Вопросы к экзамену по дисциплине «Основы алгоритмизации и программирования» для студентов 2-го курса специальности 09.02.07 «Информационные системы и программирование»

1. Понятие объектно-ориентированного программирования

ООП – технология создания сложного программного обеспечения, которая представляет собой программный продукт в виде совокупности объектов.

1. Понятия класса и объекта в ООП

Объект – модель реальной сущности в программной системе.

Класс – описание предметной области.

1. Принципы ООП

В ООП существует ряд принципов, лежащих в основе правил создания и использования всех структурных элементов, включая классы, объекты, методы и прочие компоненты:

1) Инкапсуляция

2) Наследование

3) Полиморфизм

1. Принцип инкапсуляции в ООП

Инкапсуляция – объединение в единое целое данных и методов обработки этих данных, позволяющие изолировать объект от внешнего окружения.

Пример:

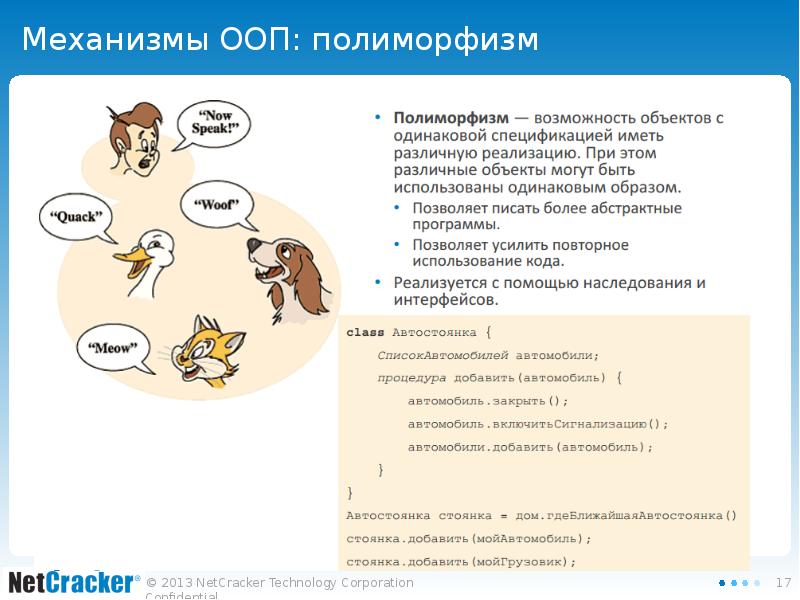
Пользователь может взаимодействовать с объектом только через этот интерфейс. Реализуется с помощью ключевого слова: public.

Пользователь не может использовать закрытые данные и методы. Реализуется с помощью ключевых слов: private, protected, internal.

1. Принцип полиморфизма в ООП

Полиморфизм – возможность использовать одинаковые имена для методов, входящих в различные классы, таким образом имеющих различные алгоритмы.

Пример: вычисление периметра в четырёхугольнике, параллелограмме, квадрате называется одинаково, но реализация разная.



1. Принцип наследования в ООП

Наследование – возможность описать новый класс на базе существующего, при этом свойства и функциональность родительского класса заимствуется новым классом.

Пример:



1. Понятие исключения, виды исключений, способы обработки исключений

Исключение - событие периода выполнения, которое может стать причиной отказа программы.

Виды:

- Аппаратные. Генерируются процессором. К ним относятся, например:

* деление на 0;
* выход за границы массива;
* обращение к невыделенной памяти;
* переполнение разрядной сетки.

- Программные.

Генерируемые операционной системой и прикладными программами – возникают тогда, когда программа их явно инициирует. Когда встречается аномальная ситуация, та часть программы, которая ее обнаружила, может сгенерировать, или ***возбудить***, исключение.

1. Понятие оптимизации программного кода, способы оптимизации

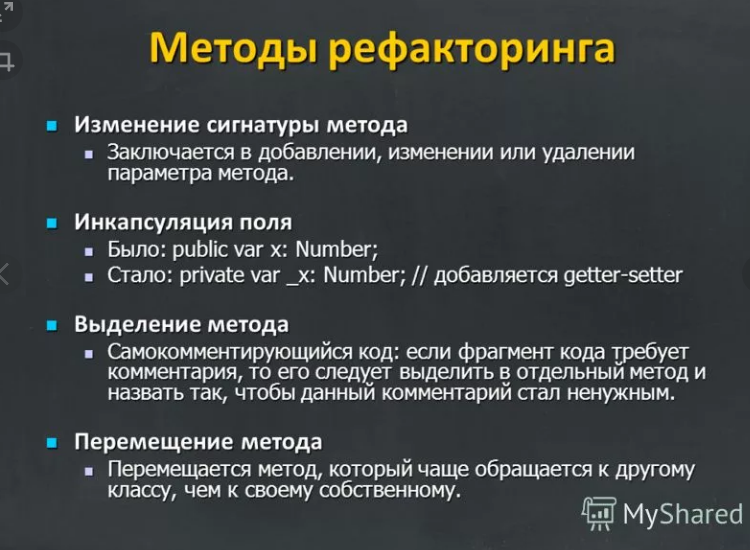
Оптимизация – обработка, связанная с переупорядочиванием и изменением операций в компилируемой программе с целью получения более эффективной результирующей объектной программы.

Способы:

1. Линейных участков программы
2. Логических выражений
3. Циклов
4. Вызов процедур и функций
5. Других конструкций входного языка
6. Понятие рефакторинга программного кода, способы рефакторинга

Рефакторинг – процесс изменения программы, при котором не меняется внешнее поведение кода, но улучшается его внутренняя структура.

Способы:



1. Понятие тестирования программного обеспечения. Виды тестирования

Тестирование ПО **—** процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов.

Виды:

1. Функциональные — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.
2. Нефункциональные:

- контролирует работоспособность системы под различными нагрузками;

- адаптация приложения для различных платформ;

- требования к масштабируемости приложения;

- гарантия продолжения работы приложения в случаях непредвиденных ситуаций.

1. Связанные с изменениями -тестирование, связанное с изменениями предназначено для проверки исправления дефектов и проверки работоспособности системы после внесения изменений, таких как добавление нового функционала или корректировка старого.
2. Понятие отладки программного обеспечения. Способы отладки

Отладка - это поиск (локализация), анализ и устранение ошибок в программном обеспечении, которые были найдены во время тестирования.

Методы:

* ручного тестирования - при обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка.
* Индукции - Метод основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке.
* Дедукции - по методу дедукции вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки. Затем анализируя причины, исключают те, которые противоречат имеющимся данным.
* обратного прослеживания - строится гипотеза о значениях основных переменных, которые могли бы привести к получению имеющегося результата. Исходя из этой гипотезы, делают предложения о значениях переменных в предыдущей точке. Процесс продолжают, пока не обнаружат причину ошибки.

1. Понятие алгоритма. Свойства алгоритмов
2. Способы записи алгоритма
3. Базовые алгоритмические структуры, их описание
4. Понятие интегрированной среды разработки (ИСР). Примеры ИСР. Состав ИСР
5. Язык программирования C#. Возникновение, основное назначение, особенности
6. Понятие интерфейса. Виды интерфейсов
7. Принципы разработки интерфейса
8. Понятие жизненного цикла программного обеспечения. Модели жизненного цикла
9. Этапы разработки приложений
10. Понятие строки в C#. Методы строк
11. Понятие массива в C#. Одномерные и многомерные массивы, принципы работы с массивами
12. Понятие множества в C#. Класс HashSet – свойства, методы
13. Понятие списка в C#. Класс List<T> - свойства, методы
14. Понятие файла в C#. Классы StreamWriter и StreamReader – свойства, методы
15. Принцип модульности при разработке приложений. Добавление класса в проект и его использование
16. Создание и использование библиотеки класса