**ОТВЕТЫ НА ВОПРОСЫ К 4-ОЙ ЛАБОРАТОРНОЙ РАБОТЕ**

1) Для чего используют статические классы

1) при создании метода расширения (позволяют добавлять новые методы в уже существующие типы без создания нового производного класса)

2) для хранения совокупности связанных друг с другом статических методов

2) Что может содержать статический класс?

► экземпляры такого класса создавать запрещено

► не должен реализовывать никаких интерфейсов

► нельзя использовать в качестве поля, параметра метода или локальной переменной

► от него запрещено наследовать

► все элементы такого класса должны явным образом объявляться с модификатором static

► может иметь статический конструктор

► Компилятор не создает автоматически конструктор по умолчанию.

3) Что такое производный и базовый классы?

насл. все св-ва, методы, поля…

4) Как используют ключевое слово base? Ключевое слово **base** используется для доступа к членам базового из производного класса в следующих случаях:

* Вызов метода базового класса, который был переопределен другим методом.
* Определение конструктора базового класса, который должен вызываться при создании экземпляров производного класса.

Доступ к базовому классу разрешен только в конструкторе, методе экземпляра или методе доступа к свойству экземпляра.

5) В чем заключена основная задача наследования?

**Наследование** — это механизм создания нового класса на основе уже существующего. При этом к существующему классу могут быть добавлены новые элементы (данные и функции), либо существующие функции могут быть изменены. Основное назначение механизма наследования — *повторное использование кодов*, так как большинство используемых типов данных являются вариантами друг друга, и писать для каждого свой класс нецелесообразно. Также наследование формирует иерархию.

6) Пусть базовый класс содержит метод basefunc(), а производный класс не имеет метода с таким именем. Может ли объект производного класса иметь доступ к методу basefunc()? Если да, то при каких условиях?

да, при обычном наследовании !!! если basefunc() не private

7) Напишите объявление конструктора без аргументов для производного класса В, который будет вызывать конструктор без аргументов базового класса А.

class X  
 public X() {} *//к-ры не наслед.*  
 public X(int key) {}  
class Y : X  
 public Y(int key) {}  
 public Y() : ***base***(125) {}  
void main()  
 Y y0 = new Y(); *//авто- вызов к-р базового класса без парам*  
 Y y1 = new Y(125);

8) Что такое полиморфизм? Приведите пример.

- способность вызывать метод потомка через экземпляр предка

- способность программы идентично использовать объекты с одинаковым интерфейсом без информации о конкретном типе этого объекта

- переопределение членов баз. класса

**virtual** public void A\_method() {}  
переопределение виртуального метода в произв. классе:  
**override** public void A\_method() {}

вызов метода по типу ссылки *//просто*  
вызов метода по типу объекта *//virtual + override*

9) Определите назначение виртуальных функций.

**Виртуальная функция** — это функция-член, которую предполагается переопределить в производных классах.

**override** – изменение аб. или вирт. реализации  
**virtual** – разрешает переопределение (↑)

переопределение:  
\* те же парам.  
\* не static, не abstract

10) Кому доступны переменные с модификатором protected?

Базовому и производным классам

11) Наследуются ли переменные с модификатором private?

Нет

12) As, is – что это, как применяется?

**Is – можно преобразовать данное выр. в указ. тип? (t/f)**

Возвращает булевское значение, говорящее о том, можете ли вы преобразовать данное выражение в указанный тип  
int j = 12  
object boxed = j  
object obj = new Object()  
bool a = boxed is int //true  
bool b = obj is int //false

**As – позв. преобр. тип в опр. ссылочный тип:**

позволяет преобразовывать тип в определенный ссылочный тип с применением следующего синтаксиса:

операнд as <тип> //если оп-д имеет <тип>, мб неявно-, мб упакован

13) Поддерживает ли С# множественное наследование?

Для классов запрещено  
Для интерфейсов можно

14) Можно ли запретить наследование от класса?

**sealed** – запрещает другим классам наследовать от данного  
*sealed class B : A {} – класс В наследует от класса А, но никакие не могут насл. от В*при применении к методу/свойству, исп. OVERRIDE

15) Можно ли разрешить наследование класса, но запретить перекрытие метода?

**Да, можно**. Пример:

class A  
 public virtual int Sum()  
 { return 2 + 3; }  
class B : A  
 public **sealed override** int Sum() *//метод в незапеч. классе явл. запеч.* { return 2 + 5; }

16) Что такое абстрактный класс?

**Абстрактный класс** — базовый класс, который не предполагает создания экземпляров. Абстрактные классы реализуют на практике один из принципов ООП — полиморфизм. Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором описан, однако должен быть реализован для его неабстрактных потомков. Абстрактный класс – это фактически чертёж нормального класса без реализации.

\* предост. базовый функционал для насл.  
\* служит только для порождения потомков  
\* м. сод. абстр. методы  
\* м. опред. методы, пер., к-ры, св-ва  
\* нельзя созд. объекты абстр. Класса

abstract class Shapes  
 abstract public int Area();  
  
class Square : Shapes {  *//д. обесп. реализацию Area* public override int Area()  
 { return side\*side; }17) В каком случае вы обязаны объявить класс абстрактным?

если он содержит абстрактный метод

18) В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами?

* **абстрактный метод** не имеет реализации и должен быть определен в дочерних классах;
* **виртуальный метод** - может быть как абстрактным, так и иметь реализацию + разрешает переопределение

**Абстрактный** **класс** это **класс**, содержащий хотя бы один метод (**abstract**). **Виртуальный** метод имеет реализацию и м.б. переопределен в производном **классе**. **Абстрактный** метод не имеет реализацию, только описание метода, который д.б. реализован в производных **классах**.

19) Какие компоненты класса могут быть виртуальными?

методы, свойства, индексаторы

20) Что такое интерфейс?

**Интерфейсы**, как и классы, определяют набор свойств, методов и событий. Но, в отличие от классов, они не содержат их реализации. Интерфейсы реализуются классами и определяются как самостоятельные сущности.

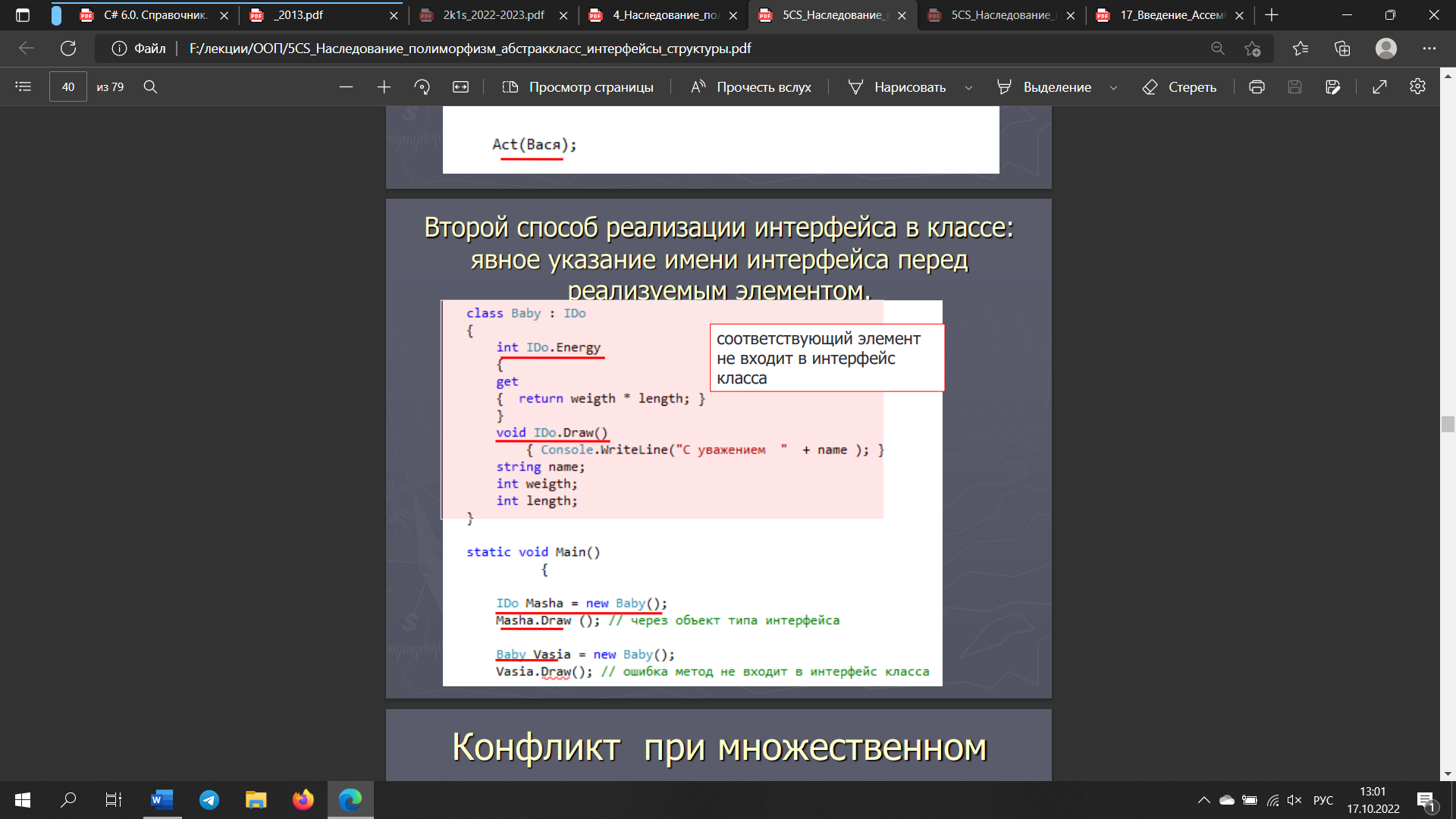
Interface IDo  
 void Go()  
 int Jumn (int a)  
 void Sleep()  
 int Energy { get; }

21) Что может содержать интерфейс?

1. Методы
2. Свойства
3. Индексаторы
4. События

22) Как работать с объектом через унаследованный интерфейс?

23) Приведите пример явной реализации интерфейса.



24) Почему нельзя указать модификатор видимости для методов интерфейса?

Потому что все они должны иметь модификатор ***public***, который и установлен по умолчанию.

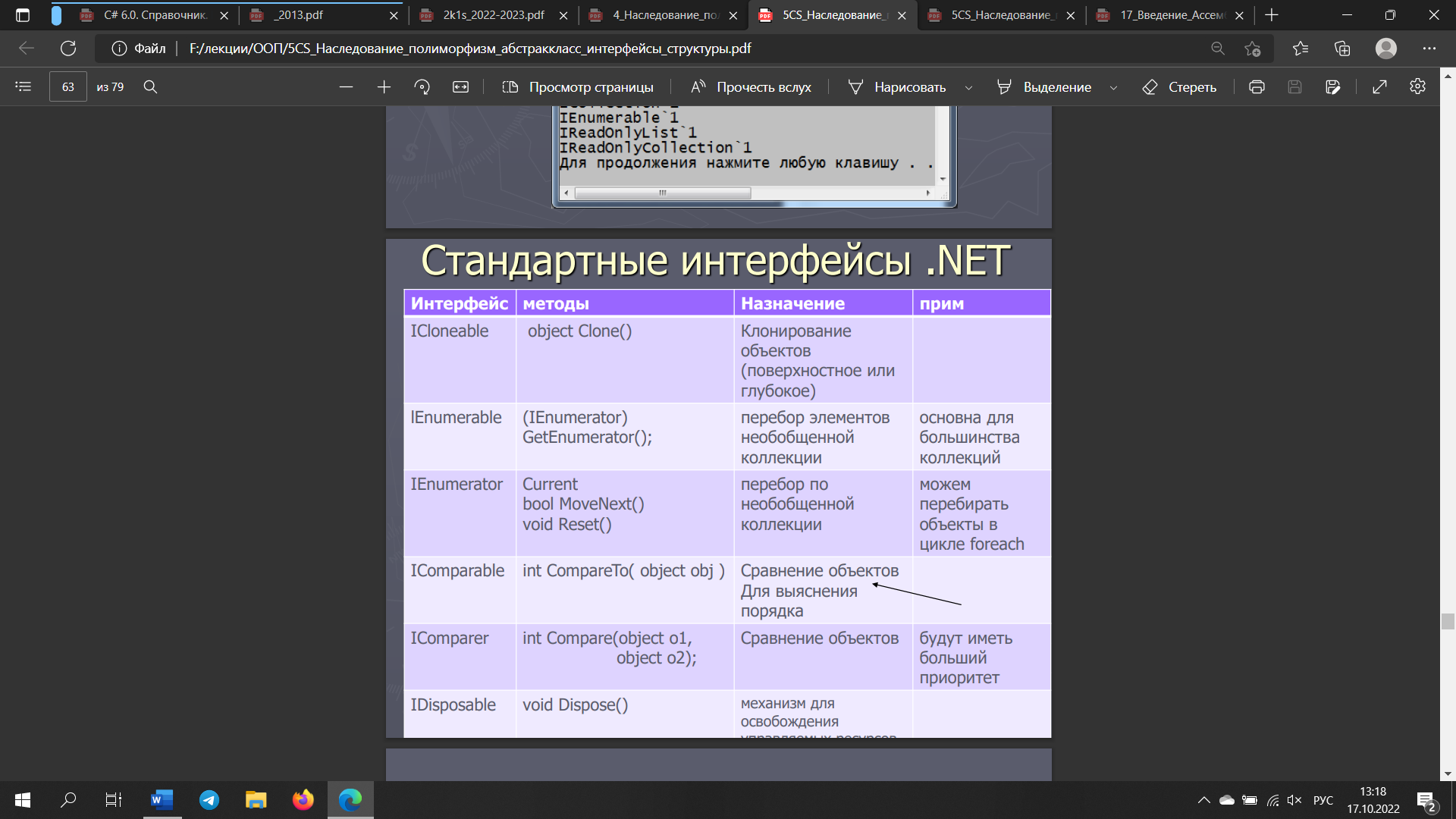
25) Можно ли наследовать от нескольких интерфейсов?

Да, через запятую.

26) Назовите отличия между интерфейсом и абстрактным классом.



27) Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, IEnumerable?



1. В какой строке приведенного ниже фрагмента листинга не содержится ошибки?

class A {

public virtual abstract void m() { } //1

public virtual void g() { } //2

publicvirtual new new void f() { } //3

public static virtual void h() { } //4

}

(2)

29) Что будет выведено на консоль в результате выполнения следующего фрагмента?

class A {

public int x = 1;

}

class B : A {

public new int x = 2;

public void m(int a, int b)

{

x = a;

base.x = b;

Console.Write(x + " " + base.x);

}

}

class Test {

static void Main(string[] args) {

A a = new A();

B b = new B();

b.m(3, 4);

}

}

3 4

30)

class A {

public

class B : A {

public

override void mA() { Console.WriteLine("B "); }

} public virtual void mA() {

Console.WriteLine("A ");

}

} class Program {

static void Main(string[] args) {

A a = new A();

A.B b = new A.B();

a.mA();

b.mA();

}

}

А

В

31)

1)M4 - только интерфейс

32) Выберите верное присваивание для объектов, определенных в листинге.

1) b = a;

2) a = b;

3) c = a;

4) d = a;

5) с = b;

33) Что будет выведено на консоль в результате выполнения следующего фрагмента, если раскомментировать строчку 1?

public abstract class A {

public virtual void method() {

Console.Write("A ");

}

}

public class B : A {

public override void method() {

// base.method(); // 1

// this.method(); // 2

Console.Write("B ");

}

}

class Program2 {

static void Main(string[] args) {

A my = new B();

my.method();

}

}

А В

34)

publica bstract class A {

public virtual string m() { return "A"; } // 1

}

public class B : A {

public override new string m() { return "B"; } // 2

}

public class C : B {

public string m() { return "C"; } // 3

}

class Program {

static void Main(string[] args) {

A ac = new C(); // 4

Console.WriteLine(ac.m());

}

}

35) Почему приведенный ниже фрагмент листинга содержит ошибку?

abstract class Student // 1

{

public

int Age {

get;

set;

} // 2

public

string Name {

get;

set;

} // 3

} static void Main(string[] args) {

Student Olga = new Student(); // 4

}

(4), т.к нельзя создавать объекты абстр. класса

36) В какой строке может быть ошибка компиляции?

(4)

class A { }

class B : A { }

class C : A { } // 1

class Program4 {

static void Main() {

A one = new B(); // 2

A two = new C(); // 3

one = two; // 4

}

}

37)

interface Interface1 {

void f();

void g();

}

class A {

public void f() { System.Console.WriteLine("F"); }

public void g() { System.Console.WriteLine("G"); }

} class B : A, Interface1 {

new public void g() { System.Console.WriteLine("new G"); }

} class Program5 {

static void Main(string[] args) {

// Interface1 obj = new B();

// obj.g();

B obj = new B();

obj.g();

}

}