";" + printer.guarantee + "\n";

}

File.WriteAllText("prn.txt", saveText.Trim());

saveText = "";

foreach (Cartridge cartridge in cartridges)

{

saveText += cartridge.id + ";" + (int)cartridge.Vendor + ";" + cartridge.model + ";" + cartridge.dateOfPurchase + ";" + cartridge.dateOfRefill + ";" + cartridge.pages + "\n";

}

File.WriteAllText("crt.txt", saveText.Trim());

}

public static void Load()

{

string[] file = File.ReadAllText("crt.txt").Split('\n');

cartridges = new List<Cartridge>();

foreach (string line in file)

{

string[] parameters = line.Split(';');

cartridges.Add(new Cartridge() { id = int.Parse(parameters[0]), Vendor = (Printer.vendor)int.Parse(parameters[1]), model = parameters[2], dateOfPurchase = long.Parse(parameters[3]), dateOfRefill = long.Parse(parameters[4]), pages = int.Parse(parameters[5]) });

}

file = File.ReadAllText("prn.txt").Split('\n');

printers = new List<Printer>();

foreach (string line in file)

{

string[] parameters = line.Split(';');

Printer printer = new Printer(int.Parse(parameters[0]), (Printer.vendor)int.Parse(parameters[1]), (Printer.type)int.Parse(parameters[2]), parameters[3], cartridges.FirstOrDefault(c => c.id == int.Parse(parameters[4])), long.Parse(parameters[5]), byte.Parse(parameters[6]));

printers.Add(printer);

if (printer.cartridge != null)

printer.cartridge.printer = printer;

}

}

}

public class Printer

{

public enum vendor

{

Hewlett\_Packard, Sonny, Samsung

}

public enum type

{

LaserJet, Inkjet, \_3D

}

public int id;

public vendor Vendor;

public type Type;

public string model;

public Cartridge cartridge;

public long dateOfPurchase;

public byte guarantee;

public Printer(int id, vendor Vendor, type Type, string model, Cartridge cartridge, long dateOfPurchase, byte guarantee)

{

this.id = id;

this.Vendor = Vendor;

this.Type = Type;

this.model = model;

this.cartridge = cartridge;

this.dateOfPurchase = dateOfPurchase;

this.guarantee = guarantee;

}

public void Print(int pages)

{

if (cartridge == null)

{

Console.WriteLine("Ошибка! Отсутствует картридж!");

return;

}

if (cartridge.pages < pages)

{

Console.WriteLine("Ошибка! Недостаточно страниц!");

return;

}

Console.WriteLine("Printed " + pages + " pages");

cartridge.pages -= pages;

}

public int CountDaysUntillGuaranteeExpire()

{

return (int)(new DateTime((int)dateOfPurchase % 10000, (int)dateOfPurchase / 1000 % 100, (int)dateOfPurchase / 1000000) - DateTime.Today).TotalDays;

}

public bool CheckCartridge()

{

return cartridge != null;

}

public bool CheckNotEmptyCartridge()

{

if (cartridge == null)

return false;

return cartridge.pages > 0;

}

public static Printer operator ++(Printer p)

{

p.Print(1);

return p;

}

public static Printer operator +(Printer p, int pages)

{

p.Print(pages);

return p;

}

public static Printer operator --(Printer p)

{

p.cartridge = null;

return p;

}

public static Printer operator !(Printer p)

{

p.cartridge = new Cartridge();

return p;

}

public override bool Equals(object? obj)

{

if (obj == null || !(obj is Printer))

return false;

else

return Vendor == ((Printer)obj).Vendor && Type == ((Printer)obj).Type;

}

public override string ToString()

{

return $"{id}. {Vendor} {Type} {model} ({dateOfPurchase / 1000000}.{dateOfPurchase / 1000 % 100}.{dateOfPurchase % 10000})";

}

public override int GetHashCode()

{

int hash = id;

for (long i = dateOfPurchase; i > 0; i /= 10)

hash += (int)i % 10;

hash += (int)Type;

hash += (int)Vendor;

return hash;

}

}

public class Cartridge

{

public int id;

public Printer.vendor Vendor;

public Printer printer;

public string model;

public long dateOfPurchase;

public long dateOfRefill;

public int pages;

public int CountPages()

{

return pages;

}

public bool IsOriginal()

{

return printer.Vendor == Vendor && dateOfPurchase == dateOfRefill;

}

public static Cartridge operator !(Cartridge c)

{

c.pages = 0;

return c;

}

public static Cartridge operator ~(Cartridge c)

{

c.pages = 200;

c.dateOfRefill = DateTime.Today.Year + DateTime.Today.Month \* 10000 + DateTime.Today.Day \* 1000000;

return c;

}

public override bool Equals(object? obj)

{

if (obj == null || !(obj is Cartridge))

return false;

else

return Vendor == ((Cartridge)obj).Vendor;

}

public override string ToString()

{

return $"$$ {id}. {Vendor} {model} ({pages} стр.) [{dateOfPurchase / 1000000}.{dateOfPurchase / 1000 % 100}.{dateOfPurchase % 10000}] $$";

}

public override int GetHashCode()

{

int hash = pages;

hash += id;

foreach (char c in model)

{

hash += c;

}

return hash;

}

}

}

1. **Расчет тестовых примеров на ПК**

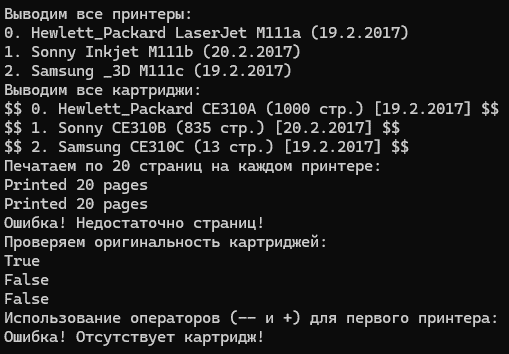


Рисунок 7 – Расчёт тестового примера

1. **Вывод по работе**

Научился создавать и работать с классами и их объектами, которые включают в себя конструкторы, методы, операторы и другие функциональные возможности. Реализовал методы, операторы и другие компоненты класса, соответствующие заданным требованиям, вывод текста в заданном формате, расчет хэша. Также включил в работу методы для чтения и записи текстовых надписей и их параметров в текстовый файл.