Вопросы по дисциплинам

“Объектно-ориентированные технологии программирования и стандарты проектирования”

1- 40 01 01- Программное обеспечение информационных технологий (специализация «Программирование интернет-приложений»)

2022/2023 гг.

1. Состав. NET Framework. Структура среды выполнения CLR.
2. Структура управляемого модуля - portable executable (PE). Понятие и исполнение сборки. CIL.
3. CTS (Common Type System). Типы данных C#. Ссылочные и типы значений.
4. Понятие упаковки и распаковки типов. Типы Nullable: преобразование, проверка, null-объединение
5. Тип данных String: операции, литералы, пустые и нулевые строки, форматированный вывод.
6. Неявная типизация – назначение и использование.
7. Массивы C# одномерные, прямоугольные и ступенчатые.
8. Понятие кортежей. Свойства, создание
9. Принципы объектно-ориентированного программирования.
10. Класс. Элементы класса. Свойства и индексаторы.
11. Класс. Константы. Поля только для чтения. Инициализаторы класса.
12. Спецификаторы доступа C#. Видимость типов. Доступ к членам типов.
13. Класс. Конструкторы и их свойства. Деструкторы
14. Класс и методы System.Object.
15. Статические методы и статические конструкторы класса.
16. Статические классы. Методы расширения и правила их определения.
17. Анонимные типы.
18. Модификаторы параметров - ref , out, params. Необязательные и именованные аргументы.
19. Перегрузка методов и операторов. Правила перегрузки операторов.
20. Операции преобразования типа. Явная и неявная форма. Ограничения.
21. Вложенные типы. Вложенные объекты
22. Правила наследования C#.
23. Сокрытие имен при наследовании. Обращение к срытым членам
24. Использование операций is и as
25. Полиморфизм. Виртуальные методы, свойства и индексаторы. Правила переопределения.
26. Понятие раннего и позднего связывания.
27. Абстрактные классы и методы. Бесплодные классы.
28. Структур в C#.
29. Интерфейсы. Свойства интерфейсов. Реализация интерфейсов.
30. Явная и неявная реализация интерфейсов. Работа с объектами через интерфейсы.
31. Ковариантность интерфейсов. Контравариантность интерфейсов
32. Стандартные интерфейсы .NET. Назначение и применение.
33. Исключительные ситуации. Генерация и повторная генерация исключений.
34. Исключительные ситуации. Варианты обработки исключений. Фильтры исключений
35. Обобщения (generics). Свойства обобщений.
36. Концепция ограничений обобщений. Статические члены обобщений.
37. Делегаты. Определение, назначение и варианты использования. Обобщенные делегаты.
38. Анонимные функции. Лямбда-выражения.
39. Обобщённые делегаты .NET. Action, Func, Predicate
40. События и делегаты.
41. Стандартные коллекции .NET. Типы коллекций.
42. Стандартные интерфесы коллекций.
43. IEnumerable и IEnumerator
44. LINQ to Objects. Синтаксис. Форма. Возврат результата. Грамматика выражений запросов. Отложенные и неотлаженные операции.

45. LINQ to Objects. Операции Where, Select, Take, OrderB, Join, GroupBy
46. Рефлексия. System Type.
47. Классы для работы с файловой системой.
48. Синтаксическая конструкция using. Чтение и запись файлов. Потоковые классы.
49. Классы адаптеры потоков.
50. Сериализация. Форматы сериализации.
51. Сериализация контрактов данных. интерфейс ISerializable.
52. Атрибуты. Создание собственного атрибута.
53. Процесс. Домен приложений. Поток выполнения.
54. Создание потоков , классы приоритетов. Состояния потоков
55. Синхронизация потоков. Lock. Monitor. Mutex. Semaphore
56. Библиотека параллельных задач TPL. Класс Task. Состояние задачи.
57. Способы создания Task. Возврат результата. Отмена выполнения задач. Продолжения.
58. Параллелизм при императивной обработке данных. Класс Parallel
59. Асинхронные методы. async и await
60. Проектирование отношений. Агрегация, композиция и ассоциация
61. Антипаттерны проектирования. Рефакторинг. Методы рефакторинга.
62. Чистый код. Требования к именам, функциям, форматированию.
63. Чистый код. Требования к классам и объектам. Закон Деметры. DTO. Избыточный код.
64. Паттерны проектирования. Классификация. Порождающие Abstarct Factory, Factory Method
65. Порождающие паттерны проектирования :Builder, Object pool, Prototype.
66. Порождающие паттерны проектирования: Singleton – 4 реализации, Lazy initialization,
67. Структурные паттерны: Adapter, Decorator
68. Структурные паттерны: Composite, Facade
69. Структурные паттерны: Proxy, Bridge
70. Паттерны поведения: Chain of Responsibility, Command
71. Паттерны поведения: Iterator, Mediator, Visitor
72. Паттерны поведения: Memento, Observer, Null Object, Strategy
73. Принципы проектирования SOLID.

Темы задач

C# базовый	Типы, кортежи, класс, структуры, конструкторы свойства, индексаторы, константы, доступность, переопределения, преобразования, массивы, наследование, интерфейс, станд. интерфейсы, статика, виртуальность, коллекции, делегаты, стандартные делегаты, события, обобщения, исключения, методы расширения, перегрузка операций, анонимные функции, лямбда выражения, LINQ, String, потоковые классы, классы для работы с файловой системой, рефлексия, сериализация, ***процессы, потоки – управление и синхронизация, Task, TPL, паттерны .*** *Пример задачи:* **№ 1** **11. Создайте интерфейс Figure с виртуальным методом print. Создать класс Rectangle с координатами x, y, длиной и шириной h, l и color (строка) цветом и реализацией интерфейса. Метод print должен выводить прямоугольник на консоль. Класс должен содержать: конструктор без параметров и с тремя, пятью параметрами (используйте делигирование конструкторов).** **2. Переопределите в Rectangle метод ToString, оператор + int (добавление к ширине и высоте целого числа), метод вычисления площади. Создайте коллекцию List<> из 6 прямоугольников. Продемонстрируйте работу оператора + и метода print.** **3. Используя LINQ отсортируйте ее по x, а затем по y, затем по площади, возьмите первый и последний объекты и выведите их.** **4. Сериализуйте коллекцию в формате json, координаты x,y – сделайте не сериализуемыми.**

	5. Создайте потокобезопасный класс RectengleSet на основе паттерна Singleton. Класс содержит Rectangle. В конструкторе создается Rectangle. Продемонстрируйте получение доступа к Rectangle несколько раз и докажите, что это один и тот же объект.