|  |
| --- |
| EPAM Systems, RD Dep. |
| Контрольный тест  NET.C# Контрольные вопросы по ООП в C# |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial version | Oleg | 12.06.2014 |  |  |

ФИО Студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата опроса\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Пояснения:**

1. Студент заполняет колонки:
2. «Верно?». Если верно, то укажите «Да», если нет – «нет».
3. «Объяснения и комментарии»:
   * ответы на вопросы «Почему не правильно? Почему правильно?»;
   * при необходимости пошаговое описание с указанием состояния переменных на каждом шаге.
   * нужно изложить объяснение, которое покажет понимание теории и умение применять знания на практике
   * в некоторых вопросах и вариантах ответах нужно исследовать практическую задачу, разобраться и написать объяснение.
   * в некоторых вариантах желательно, указать как исправить код, чтобы он успешно компилировался или выполнялся без ошибок.
4. Используя стиль «EDU\_Student».
5. Преподаватель заполняет колонки «Верный ответ», «Правильное объяснение», «Зачтено».
6. А также задает вопросы и даёт свои комментарии в секции «Объяснения и комментарии».
7. Используя стиль «EDU\_Reviewer».
8. При выполнении можно и нужно пользоваться:
   1. Лекцией;
   2. MSDN;
   3. Учебниками вроде Шилда и т.п.
9. **Что такое Абстракция (как одна из концепций ООП)?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Выделение из всех возможных характеристик объекта те характеристики, которые важны при решении конкретной задачи. |  |  |  |

Сложность: [1]

1. **Что такое Инкапсуляция (как одна из концепций ООП)?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Сокрытие деталей внутренней реализации класса. Инкапсуляция необходима для того, чтобы каждый пользователь класса не мог сломать его поведение (нарушить инвариант, реализованный создателем класса). |  |  |  |

Сложность: [1]

1. **Что такое Наследование (как одна из концепций ООП)?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Возможность каждый раз не переписывать заново одну и ту же реализацию класса, а дописать новую к уже имеющейся. |  |  |  |

Сложность: [1]

1. **Что такое Полиморфизм (как одна из концепций ООП)?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Возможность использовать объекты классов с одинаковой реализацией без необходимости узнавать эту реализацию (один интерфейс – множество реализаций). |  |  |  |

Сложность: [1]

1. **Что такое класс? Что такое объект? Чем они отличаются? Какие между ними отношения?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Класс - сущность реального мира, описанная на языке программирования. Объект – конкретная реализация этой сущности со специфическим поведением. Отличие состоит в том, что класс описывает объект (хранит данные) и его поведение, а объект выполняет конкретные действия, предписанные его поведением и знанием о себе. Объект является одной из конкретных реализаций класса, который его описывает.  Например, классом можно считать чертежи, а объектом автомобили, изготовленные по этим чертежам. |  |  |  |

Сложность: [1]

1. **Что такое жизненный цикл объекта? Как он определяется?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Жизненный цикл – время жизни объекта. Жизненный цикл объекта включает:   1. Построение (инициализация). 2. Использование. 3. Уничтожение (например, освобождение памяти). |  |  |  |

Сложность: [1]

1. **Чем область видимость объекта отличается от жизненного цикла этого объекта?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Отличие состоит в том, что при окончании жизненного цикла объекта, он уничтожается.  Если объект имеет ограниченную область видимости, то он будет не доступен только там, где действует ограничение, по факту, при этом, оставаясь живым. |  |  |  |

Сложность: [1]

1. **Зачем вообще нужно это ООП? Без него никак нельзя было обойтись?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | ООП легко описывает реальный мир, т.к. все в языках такой парадигмы является объектами. Преимущество перед функциональными языками программирования в том, что можно описать множество сущностей со своим поведением, как объекты реального мира. Читаемость при этом улучшается, а количество кода уменьшается.  В таких языках создается большое количество модулей, которые выполняют только конкретные действия. Это позволяет добиться большой гибкости и многократно использовать написанный код. |  |  |  |

Сложность: [1]

1. **Какие минусы у ООП?**
   * Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Необходимо изучение принципов ООП и большого количества библиотек. Так же, необходимо уметь правильно проектировать разрабатываемое приложение. Еще, из-за того, что детали реализации сокрыты, то при изучении чего-либо, необходимо пользоваться документацией. |  |  |  |

Сложность: [3]

1. **Как виды полиморфизма поддерживает C#?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Верно? | Название | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Статический | Выбор реализации на этапе компиляции. Например, перегрузка, перекрытие, обобщения. |  |  |  |
|  | Динамический | Выбор реализации на этапе выполнения. Например, переопределение. |  |  |  |
|  |  |  |  |  |  |

Сложность: [3]

1. **Нужно ли создавать объект для вызова статического метода его класса? К каким членам можно обращаться изнутри статического метода?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Нет, не нужно, т.к. такой метод относится не к конкретной реализации, а к классу в целом.  Внутри статического метода можно обращаться только к статическим членам. |  |  |  |

Сложность: [1]

1. **Нарисуйте UML-диаграмму, подобную приведенным в главе, для перечисленных ниже классов и интерфейса.**

**• Абстрактный класс с именем HotDrink (Горячий напиток), методами Drink (Пить), AddMilk (Добавить молока) и AddSugar (Добавить сахара) и свойствами Milk (Молоко) и Sugar (Сахар).**

**• Интерфейс ICup (Чашка), содержащий методы Refill (Наполнить снова) и Wash (Помыть), а также свойства Color (Цвет) и Volume (Объем).**

**• Класс с именем CupOfCoffee (Чашка кофе), порожденный от класса HotDrink (Горячий напиток), поддерживающий интерфейс ICup и обладающий дополнительным свойством BeanType (Сорт кофе).**

**• Класс с именем CupOfTea (Чашка чая), порожденный от класса HotDrink, поддерживающий интерфейс ICup и обладающий дополнительным свойством LeafType (Сорт чая).**

* Нарисуйте на бумаге, сфоткайте и вставьте картинку в этот документ (не более 200 Кб).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  |  |  |  |  |

Сложность: [2]

1. **Напишите код для метода, который в качестве параметра принимает один из двух возможных в предыдущем примере объектов типа Cup... (CupOfCoffee или CupOfTea).**

**Эта функция должна вызывать методы AddMilk, Drink и Wash для любого передаваемого ей объекта Cup..**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Interface ICup  {  void AddMilk();  void Drink();  void Wash();  }  Void Method(ICup cup)  {  cup.AddMilk();  cup.Drink();  cup.Wash();  } |  |  |  |

Сложность: [2]

1. **Чем отличаются перегрузка и переопределение метода?**

* Приведите примеры.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Перегрузка – объявление метода с тем же названием, но разной сигнатурой.  Пример:  string GetMessage(int a, int b){}  string GetMessage(string a, string b){}  Переопределение – новая реализация метода (virtual/abstract) в классе-потомке.  Пример:  virtual int GetMessage(){}  overrider int GetMessage(){} |  |  |  |

Сложность: [2]

1. **Что такое конструктор?**

**Что такое конструктор по умолчанию и автоматический конструктор?**

**Чем отличаются?**

**Как вызвать конструктор родительского класса?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Конструктор – метод, с которого начинается жизненный цикл объекта. Конструктор необходим для инициализации полей объекта.  Конструктор по умолчанию – конструктор, не имеющий параметров. Автоматический конструктор – то же самое, что и конструктор по умолчанию. Разница в том, что автоматический конструктор явно не определяется. Если объект имеет конструкторы, кроме конструктора без параметров, то, чтобы использовать конструктор без параметров, его нужно явно определить.  Вызов конструктора родительского класса происходит с помощью ключевого слова base (например, base() – вызов конструктора по умолчанию). |  |  |  |

Сложность: [1]

1. **Что такое статический конструктор? Какие у него особенности?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Статический конструктор – конструктор помеченный как static. Класс может иметь только один статический конструктор. Особенность состоит в том, что он вызывается один раз при инициализации первого экземпляра класса или при обращении к его (класса) статическому полю. |  |  |  |

Сложность: [1]

1. **Что такое абстрактный класс? Какие у него особенности?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Абстрактный класс – класс помеченный как abstract. Такой класс может иметь абстрактные методы или поля (хотя не обязательно). Особенность класса состоит в том, что у такого класса не может быть создан экземпляр. Также, класс может наследовать неабстрактный класс или реализовывать интерфейсы. |  |  |  |

Сложность: [1]

1. **Что такое интерфейс? Какие у него особенности?**

**Что может содержать интерфейс?**

**Какие два варианта реализации интерфейса? Чем отличаются?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Интерфейс – ссылочный тип, который может содержать набор методов, являющийся протоколом связи интерфейса с реализуемыми его классами. Особенности: не может быть создан экземпляр интерфейса; класс может реализовывать множество интерфейсов; все методы в интерфейсе неявно public и не могут содержать другой модификатор доступа; содержат только описание методов (их сигнатуры и возвращаемые типы) без конкретной реализации; интерфейс может содержать помимо методов: поля, события, свойства.  Бывает явная и неявная реализации интерфейсов. Отличие состоит в том, что при явной реализации методов интерфейса необходимо указывать название интерфейса и метод через точку, например, I1.GetMesssage(){}. Кроме того, при явной реализации недопустимо использование модификаторов доступа.  Явная реализация используется, когда несколько интерфейсов, которые реализует класс, имеют одинаковые методы. |  |  |  |

Сложность: [2]

1. **Чем отличаются абстрактные классы и интерфейсы?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Отличия:   1. Наследование. Класс может реализовывать множество интерфейсов, но наследовать один класс. 2. Реализация. В абстрактном классе может быть описана стандартная реализация методов. 3. Модификаторы доступа. Члены интерфейсов неявно public и не могут быть помечены другим модификатором доступа. 4. Члены сущностей. Интерфейсы содержат только свойства и методы. |  |  |  |

Сложность: [2]

1. **Что такое тип object? Какие у него есть методы и свойства? Как думаете зачем его сделали?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Object – родительский тип для всех объектов.  Методы:   1. Equals(). 2. ReferenceEqaul(). 3. ToString(). 4. GetType().   Я думаю, что его сделали для совместимости типов, чтобы не привязываться к конкретной реализации. |  |  |  |

Сложность: [2]

1. **Что такое null? Что такое nullable?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Null – стандартное значение для reference type. Nullable – типы, допускающие значение Null.  Например, int 5 – значимый тип (null не допустимо); int? 5 – значимый тип (допустимо null). |  |  |  |

Сложность: [1]

1. **Что такое свойства? Как они устроены? Чем удобны? Чем отличаются от полей?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Свойства – пара методов, необходимых для правильной инкапсуляции поля. Свойства содержат два метода – get (accessor) и set (mutator).  Accesor необходим для предоставления информации и не должен содержать долгих вычислений.  Mutator – необходим для получения информации объектом с помощью ключевого слова value. Должен проверять значения и передавать уведомления.  Удобны тем, что можно ограничивать доступ к данным (public int A{get; private set;}).  Отличаются от полей те, что внутренняя реализация свойств – методы, поэтому свойства могут выполнять действия. |  |  |  |

Сложность: [2]

1. **Что такое свойства доступные только для чтения? Когда можно присвоить им значение? Почему именно так?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Свойства доступные только для чтения => Public int A{get;}  Присвоить значение им можно так же, как и readonly полям, т.е. при объявлении или в конструкторе.  Это необходимо, чтобы значение свойства не изменялось во всем жизненном цикле объекта. |  |  |  |

Сложность: [2]

1. **Каким образом можно проверить совместимость типа объекта или узнать тип объекта?**

* Приведите все варианты и примеры. Объясните чем они отличаются и когда следует применять.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | При использовании ключевого слова is. Выражение возвращает bool значение (true, если совместимы, false – в противном случае). Используется в операторах ветвления или, когда точно не известно, что типы совместимы.  Пример:  If (a is MyType)  return; |  |  |  |
|  | При использовании ключевого слова as. Выражение возвращает объект типа, после as, если типы совместимы или null – в противном случае. Используется, когда точно известно, что типы совместимы (явное приведение). Операция дольше, чем is.  BType b = a as BType; |  |  |  |
|  | При использовании GetType() можно узнать тип объекта.  Пример:  Type t = o.GetType(); |  |  |  |

Сложность: [2]

1. **Что такое деструктор? Какие особенности деструкторов в C#?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Деструктор – метод, который необходим для освобождения ресурсов. Особенности в C#:   1. Деструктор может быть явно определен (~SomeClass()), но не может быть явно вызван. 2. Не может быть определен в структуре (только в классе). 3. Класс может иметь только один деструктор. 4. Не может быть унаследован или перегружен. 5. Имеет название класса, как и конструктор. 6. Не может иметь модификаторов или параметров. |  |  |  |

Сложность: [2]

1. **Какие модификаторы доступа есть в С#?**

* Описывайте их в порядке возрастания уровня доступа.
* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Private – доступ только внутри класса. Применяется к классам, методам, полям. |  |  |  |
|  | Protected – доступ только внутри базового класса или в классе-потомке. Применяется для методов и полей. |  |  |  |
|  | Internal – доступ внутри сборки. |  |  |  |
|  | Protected internal – protected + internal. |  |  |  |
|  | Pubic – доступен везде. |  |  |  |

Сложность: [1]

1. **Какая область видимости у членов класса по умолчанию (если не указан явно)?**

**Какая у членов перечислений?**

**У членов интерфейсов?**

**У классов?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Scope для членов класса по умолчанию – private, для членов перечислений – public, для членов интерфейсов – public, для классов – internal. |  |  |  |

Сложность: [2]

1. **Что такое константы?**

**Какие ограничения для типов значений констант?**

**Чем отличаются от «свойств доступных только для чтения»?**

**Как выглядят константы после компиляции в «промежуточный код»?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Константы – неизменяемые значения. Константами могут быть только примитивные типы (не могут быть перечисления). От свойств доступных только для чтения отличатся тем, что значение константам должно быть присвоено сразу же после объявления. |  |  |  |

Сложность: [3]

1. **Какие принципы были сформулированы Аланом Кеем?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | 1. Все является объектом. 2. Каждый объект является представителем класса, который выражает общие свойства объекта. 3. В классе задается поведение объекта. Все объекты, являющиеся экземплярами одного класса, могут выполнять одни и те же действия. 4. Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования. 5. Каждый объект имеет независимую память, которая состоит из других объектов. 6. Вычисления осуществляются путем взаимодействия между объектами, при котором один объект требует, чтобы другой объект выполнил некоторое действие. |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Сложность: [1]