Asipuk Raman

1. Перечислите на какие группы деляться все типы данных в .NET

Типы значений (значимые типы) и ссылочные типы.

2. Отличия ссылочных и значимых типов

Тип значения хранит данные непосредственно, а ссылочный тип хранит ссылку на значение.

3. Отличия в применении ключевых слов *ref*/*out* при передаче параметров по ссылке

Out обязательно должен быть инициализирован в методе, в которой передают этот параметр.

4. Что такое упаковка/распаковка (*boxing*/*unboxing*) и какой из этих процессов больше влияет на производительность (занимает больше времени)

Boxing – value type –> ref type (занимает больше времени)

Unboxing – ref type –> value type

5. Опишите что происходит при создании объектов с использованием ключевого слова *new*

1. расчитывается необходимое кол-во памяти для создаваемого объекта (кастомные филды + служебные)

2. выделяется память на куче

3. вызывается необходимый конструктор

4. возвращается ссылка на созданный объект

6. Нужно ли определять список параметров в конструкторе типа структуры

Не обязательно. В случае вызова конструктора по умолчанию, подставятся дефолтные значения ***(default of T).***

7. Какое наследование можно реализовать для структур

Интерфейсов.

8. К какой группе типов относяться перечисления (*enum*) - ссылочному или значимому

Значимому.

9. Что позволяет делать тип *dynamic*

Определять тип объекта в runtime.

10. В каких случаях сборщик мусора начинает процесс сборки

1. Недостаток физической памяти
2. Переполнение кучи
3. Вызов ***GC.Collect()***

11. Отличия поверхностного (shallow) и глубокого (deep) копирования (клонирования) объектов

Shallow – копирует объект без внутренних ссылок на другие объект (устанавливает как ***null***)

Deep – копирует объект со всеми внутренними ссылками на другие объекты (нужно реализовать ***ICloneable***)

12. Какой вид копирования объектов реализован в защищенном (*protected*) методе *MemberwiseClone* типа *Object*

Shallow copy.

13. Назначение интерфейса *ICloneable*

Настройка создания копии существующего объекта.

14. Перечислите базовые шаги для определения события в классе

1. определить поле типа ***event***

2. определить методы, который будет вызывать событие (зачастую как **On{*EVENT NAME*}(EventArgs e) {}**)

15. Какие сущности языка C# могут быть обобщенными (generic)

1. Интерфейс

2. Клас, абстрактный класс

3. Структура

4. Делегат

5. Метод, статический метод

16. Ограничения в обобщениях

Основные:

1. class – только ссылочный тип

2. struct – только значимый тип

3. new() – не абстрактый тип (можно создать экземпляр)

4. Интерфейс – наследуется от опеределенного интерфейса

5. Класс – наследуется от опеределенного класса

17. Можно ли реализовать перегрузку операторов для обобщенных типов

Да, можно.

18. Назначение типа *Nullable<T> (?)*

Возможность установки ***null*** значения для значимых типов.

19. Как обработать исключение в середине выполняющейся цепочки делегатов с сохранением выполнения других делегатов в цепочке

1. пройтись по всей цепочке делегатов (GetInvocationList())
2. в try вызывать делегат (DynamicInvoke())
3. catch ингоре/лог

20. Приведите возможные источники утечек памяти в .NET

1. Не освобожденные unmanaged-ресурсы

21. Что такое слабые ссылки (weak reference)

22. Назначение метода финализации

Очистка неуправляемых ресурсов.

23. Как рекомендуется реализовывать вызов метода финализации в типах

1. наследование ***IDisposable***
2. реализация внутренего метода ***Dispose(bool disposing)***
3. реализация интерфейса ***IDisposable***
4. реализация деструктора

24. Можно ли обрабатывать исключения внутри блока using

Да, можно.

25. Приведите фрагмент кода, позволяющий пробросить перехваченное исключение вызвающему коду с сохранением трассировки стэка

```
try
{
	throw new Exception();
}
catch
{
	throw;
}
```

24. Для чего нужны домены приложений

1. изоляция приложения от других

2. менеджмент сборок

25. Приведите пример использования доменов приложений

```
var domain = AppDomain.CurrentDomain;
var assemblies = domain.GetAssemblies();
```

26. Перечислите ключевые механизмы .NET

1. exception

2. GC

3. domain app

4. serialization

5. reflection

27. Что означает принцип just-in-time compiling

Компилирование участков кода в машинный код во время выполнения программы.

28. На чём основана работа механизма рефлексии (reflection)

Основана на чтении метаданных.

29. Что такое сериализация

Преобразование информации объекта в поток байтов.

30. Стандартные форматы сериализации данных .NET

1. Binary
2. XML
3. JSON
4. SOAP

31. Аттрибуты для управления процессом сериализации

1. ***[Serializable]***
2. ***[NonSerialized]***

32. Отличия интерфейсов *IEnumerable* и *IQueryable*

IEnumerable – манипулирует объектами в памяти на сторое клиента

IQueryable – манипулирует объектами на стороне сервера

33. Для чего нужны интерфейсы

Абстрагироваться от конкретики и манипулировать лишь поведением, «договором», который должен иметь тип, реализующий интерфейс.

34. Основное свойство логирования

Разрешение непонятных ситуаций, когда приложение ведет себя странным образом, и нужно найти причину такого поведения.

35. Что такое объектно-реалиционное отображение и какие возможности оно предоставляет разработчикам

ORM – работа с БД ввиде ООП подхода, манипулируя абстракциями.

Даёт возможность работы с данными ввиде классов, и наоборот, данные классов ввиде данных БД.

36. Опишите кратко работу с Database First принципом работы в Entity Framework

1. создаём бд
2. создаём таблицы
3. настраиваем связи в таблицах
4. подлючаемся к бд
5. импортируем сущности, конфигурацию и контекст

37. Опишите кратко работу с Code First принципом работы в Entity Framework

1. описываем сущности
2. настраиваем связи между сущностями
3. создаём контекст
4. добавляем ***DbSet***'ы

38. Назначение кэширования и как оно может быть реализовано

Кэширование используется для сохранения наиболее часто используемых данных или для оптимизации обращени к этим данным. Реализации: in-memory, cloud.

39. Какие SOLID принципы гарантируют слабую связь между компонентами

1. принцип подстановки Лисков

2. принцип инверсии зависимостей

40. Каким требованиям должны удовлетворять юнит-тесты (AAA/FIRST)

AAA:

1. разбитие кода внутри теста на три секции (Arrange section, Act section, Assert section)

2. быстро запускаемые

3. независимые от других тестов

4. при одих и тех же входных данных, возвращать одинаковый результат

5. правильный нейминг конвеншин

6. новая фича => покрытие тестами