

Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

УТВЕРЖДЕНО:

Директор

А. В. Замятин

Оценочные материалы по дисциплине

Технологии программирования

по направлению подготовки / специальности

10.05.01 Компьютерная безопасность

Направленность (профиль) подготовки/ специализация:

Анализ безопасности компьютерных систем

Форма обучения

Очная

Квалификация

Специалист по защите информации

Год приема

2024

СОГЛАСОВАНО:

Руководитель ОП

В.Н. Треньяев

Председатель УМК

С.П. Сущенко

Томск – 2024

1. Компетенции и индикаторы их достижения, проверяемые данными оценочными материалами

Целью освоения дисциплины является формирование следующих компетенций:

ОПК-13 Способен разрабатывать компоненты программных и программно-аппаратных средств защиты информации в компьютерных системах и проводить анализ их безопасности.

ОПК-19 Способен оценивать корректность программных реализаций алгоритмов защиты информации.

ОПК-7 Способен создавать программы на языках высокого и низкого уровня, применять методы и инструментальные средства программирования для решения профессиональных задач, осуществлять обоснованный выбор инструментария программирования и способов организации программ.

ПК-3 Способен проектировать программно-аппаратные средств защиты информации компьютерных систем и сетей.

Результатами освоения дисциплины являются следующие индикаторы достижения компетенций:

ИОПК-13.1 Предпринимает необходимые действия по сбору и анализу исходных данных для проектирования компонент программных и программно-аппаратных средств защиты информации в компьютерных системах

ИОПК-13.2 Определяет параметры функционирования, архитектуру и интерфейсы компонент программных и программно-аппаратных средств защиты информации в компьютерных системах

ИОПК-19.1 Обладает знанием формальных приёмов, правил, алгоритмов, технологий создания и документирования контрольных примеров и тестовых наборов данных

ИОПК-19.2 Осуществляет подготовку тестовых наборов данных в соответствии с выбранной методикой, а также проверку работоспособности программного обеспечения на основе разработанных тестовых наборов данных

ИОПК-19.3 Осуществляет сбор и анализ полученных результатов проверки работоспособности программного обеспечения, оценку соответствия программного обеспечения требуемым характеристикам

ИОПК-7.3 Демонстрирует навыки создания программ с применением методов и инструментальных средств программирования для решения различных профессиональных, исследовательских и прикладных задач

ИОПК-7.4 Осуществляет обоснованный выбор инструментария программирования и способов организации программ

ИПК-3.1 Разработка технических заданий, эскизных, технических и рабочих проектов работ по защите информации

ИПК-3.2 Разработка проектов программных и аппаратных средств защиты информации в соответствии с техническим заданием

2. Оценочные материалы текущего контроля и критерии оценивания

Элементы текущего контроля:

- контрольная работа;
- лабораторные работы;

Контрольная работа (ОПК-13, ОПК-7, ОПК-19, ПК-3)

Контрольная работа состоит из трёх вопросов, 2 теоретических и 1 задача. Теоретические вопросы формируются из таких тем:

– Принципы SOLID. Здесь необходимо рассказать об одном из принципов SOLID, на конкретном примере показать нарушение принципа, а также предложить решение, исправляющее возникшее нарушение.

Пример вопроса.

Задание №1

1. Базовые принципы проектирования (SOLID). Принцип – единственной ответственности. На представленном примере определить, существует ли нарушение принципа единственной ответственности. Если да, то предложите решение. Ответ обоснуйте.

Легенда

Рассмотрим следующий фрагмент псевдокода. Здесь представлен класс, обеспечивающий перенос данных с одного сервера на другой сервер.

```
//Класс обеспечивающий работу с данными типа время в формате
class DateTime;
//Класс для хранения параметров по подключению к серверу
class ServerData;
//Класс обеспечивающий перенос данных с одного сервера на другой сервер.
class DataMigrationFromServer
{
private:
Server server;
public:
    DataMigrationFromServer() {}
    //Получение данных
    list<ServerData> *GetData(DateTime initialDate, DateTime endDate) {
        //Получение данных с сервера
        return new list<ServerData>();
    }
    //Обработка данных
    list<ServerData>* ProcessData(list<ServerData> &rawData) {
        //Применяем пользовательские правила по обработке,
        //полученных серверных данных
        return &rawData;
    }
    //Перенос данных
    void Migrate(list<ServerData>* rawData, Server serverUSR) {
        //Перенос данных от сервера к серверу serverUSR
    }
};
int main()
{
    Server newConnection("ssh user@196.134.97.129");
    DateTime StartDate("12","01","2023"), EndDate("12", "01", "2024");
    DataMigrationFromServer* dataMigrater = new DataMigrationFromServer();
    //Получение данных с сервера
    list<ServerData> * rawData = dataMigrater->GetData(StartDate, EndDate);
    //Обработка данных
    list<ServerData> * processedData = dataMigrater->ProcessData(*rawData);
    //Перенос данных на другой сервер
    dataMigrater->Migrate(processedData, newConnection);
    return 0;
}
```

– Паттерны проектирования. Здесь необходимо рассказать о предлагаемом паттерне, его назначении, продемонстрировать его архитектуру и способ применения на примере простой задачи.

Пример вопроса.

Задание №2

Паттерн «Шаблонный метод». Рассмотреть назначение и архитектуру *шаблонного метода*. Применить к предложенной легенде рассматриваемый паттерн. Решение представить в виде кода с подробными объяснениями в онлайн компиляторе, также предоставить UML диаграмму для предложенного решения.

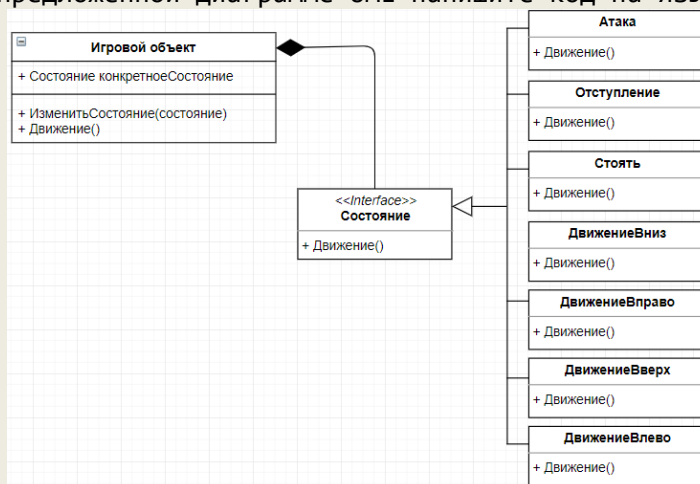
Легенда

Необходимо реализовать класс класса под названием **SecurityManager**, в котором присутствует метод по созданию пользователя в системе (void CreateUser(string username, string realName, string password). При создании пользователя выполняются следующие действия: шифруется пароль пользователя, вставляются все необходимые записи в базу данных, а также, по желанию, заполняется журнал аудита о создании пользователя.

– Предлагается решить задачу, которая сводится к написанию кода по предложенной UML диаграмме.
Пример задачи.

Задание №3

По предложенной диаграмме UML напишите код на языке C++.



Критерии оценивания:

Результаты контрольной работы определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Оценка «отлично» выставляется, если даны правильные ответы на все теоретические вопросы и все задачи решены без ошибок.

Оценка «хорошо» выставляется, если даны ответ на все вопросы и все задачи решены, но присутствуют неточности в ответах, в решении задач.

Оценка «удовлетворительно» выставляется, если даны ответы на все вопросы, но при этом наблюдается не полное раскрытие темы, в решении задач присутствуют ошибки, но при этом ход решения верный.

Оценка «неудовлетворительно» выставляется, если нет правильных ответов на теоретические вопросы, в решении задачи предлагается неверное решение.

Варианты лабораторных заданий (ИОПК-19.1, ИОПК-19.2, ИОПК-19.3, ИПК-3.1, ИПК-3.2, ИОПК-7.3, ИОПК-7.4)

Примеры лабораторных работ

Задание №1

Запрограммируйте шаблонный класс, реализующий стек. Класс должен поддерживать следующие операции:

1. Помещение объекта в стек;
2. Извлечение объекта из стека;
3. Получение размерности стека.

В случае попытки вызова операции извлечение объекта из стека при условии, что стек пуст, должно генерироваться исключение класса **EStackEmpty**(наследник класса EStackException). Данный класс должен содержать публичный метод `char* what()`, возвращающий диагностическое сообщение

Задание №2

На основе паттерна Наблюдателя, написать программу для слежения за состоянием выбранного файла.

Ограничимся двумя характеристиками:

1. Существует файл или нет;
2. Каков размер файла.

Программа будет выводить на консоль уведомление о произошедших изменениях в файле.

Существует несколько ситуаций для наблюдаемого файла

1. Файл существует, файл не пустой - на экран выводится факт существования файла и его размер.
2. Файл существует, файл был изменён - на экран выводится факт существования файла, сообщение о том, что файл был изменён и его размер.
3. Файл не существует - на экран выводится информация о том, что файл не существует.

В главной программе создаём объект для отслеживания состояния конкретного файла,

затем определяется объект наблюдатель, после чего связываем Наблюдателя с Источником.

Можно использовать бесконечный цикл, в котором будем обновлять состояние объекта Источника каждые 100 миллисекунд (например так `std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );`).

Текущий контроль успеваемости проводится во время сдачи лабораторных работ.

Каждая работа оценивается оценками «зачтено»/ «не зачтено» в соответствии со следующими критериями:

- предложенные реализации программ являются корректными и решают поставленную задачу;
- студент уверенно отвечает на вопросы, связанные как по предложенной реализации, так и по технологиям, примененным к предложенному решению;
- даёт полные ответы на вопросы по теории из соответствующего раздела курса;
- умеет исправлять ошибки и оперативно вносить изменения в программу.

3. Оценочные материалы итогового контроля (промежуточной аттестации) и критерии оценивания)

Зачёт с оценкой проводится в письменной форме по билетам. Билет содержит теоретический вопрос и практическую задачу. Студент письменно готовит ответ на вопросы в билете, решение практической задачи, после чего, в устной форме объясняет/защищает преподавателю подготовленный материал.

Примеры контрольных вопросов (ОПК-13, ОПК-7, ОПК-19, ПК-3)

1. Базовые принципы проектирования (SOLID). Принцип единственной ответственности.
2. Базовые принципы проектирования (SOLID). Принцип открытости/закрытости.
3. Базовые принципы проектирования (SOLID). Принцип подстановки Барбары Лисков.
4. Базовые принципы проектирования (SOLID). Принцип разделения интерфейсов.
5. Базовые принципы проектирования (SOLID). Принцип инверсии зависимости.
6. Паттерны проектирования. Абстрактная фабрика.
7. Паттерны проектирования. Шаблонный метод.
8. Паттерны проектирования. Наблюдатель.
9. Паттерны проектирования. Адаптер. Строитель.

10. Паттерны проектирования. MVC.
11. GRASP паттерны.
12. Идиомы RAII, CRTP.
13. Методологии управления проектами: Agile.
14. Методологии управления проектами: Scrum.
15. Методологии управления проектами: Kanban.

Примеры практических задач (ИОПК-19.1, ИОПК-19.2, ИОПК-19.3, ИПК-3.1, ИПК-3.2, ИОПК-7.3, ИОПК-7.4)

Задача 1.

Паттерн Наблюдатель. Его назначение, архитектура. Рассмотрите предложенную легенду, примените к ней паттерн наблюдатель. Решение представить в виде кода с подробными объяснениями.

Легенда

В вузе есть разные библиотеки по разным дисциплинам. Когда студент зачислен на дисциплину, ему рекомендуется присоединиться к соответствующей библиотеке. Библиотека содержит собрание книг. Каждый раз, когда новая книга добавляется в библиотеку, она уведомляет всех участников.

Участвующие классы:

Книга - имеет идентификатор и имя;

Библиотека - есть коллекция книг;

Студент - имеет идентификатор и имя;

Некоторый вуз - имеет набор студентов по разным дисциплинам и соответствующую библиотеку по каждой дисциплине.

Задача 2.

Паттерн стратегия. Его назначение, архитектура. Рассмотрите предложенную легенду, примените к ней паттерн стратегия. Решение представить в виде кода с подробными объяснениями.

Легенда

Рассмотрим индикатор выполнения – это окно, которое приложение может использовать для индикации хода длительности операции (например, процесса установки). Обычно это прямоугольное окно, которое постепенно заполняется слева направо цветом выделения по мере выполнения операции. У него есть диапазон и текущая позиция. Диапазон представляет собой всю продолжительность операции, а текущая позиция представляет прогресс, достигнутый приложением в завершении операции. Диапазон и текущая позиция используются для определения процента индикатора выполнения, который нужно заполнить цветом выделения. Существуют различные направления заполнения, такие как справа налево, сверху вниз и снизу вверх, также с заданным направлением заливки можно использовать различные типы заливок, такие как непрерывная заливка, прерывистая заливка или заливка на основе узора.

Задание.

Для пользовательского приложения реализовать возможность настройки индикатора выполнения с конкретным классом-заполнителем.

Студент допускается к зачёту с оценкой, в том случае, если в течение семестра успешно сдал все лабораторные работы по курсу.

Если студент в течении семестра успешно написал контрольную работу, защитил все лабораторные работы, то зачёт с оценкой проставляется *автоматически*.

4. Оценочные материалы для проверки остаточных знаний (сформированности компетенций) (ОПК-13, ОПК-7, ОПК-19, ПК-3, ИОПК-19.1, ИОПК-19.2, ИОПК-19.3, ИПК-3.1, ИПК-3.2)

1. Базовые принципы проектирования (SOLID). Принцип единственной ответственности.
2. Базовые принципы проектирования (SOLID). Принцип открытости/закрытости.
3. Базовые принципы проектирования (SOLID). Принцип подстановки Барбары Лисков.
4. Базовые принципы проектирования (SOLID). Принцип разделения интерфейсов.
5. Базовые принципы проектирования (SOLID). Принцип инверсии зависимости.
6. Паттерны проектирования. Абстрактная фабрика.
7. Паттерны проектирования. Шаблонный метод.
8. Паттерны проектирования. Наблюдатель.
9. Паттерны проектирования. Адаптер. Строитель.
10. Паттерны проектирования. MVC.
11. GRASP паттерны.
12. Идиомы RAII, CRTP.

Требуется описать назначения принципов проектирования, паттернов проектирования объектно-ориентированных систем, идиом проектирования, привести конкретные примеры.

Информация о разработчиках

Андреева Валентина Валерьевна, к.т.н., доцент, каф. компьютерной безопасности, ИПМКН.