Вопросы для теста по программированию

Какой принцип ООП должен стоять на месте многоточия? … - свойство объектов с одинаковой спецификацией иметь различную реализацию.

Какой принцип ООП должен стоять на месте многоточия? … - возможность создавать новые классы на основе уже существующих.

Какой принцип ООП должен стоять на месте многоточия? … - свойство объектов скрывать реализацию своего поведения от внешнего окружения, предоставляя интерфейс доступа к ним.

Какой принцип ООП должен стоять на месте многоточия? … - выделение наиболее значимых характеристик объекта для его реализации.

Что выведет в консоль следующая программа?

```
class Auto  // легковой автомобиль
{
   public int Seats { get; set; } // количество сидений
   public Auto(int seats)
   {
      Seats = seats;
   }
}
class Truck : Auto      // грузовой автомобиль
{
   public decimal Capacity { get; set; } // грузоподъемность
   public Truck(int seats, decimal capacity)
   {
      Seats = seats;
      Capacity = capacity;
   }
}
class Program
{
   static void Main(string[] args)
   {
      Truck truck = new Truck(2, 1.1m);
      Console.WriteLine($"Грузовик с грузоподъемностью {truck.Capacity} тонн");
      Console.ReadKey();
   }
}
```

Что выведет в консоль следующая программа?

```
class Person
   {
      private string name;
      Person(string name)
      {
```

```
        this.name = name;
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        Person p1 = new Person("Александр");
        p1.name = "Пётр";
        Console.WriteLine(p1.name);
    }
}
```

Может ли класс иметь несколько методов с одинаковым названием, но разными параметрами?

Разрешено ли в C# множественное наследование?

Даны 2 класса - Person и Man. Выполнится ли программа?

```
class Person
{
    public void talk(){}
}
class Man: Person
{
    override void talk() {}
}
```

Даны 2 класса - Person и Man. Выполнится ли программа?

```
class Person
{
    Virtual public void talk(){}
}
class Man: Person
{
    override void talk() {}
}
```

Даны 2 класса - Person и Man. Выполнится ли программа?

```
class Person
{       public void talk(){}    }
class Man: Person     {       void talk() {}    }
```

С помощью чего реализуется принцип инкапсуляции в C#?

Данная программа не компилируется. В чем причина?

```
class Base
{
   public virtual void Display()
   {
      Console.WriteLine("This is Base");
   }
}
abstract class Derived : Base
{
   public override void Display()
   {
      Console.WriteLine("This is Derived");
   }
}
class Program
{
   static void Main(string[] args)
   {
      Base _base = new Derived();
      _base.Display();
} }
```

Что будет выведено в консоль в результате работы программы?

```
class Person
{
   private int age = 15;
   public int Age
   {
      get { return age; }
      set { }
   }
}
class Program
{
   static void Main(string[] args)
   {
      Person tom = new Person();
      tom.Age = 25;
      Console.WriteLine(tom.Age);
   }
}
```

Что нельзя сделать в классе-наследнике?

Есть два класса. Как можно преобразовать объект Person к типу Employee?

```
class Person
{
   public string Name { get; set; }
}
class Employee : Person { }
```

В чем отличие структуры (struct) от класса (class)?

Каких модификаторов нет в C#?

Выберите все существующие принципы ООП из данных.

Выберите код, который будет работать.

1. public class Person {}
   public class Man: Person {}
2. public class Person {}
   public class Man: Person {}
   public class Child: Person, Man{}
3. public class Person {}
   public class Man: Person {}
   public class Child: Man{}
4. private class Person {}
   public class Man: Person {}
   public class Child: Man{}

Выберите верные утверждения, описывающие инкапсуляцию.

Выберите верные утверждения, описывающие наследование.

Выберите верные утверждения, описывающие полиморфизм.

Какой код написан вызовет ошибку на стадии компиляции?

1. private class Person { virtual void talk(){}}
    public class Man: Person {override void talk(){}}
2. public class Person { void talk(){}}
   public class Man: Person {override void talk(){}}

3. public class Person { void talk(){}}
   public class Man: Person {override void scream(){}}
4. public class Person { void talk(){}}
   public class Man: Person { void scream(){}}

Можно ли переопределять все свойства и методы базового класса?

Что должно стоять вместо пропуска?
_____ – метод класса, который вызывается при создании нового объекта данного класса.

a) Конструктор
b) Деструктор
c) Индексатор
d) Инициализатор

Что должно стоять вместо пропуска?
_____ объектов позволяет присваивать значения всем доступным полям и свойствам объекта во время создания

Какой смысл имеет операция new?

Что такое класс?

Инкапсуляция – это……

Выберите режимы инкапсуляции…..

Указатель this-

a) это указатель на объект, для которого был вызван нестатический метод.
b) Это указатель на класс
c) обеспечивает доступ к текущему экземпляру класса.
d) Это указатель на статический метод

Заимствование методов и свойств класса родителя называется….

Выберите правильное объявление класса наследника

a) class [имя_класса_наследника] : [имя_базового_класса]
b) class [имя_базового_класса] : [имя_класса_наследника]
c) class [имя_класса_наследника] = [имя_производного_класса]

d) class [имя_класса_наследника] as [имя_базового_класса]
e) class [имя_базового_класса] => [имя_класса_наследника]

Производный класс – это

a) Класс который порожден от другого
b) Класс который порождает другой
c) Класс на котором базируются другие классы

Базовый класс –

a) это класс, находящийся выше по иерархии наследования классов
b) это класс который порожден от другого
c) это контейнер, который хранит информацию в определенном виде

Отметьте верные утверждения

a) Класс может быть порождён из одного или нескольких классов
b) Класс может быть порождён только из одного класса
c) Наследуемый класс называется базовым
d) Наследуемый класс называется производным
e) в производном классе могут быть объявлены собственные дополнительные данные и функции
f) в производном классе могут быть объявлены только дополнительные функции

Отметьте верное определение класса

a) Абстрактный тип данных, описывающий характеристики и поведение объекта
b) это контейнер, который хранит информацию в определенном виде
c) это линейная структура данных, которая хранит элементы последовательно

Дан фрагмент кода:

```
class Car
{
  private double mileage;
  private double fuel;
  public Car(double mileage, double fuel)
  {
        this.mileage = mileage;
        this.fuel = fuel;
  }
}
```

как вызвать конструктор класса Car? (может быть несколько вариантов)

Выберите верное утверждение

a) Деструктор, это указатель на объект, для которого был вызван нестатический метод. Общая форма деструктора имеет вид: ~имя_класса(){// код деструктора}
b) Деструктор, это метод который вызывается для при создании нового объекта данного класса. Общая форма деструктора имеет вид: ~имя_класса(){// код деструктора}
c) Деструктор, это метод который вызывается для гарантированного освобождения памяти под объект. Общая форма деструктора имеет вид:

имя_класса(){// код деструктора}

d) Деструктор, это метод который вызывается для гарантированного освобождения памяти под объект. Общая форма деструктора имеет вид:

~имя_класса(){// код деструктора}

e) Деструктор, это метод который вызывается для гарантированного освобождения памяти под объект. Общая форма деструктора имеет вид:

~имя_класса([параметры]){// код деструктора}

Какие члены класса могут быть объявлены в теле класса:

a) Поля
b) Свойства
c) Методы
d) Конструкторы
e) Экземпляры класса

Выберите верные утверждения:

a) класс – является ссылочным типом (*reference type*), а структуры – значимым типом (*value type*)
b) структура – является ссылочным типом (*reference type*), а класс – значимым типом (*value type*)
c) В структурах не допускается определять деструкторы.
d) Для структуры нельзя определить конструктор, используемый по умолчанию
e) Структуры так же как и классы поддерживают наследование.
f) Структуры не поддерживают инкапсуляцию

Выберите верные утверждения:

a) Модификатор abstract в объявлении класса указывает, что класс предназначен только для использования в качестве базового класса для других классов.
b) Члены, помеченные как абстрактные или включенные в абстрактный класс, должны быть реализованы с помощью классов, производных от абстрактных классов.
c) Абстрактный класс не может содержать методов
d) Модификатор abstract может использоваться с классами, методами, свойствами, индексаторами и событиями.
e) Экземпляр абстрактного класса можно создать только с помощью конструктора без параметров.

Для каких элементов класса можно применять полиморфизм?

a) методов;
b) свойств;
c) индексаторов;
d) событий.

Отметьте наборы операторов, которые можно перегрузить явным образом

a) +, -, > , <
b) ==, != , ++
c) +=, *=, +, true
d) <=, >=, ++

Какие операторы перегружаются только парами

a) < и >
b) == и !=
c) + и –
d) true и false
e) * и /

Для понятия «полиморфизм» справедливы следующие утверждения

a) полиморфизм определяется для семейства классов, связанных отношением наследования
b) полиморфизм определяется для семейства классов, связанных отношением вложенности
c) полиморфизм означает, что в классе заданы перегруженные методы
d) реализация полиморфизма построена на динамическом связывании
e) реализация полиморфизма предполагает статический контроль типов

Выберите верные утверждения

a) Связывание — это процесс разрешения, или соотнесения идентификаторов типов (членов типа, операций) с логическими частями программы, представляющими соответствующие типы.
b) Статическое связывание позволяет отложить этот процесс сопоставления имени с конкретной функцией до момента выполнения.
c) Динамическое связывание позволяет отложить этот процесс сопоставления имени с конкретной функцией до момента выполнения.
d) Статическое связывание - сопоставление имени с конкретной функцией при компиляции выражения.
e) Динамическое связывание - сопоставление имени с конкретной функцией при компиляции выражения.

Можно ли переопределить метод PrintPerson() в классе наследнике? Если, да то какой синтаксис подходит для переопределения?

```
class Person
{
   public string FirstName { get; set; }
   public string SecondName { get; set; }
   public virtual void PrintPerson()
   {
      Console.WriteLine(FirstName + ‘ ’ + SecondName);
   }
```

```
}
```

Отметьте верные утверждения

a) Диаграммы классов представляют собой экземпляр диаграммы объектов в конкретный момент.

b) Диаграмма классов описывает атрибуты и операции класса, а также ограничения, налагаемые на систему

c) Диаграмма классов показывает набор классов, интерфейсов, ассоциаций, взаимодействий и ограничений.

d) Диаграммы объектов представляют собой экземпляр диаграммы классов в конкретный момент.

Что из перечисленного относится к коллекциям?

a) ArrayList
b) List<T>
c) Stack<T>
d) String
e) ArrayStack
f) Array

Отметьте верные утверждения -

a) В необобщенной коллекции хранятся элементы разного типа.
b) В обобщенной коллекции хранятся элементы одного типа.
c) Элементы обобщенной коллекции нельзя сортировать
d) В списках можно добавлять элементы только одного типа
e) В коллекциях можно добавлять элементы только в конец

Объявлен список:

```
var myList = new List<string>();
```

Что из перечисленного выдаст ошибку

a) myList.Add("sometext");
b) myList.Add("c");
c) myList.Add("1234");
d) myList.Add(1);

Что выведет код?

```
var myList = new List<string> {"some", "text", "for", "you"};
myList.Remove("Some");
foreach (var li in myList)
{
   Console.Write(li + " ");
}
```

Дана часть кода, какие цифры выведутся на экран после запуска?

```
List<int> numbers = new List<int>() { 1, 4};
numbers.Add(5);
```

```
numbers.AddRange(new int[] { 2, 3});
numbers.Insert(0, 6);
numbers.RemoveAt(1);
foreach (int i in numbers)
    {   Console.WriteLine(i); }
```

Какие библиотеки нужно подключить при работе с коллекциями?

a) Collections.Generic
b) System.Collections
c) System.Xml.Linq
d) System.Collections.Generic

Заполните пропуски

Класс __[1]__представляет __[2]__ список, в котором каждый элемент хранит ссылку одновременно на следующий и на предыдущий элемент.

a) LinkedList<T>, двухсвязный
b) LinkedList<T>, несвязный
c) SortedList<T>, двухсвязный
d) SortedList<T>, отсортированный

Что выведется при запуске фрагмента кода

```
LinkedList<int> numbers = new LinkedList<int>();
numbers.AddLast(1);
numbers.AddFirst(2);
numbers.AddAfter(3);
foreach (int i in numbers)
{     Console.WriteLine(i);   }
```

Что из перечисленного относится к коллекциям?

a) Список
b) Стек
c) Очередь
d) Строка
e) Двухсвязный список

Какие интерфейсы НЕ реализуются в классе ArrayList?

a) ICollection
b) IList
c) IEnumer
d) ICloneable

Дан фрагмент кода

```
ArrayList al = new ArrayList();
al.Add(10);
al.Add(20);
```

```
al.Add(30);
al.Reverse();
al.Reverse(2, 3);
```

Чему равно значение al?

Заполните пропуски

Класс [1] представляет обычную очередь, работающую по алгоритму [2] ("первый вошел - первый вышел").

a) Queue<T> , FIFO
b) List<T>, FIFO
c) Dictionary<T>, FOFI
d) Queue<T> , input-output

Дан фрагмент кода

```
Queue<int> numbers = new Queue<int>();
numbers.Enqueue(3);
numbers.Enqueue(5);
numbers.Enqueue(8);
int queueElement = numbers.Dequeue();
int pp = numbers.Peek();
Console.WriteLine(queueElement);
```

Что выведет код?

Какие методы манипулируют данными в стеке?

Отметьте верные утверждения

a) Коллекции предназначены для стандартизированной обработки групп объектов в программе.
b) Динамические структуры данных такие как: очереди, линейные списки. Не относятся к коллекциям.
c) в коллекциях обеспечиваются распространенные операции обработки массивов данных, например, сортировка.
d) Все коллекции реализованы на базе четко определенных интерфейсов.

Как удалить все элементы из массива типа ArrayList?

a) Метод Clear()
b) Метод Clean()
c) Метод Remove()
d) Метод Delete()

Дан словарь

```
Dictionary<string, string> countries = new Dictionary<string, string>
```

Какое из объявлений его элементов будет не верным

a) Dictionary<string, string> countries = new Dictionary<string, string>
```
{
    {"Франция", "Париж"},
    {"Германия", "Берлин"},
    {"Великобритания", "Лондон"}
};
```
b) Dictionary<string, string> countries = new Dictionary<string, string>
```
{
    {[Франция], "Париж"},
    {[Германия], "Берлин"},
    {[Великобритания], "Лондон"}
};
```
c) Dictionary<string, string> countries = new Dictionary<string, string>
```
{
    {"Франция"}= "Париж",
    {"Германия"}= "Берлин",
    {"Великобритания"}= "Лондон"
};
```
d) Dictionary<string, string> countries = new Dictionary<string, string>
```
{
    ["Франция"]= "Париж",
    ["Германия"]= "Берлин",
    ["Великобритания"]= "Лондон"
};
```

Что выведет фрагмент кода ?

```
LinkedList<int> numbers = new LinkedList<int>();
numbers.AddLast(1);
numbers.AddFirst(2);
numbers.AddAfter(numbers.Last, 3);
foreach (int i in numbers)
 {   Console.WriteLine(i);  }
```

Какие существуют методы двухсвязного списка?

a) AddAfter(LinkedListNode<T> node, LinkedListNode<T> newNode)
b) Add(T value)
c) AddAfter(LinkedListNode<T> node, T value)
d) RemoveFirst()
e) Remove()

Какие типы входят в пространство имен System.Collections.Specialized

a) ListDictionary
b) StringCollection
c) DictionaryString
d) StringArray

Какие значения не может хранить ArrayList?

Что делает метод ContainsValue() класса Dictionary<TKey, TValue>

a) Возвращает логическое значение true, если вызывающий словарь содержит значение value; в противном случае — логическое значение false

b) Возвращает логическое значение true, если вызывающий словарь содержит объект key в качестве ключа; а иначе — логическое значение false

c) Получает коллекцию ключей

d) Получает коллекцию значений

Отметьте верные утверждения:

a) HashSet<T> содержит неповторяющюся коллекцию элементов.

b) HashSet<T> содержит элементы неупорядоченно (без сортировки).

c) HashSet<T> позволяет быстро определить, есть такой элемент или нет (быстро потому что, использует индекс, который вычисляется из хэш-кода элемента).

Что выведет код?

```
Hashtable Assosiations = new Hashtable();
Assosiations.Add("Соколовский", 50.19);
Assosiations.Add("Дормидотов", 98.85);
Assosiations.Add("Загибинко", 45,96);
Assosiations.Add("Сальников", 82.66);
try
{
        Assosiations.Add("Дормидонтов", 93.66);
        Console.WriteLine("Запись добавлена");
}
catch
{
        Console.WriteLine("Ключ уже содержится в таблице");
}
```

Какой класс является основным при работе с регулярными выражениями?

Что означает символ ^ в синтаксисе регулярного выражения?

Что означает символ * в синтаксисе регулярного выражения?

Какой формат нельзя использовать для сериализации?

Что нужно сделать, чтобы объект класса можно было сериализовать?

Что нужно сделать, чтобы какое-то поле класса не сериализовалось?

Что обязательно должно быть у объекта, который нужно десериализовать?

Как расшифровывается LINQ?

1. Language-Interface-Query
2. Language-Integrated-Query
3. Language-Incapsulation-Quality
4. Language-Inference-Quard

Что нужно вставить на место пропуска в следующем запросе LINQ?
From t ____ order by t select t?

1. Условие выбора (например where t.id == 1)
2. Условие группировки (например group by t.age)
3. Цикл по объектам (например foreach(string t in teams))
4. Указание, откуда необходимо выбрать данные (например in teams)

Допустим у нас есть объект teams.
string[] teams = {"Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити", "ПСЖ", «Барселона"};
Что выведет следующая конструкция LINQ? (если мы распечатаем объект selectedTeams)

```
var selectedTeams = from t in teams
                    where t.ToUpper().StartsWith("Б")
                    orderby t select t;
```

С какими объектами не позволяет работать LINQ?

Допустим, что у нас есть список пользователей users. У объекта класса User есть имя name и возраст age. Пользователи: (Tom, 33), (Bob, 30), (Tom, 27), (Sam, 24) , (Sam, 20) , (Bob, 23)
Что будет выведено, если мы распечатаем переменную result после выполнения запроса LINQ?

```
var result = from user in users
             orderby user.Name, user.Age
             select user;
```

Какой запрос выберет из массива целых чисел numbers четные элементы, которые больше 10?

1. From I in numbers where i%2==0 select i;
2. From I in numbers where i%2==0 and i>10 select i;
3. From I in numbers where i%2==0 && i>10 select i;
4. From I in numbers where i%2!=0 && i>10 select i;

Допустим, что у нас есть шаблон регулярного выражения «World». В каких строках будут найдены совпадения?

Допустим, что у нас есть шаблон регулярного выражения «World$». В каких строках будут найдены совпадения?

Допустим, что у нас есть шаблон регулярного выражения «^World». В каких строках будут найдены совпадения?

Какие есть варианты задать регулярное выражение для нахождения телефонного номера в формате +7-111-111-1111, где 1 - любые цифры.

Допустим, что мы хотим сериализовать/десериализовать весь объект класса Person. Что не так в этом коде?

```
public class Person {
                public string name;
                public int age;
                public Person(string name, int age) {
                        this.name = name;
                        this.age = age;
}       }
```

Какие форматы можно использовать для сериализации объекта?

1. текстовый
2. HTML
3. XML
4. тернарный
5. бинарный

Выберите идентичные запросы

1

```
foreach(string s in teams){
   if (s.ToUpper().StartsWith("Б"))
      selectedTeams.Add(s);
}
selectedTeams.Sort();
```

2.

```
var selectedTeams = from t in teams
                where t.ToUpper().StartsWith(«Б")
                orderby t select t;
```

3.

```
foreach(string s in teams){
   if (s.ToUpper())
      selectedTeams.Add(s);
}
```

4.

```
var selectedTeams = from t in teams
                where t.ToUpper().StartsWith(«Б")
                select t;
```

Выберите идентичные запросы

1

```
foreach(string s in teams){
   if (s.ToUpper())
      selectedTeams.Add(s);
}
selectedTeams.Sort();
```

2.

```
foreach(string s in teams){
   if (s.ToUpper())
      selectedTeams.Add(s);
}
selectedTeams.Sort();
```

3.

```
var selectedTeams = from t in teams
                    where t.ToUpper().StartsWith(«Б")
                    orderby t select t;
```

4.

```
var selectedTeams = teams.Where(t=>t.ToUpper().StartsWith("Б")).OrderBy(t => t);
```

Что может определять интерфейс?

Какие методы и свойства должны иметь реализацию по умолчанию?

Выберите верные утверждения.

A. Слово «интерфейс» имеет разный смысл в зависимости от контекста
B. Множественное наследование интерфейсов дает те же возможности, что и множественное наследование классов
C. При наследовании интерфейса ICloneable необходимо реализовать метод MemberwiseClone
D. При наследовании двух интерфейсов имена их методов должны быть различными
E. Несколько интерфейсов могут быть наследниками одного и того же интерфейса

Выберите верные утверждения:

A. Один класс может наследовать несколько интерфейсов
B. Один интерфейс может наследоваться несколькими классами
C. Из-за коллизии имен дублируемое наследование интерфейсов запрещено
D. Интерфейс может быть наследником нескольких интерфейсов

Выберите верные утверждения. Класс с атрибутом Serializable

A. Должен быть наследником интерфейса ISerializable
B. При вызове форматером метода Serialize выполняет глубокую сериализацию, если класс не является наследником интерфейса ISerializable
C. Допускает два формата сериализации данных
D. Облегчает организацию обмена данными с удаленным приложением
E. Позволяет сохранять данные в текстовом формате

Выберите верные утверждения. При наследовании интерфейсов

A. Класс наследник должен реализовать хотя бы один из его методов
B. Класс наследник может выполнить переименование методов интерфейса
C. Класс наследник может выполнить склейку методов с одинаковой сигнатурой
D. Объекту интерфейсного типа доступны закрытые методы интерфейса, реализованные в классе
E. Объекты интерфейсного типа создаются стандартным путем с помощью конструктора

Выберите верные утверждения. Интерфейс отличается от абстрактного класса тем, что:

A. У абстрактного класса все методы абстрактны, а у интерфейса некоторые методы могут иметь реализацию
B. У интерфейса все методы абстрактны, а у абстрактного класса некоторые методы могут иметь реализацию
C. Потомок может иметь в качестве родителя несколько интерфейсов, но только один абстрактный класс

Выберите верные утверждения. Какой тип наследования поддерживает C#?

A. множественное наследование (от класса и нескольких интерфейсов)
B. множественное наследование от нескольких интерфейсов.
C. множественное наследование от нескольких классов.
D. Не поддерживает никакой тип множественного наследования

Пусть задано описание интерфейса и класса:

```
interface IP{string M(string s);
string M1(int s);}
public class C1:IP{string IP.M(string s){return (s+s);}
string IP.M1(int x){return x.ToString();}
public int M (int s) { return (s++);}}.
```

Какие из объявлений в клиентском классе выполнимы?

A. C1 it1 = new C1(); it1.M(7777)
B. C1 it2 = new C1(); string s ="ss"; s =it2.IP.M(s)
C. C1 it3 = new C1(); string s ="ss"; s =((IP)it3).M(s)
D. IP it4 = new IP(); string s= "ss"; s = it4.M(s)
E. IP it5 = (IP) new C1(); string s= "ss"; s = it5.M(s)

Пусть задано описание интерфейсов:

```
interface IN{string M(string s);}
interface IP{string M(string s);  string M1(int s);}
interface IQ{int M(int s);}.
```

Какие из объявлений классов содержат ошибки

A. public class C1:IP{string IP.M(string s){return (s+s);} string IP.M1(int x){return x.ToString();}public int M (int s) { return s++;}}
B. public class C1:IP,IN{string IP.M(string s){return (s+s);} string IP.M1(int x){return x.ToString();}}
C. public class C1:IP,IN{public string M(string s){return (s+s);} public string M1(int x){return x.ToString();}}

D. public class C1:IP,IN,IQ{public string M(string s){return (s+s);} public string M1(int x){return x.ToString();}}

Выберете верное продолжение утверждения. Интерфейс ISerializable ...

A. Автоматически реализует глубокую сериализацию
B. Позволяет управлять процессом сериализации
C. Имеет два метода, которые должен реализовать класс, наследующий интерфейс
D. Конфликтует с атрибутом класса Serializable

Могут ли методы и свойства интерфейсов не иметь реализации?

A. Могут
B. Не могут
C. Неверная постановка вопроса

Если методы и свойства интерфейса не имеют модификатора доступа, то какой доступ они имеют по умолчанию?

Какой уровень доступа по умолчанию имеют интерфейсы?

Что должен делать класс, если к нему применен интерфейс?

A. Реализовывать все методы и свойства интерфейса
B. Реализовывать только не определенные по умолчанию
C. Реализовывать только методы не определённые по умолчанию
D. Реализовывать только свойства не определённые по умолчанию

Что неправильно в определении следующих интерфейсов:

```
public abstract interface IAction
{    void Move(); }
internal interface IRunAction : IAction
{    void Run();  }
```

Дана следующая программа:

```
using System;
namespace HelloApp
{
   class Program
   {
      static void Main(string[] args)
      {
         IAction action = new RunAction();
         action.Move();
       }
   }
   class RunAction : IRunAction
```

```
    {
        internal void Move()
        {
            Console.WriteLine("I am running");
        }
    }
    interface IAction
    {
        void Move();
    }
    interface IRunAction : IAction
    {
        new void Move();
    }
}
```

Почему данная программа не скомпилируется?

Класс Tester реализует интерфейсы IFoo и IBar:

```
interface IFoo
{
    void Execute();
}
interface IBar
{
    void Execute();
}
class Tester : IFoo, IBar
{
    public void Execute()
    {
        Console.WriteLine("Tester Executes");
    }
}
```

Метод Execute() какого именно интерфейса реализует класс Tester?

Класс Tester реализует интерфейсы IFoo и IBar:

```
interface IFoo
{
    void Execute();
}
interface IBar
{
    void Execute();
}
```

```
class Tester : IFoo, IBar
{
   void IFoo.Execute()
   {
      Console.WriteLine("IFoo Executes");
   }
   void IBar.Execute()
   {
      Console.WriteLine("IBar Executes");
   }
}
```

Реализация метода Execute() из какого интерфейса будет вызвана в следующем коде:

```
  Tester t = new Tester();  t.Execute();
```

Что можно сделать в ситуации когда базовый класс реализовал интерфейс, но в классе-наследнике необходимо изменить реализацию этого интерфейса?

A. Переопределить виртуальные/абстрактные методы
B. Сокрыть метод в производном классе
C. Повторно реализовать интерфейс в классе-наследнике

Можно ли применять к интерфейсам идентификатор abstract?

A. Можно
B. Можно, только если от интерфейса наследуется другой интерфейс
C. Можно, но не рекомендуется
D. Нельзя

Могут ли методы интерфейсов использовать ключевое слово new?

A. Могут, для сокрытия методов из базового интерфейса
B. Могут, для сокрытия методов класса
C. Могут, для изменения private методов из базового класса
D. Не могут

Какой интерфейс применяется для сортировки наборов сложных объектов?

A. ICloneable
B. IComparable
C. IComparator
D. ICollection

Что реализует метод MemberwiseClone()?

A. Поверхностное копирование
B. Глубокое копирование
C. Этот метод не реализует копирование

Какую проблему можно решить с помощью реализации интерфейса ICloneable?

A. Проблема поддержки множественного наследования
B. Проблема кодировки

C. Проблема с использованием ссылочных типов
D. Проблема с доступом к скрытым свойствам

Есть делегат: delegate int Operation (int val);
Какие из следующих методов соответствуют данному делегату:

```
 static void Method1(int x)
{
   Console.WriteLine(x * x);
}
static int Method2(ref int x)
{
   return x * x;
}
static int Method3(int x, int y)
{
   return x * y;
}
static int Method4(int x, int y = 7)
{
   return x * y;
}
static int Method5(out int x)
{
   x = 7;
   return x * x;
}
```

Что будет выведено на консоль в результате выполнения следующей программы:

```
class Program
{
   delegate int Operation(int x, int y);
   static void Main(string[] args)
   {
      Operation del = Add;
      del += Multiply;
      int result = del(6, 5);
      Console.WriteLine(result);
   }
   private static int Add(int x, int y) { return x + y; }
   private static int Multiply(int x, int y) { return x * y; }
}
```

Какой будет консольный вывод при выполнении следующей программы:

```
class Program
{
   delegate void Message();
   static void Main(string[] args)
   {
      Message mes1 = new Message(Hello);
      mes1 += HowAreYou;
      mes1 += Hello;
      mes1 += Hello;
      mes1 -= Hello;
      mes1();
   }
   private static void Hello() { Console.WriteLine("Hello"); }
   private static void HowAreYou() { Console.WriteLine("How are you?"); }
}
```

Что будет выведено на консоль в результате выполнения следующей программы:

```
class Program
{
   delegate void Message();
   static void Main(string[] args)
   {
      Message mes = new Message(Hello);
      mes += Hi;
      mes -= Hello;
      mes -= Hi;
      mes();
      Console.Read();
   }
   private static void Hello() { Console.WriteLine("Hello"); }
   private static void Hi() { Console.WriteLine("Hi"); }
}
```

Все аргументы события принадлежат -

A. Классу EventArgs
B. Классу EventArgs, за исключением аргумента, задающего объект, зажигающий событие
C. Потомку класса EventArgs
D. Потомку класса EventArgs за исключением аргумента, задающего объект, зажигающий событие

Сообщение о событии получает:

A. Только один объект одного класса, имеющего обработчик события, присоединенный к объекту, задающему событие

B. Несколько объектов одного класса, имеющего обработчик события, присоединенный к объекту, задающему событие
C. Все объекты одного класса, имеющего обработчик события, присоединенный к объекту, задающему событие
D. Все объекты всех классов, имеющих обработчики события, присоединенные к событию

Зажечь событие – это значит:

A. Послать сообщение о событии всем объектам – получателям сообщения
B. Вызвать специальную процедуру, обрабатывающую событие
C. Закончить работу, передав управление операционной системе

Выберите верные утверждения. Обработчик события:

A. Всегда принадлежит классу, вызывающему событие
B. Никогда не принадлежит классу, вызывающему событие
C. Может принадлежать классу, вызывающему событие
D. Принадлежит только одному классу, слушающему событие
E. Может принадлежать многим классам, слушающим события

Выберите верные утверждения. Объекты одного класса:

A. Имеют одинаковый набор событий
B. В процессе вычислений зажигают одинаковый набор событий
C. В процессе вычислений зажигают набор событий, характерный для объекта
D. В разных сеансах работы зажигают одинаковый набор событий
E. В разных сеансах работы зажигают разные события

Выберите верные утверждения. Процедура, в которой вызывается событие -

A. Передает обработчику событий входные аргументы и продолжает свою работу
B. Передает обработчику событий входные аргументы и ждет окончания работы первого обработчика, связанного с событием
C. Передает обработчику событий входные аргументы и ждет окончания работы всего списка обработчиков, связанного с событием
D. Может использовать в своей работе выходные аргументы события, значения которых сформированы обработчиками события

Отметьте истинные высказывания -

A. Все события имеют одинаковую сигнатуру из двух аргументов с одними и теми же типами
B. Все события имеют сигнатуру из двух аргументов, но с отличающимися типами
C. Все события, не имеющие собственных аргументов, передаваемых обработчику, соответствуют стандартному встроенному делегату EventHandler
D. Для класса с большим числом событий целесообразно применять динамическое связывание
E. Для связывания событий с обработчиком можно применять только операцию +=

Отметьте истинные высказывания -

A. Объект, обрабатывающий событие, может отсоединить других обработчиков события и полностью определить реакцию на событие
B. Объект может определять, обрабатывать ли событие и какой из обработчиков присоединять к событию

C. Метод GetInvocationList используется при обработке событий
D. Все события имеют одну и ту же сигнатуру

Отметьте истинные высказывания -

A. События – это специализированный вариант делегатов
B. К каждому событию должен быть присоединен хотя бы один обработчик события
C. Методы Combine и Remove, унаследованные от класса Delegate не применимы к событиям
D. Допустимо динамическое связывание событий с обработчиками событий

Выберите верные утверждения -

A. Делегат, задающий тип события, может иметь произвольную сигнатуру
B. Класс, обрабатывающий события, может отсоединить обработчики событий других классов, чтобы они не мешали его работе
C. Обработчику события передаются входные аргументы, характеризующие событие, а он может передать объекту, создавшему событие, выходные аргументы

Выберите верные утверждения -

A. Класс-отправитель сообщения обязан знать всех своих получателей
B. Сообщение о событии следует посылать только тогда, когда гарантируется, что есть хотя бы один получатель сообщения
C. Класс получатель должен иметь обработчик события – процедуру, согласованную по сигнатуре с функциональным типом делегата, задающего событие

Выберите верные утверждения -

A. Если обработчику события не передаются дополнительные аргументы, то можно пользоваться стандартным делегатом с именем EventHandler
B. При зажигании события выполнение метода прерывается, пока не закончат работу все обработчики события
C. Стандартные интерфейсные объекты обладают набором предопределенных событий
D. Объекты классов, создаваемых программистом, наследуют предопределенные события класса object

Выберите верные утверждения. Класс EventHandler

A. Является классом, которому принадлежат все события
B. Является абстрактным классом и не может иметь экземпляров
C. Является родительским классом для событий, не имеющих собственных аргументов
D. Является потомком класса Delegate
E. Задает сигнатуру события

Для чего используются анонимные методы?

A. Для определения методов без реализации
B. Анонимные методы используются для создания экземпляров делегатов.
C. Для создания абстрактных методов

Что такое лямбда-выражения?

A. Упрощенная запись анонимных методов
B. Специальное математическое выражение
C. Метод абстрактного класса

Могут ли анонимные методы возвращать результат?

A. Не могут
B. Могут
C. Могут, если имеют входные параметры
D. Могут, если не имеют входных параметров

Выберите лишнее утверждение. Параметры лямбда-выражения и делегата

A. Должны иметь одинаковый тип
B. Не связаны друг с другом
C. Должны совпадать по количеству

Когда можно применять события?

A. Когда надо отследить определенные действия
B. Когда надо уведомить пользователя о чем-либо
C. Когда надо послать команду операционной системе
D. Ни в одном из перечисленных случае