**Тема: «Работа с файлами»**

**Задание 1: Основные операции с файлами**

**Задание НЕ ПО ВАРИАНТАМ. Все указанные пункты нужно сделать каждому варианту. Файл для варианта будет называться так «ivanov.ii» (где ivanov это фамилия, а ii – это инициалы имени и отчества)**

Каждая задача включает:

* Создание, копирование, удаление и изменение файлов.
* Получение информации о файлах (размер, дата создания, дата последнего изменения).

**Классы:**

1. **FileManager** – отвечает за создание, удаление, копирование, перемещение файлов.
2. **FileInfoProvider** – получает информацию о файле (размер, дата создания, дата изменения).

**Примеры задач:**

1. Создать файл, записать в него текст, прочитать и вывести в консоль.
2. Проверить существование файла перед его удалением.
3. Получить информацию о файле (размер, даты).
4. Скопировать файл и убедиться, что копия существует.
5. Переместить файл в новую директорию.
6. Переименовать файл в файл familiya.io
7. Обработать ошибку при удалении несуществующего файла.
8. Сравнить два файла по размеру.
9. Удалить все файлы в папке, соответствующие определенному шаблону. С расширением ii (см. выше)
10. Вывести список всех файлов в заданной директории.
11. Запретить запись в файл и попытаться записать в него.
12. Проверить доступные права к файлу (чтение, запись, выполнение).

**Задание 2: Запись данных в файлы (без сериализации)**

**Описание**

В этом задании каждая задача должна реализовывать запись данных в файл file.data.

* Должен быть **модельный класс**, представляющий данные.
* Должен быть **класс записи**, который записывает модельные объекты в файл.
* **Нельзя использовать сериализацию!**

**1. Запись списка пользователей в текстовый файл**

**Классы:**

1. User – модель пользователя с полями Name, Age, Email.
2. UserFileWriter – записывает объекты User в file.data.

**Методы UserFileWriter:**

* WriteUsers(List<User> users) – записывает список пользователей в файл, каждую строку в формате:

graphql

КопироватьРедактировать

Иван,25,ivan@example.com

Ольга,30,olga@example.com

**2. Запись списка товаров в двоичный файл**

**Классы:**

1. Product – модель товара (Name, Price, Category).
2. ProductFileWriter – записывает объекты Product в file.data.

**Методы ProductFileWriter:**

* WriteProducts(List<Product> products) – записывает товары в бинарном формате (BinaryWriter).

**3. Проверка существования файла перед записью**

**Классы:**

1. LogEntry – модель записи (Date, Message).
2. LogFileWriter – записывает LogEntry в file.data.

**Методы LogFileWriter:**

* AppendLogEntry(LogEntry entry) – добавляет запись в файл, если он существует, иначе создает.

**4. Запись данных в файл с разделителем (|)**

**Классы:**

1. Employee – модель (Name, Department, Salary).
2. EmployeeFileWriter – записывает объекты Employee в file.data.

**Методы EmployeeFileWriter:**

* WriteEmployees(List<Employee> employees, char separator) – записывает данные с указанным разделителем.

**5. Запись данных в несколько файлов в зависимости от возраста**

**Классы:**

1. Person – модель (Name, Age).
2. PersonFileSplitter – записывает людей в разные файлы (adults.data, minors.data).

**Методы PersonFileSplitter:**

* WritePeople(List<Person> people) – сортирует и записывает людей в файлы по возрасту.

**6. Перезапись файла с новыми данными**

**Классы:**

1. Client – модель (Name, Balance).
2. ClientFileWriter – записывает объекты Client в file.data.

**Методы ClientFileWriter:**

* OverwriteClients(List<Client> clients) – очищает файл перед записью новых данных.

**7. Добавление новой записи без удаления старых**

**Классы:**

1. Transaction – модель (Id, Amount).
2. TransactionFileWriter – записывает Transaction в file.data.

**Методы TransactionFileWriter:**

* AppendTransaction(Transaction transaction) – добавляет запись без удаления предыдущих данных.

**8. Очистка файла перед записью новых данных**

**Классы:**

1. SensorData – модель (Timestamp, Value).
2. SensorDataWriter – записывает данные сенсора в file.data.

**Методы SensorDataWriter:**

* ClearAndWrite(List<SensorData> data) – очищает файл перед записью.

**9. Запись только уникальных записей (без дубликатов)**

**Классы:**

1. Customer – модель (Id, Name).
2. CustomerFileWriter – записывает Customer в file.data.

**Методы CustomerFileWriter:**

* WriteUniqueCustomers(List<Customer> customers) – записывает только уникальных клиентов.

**10. Проверка корректности записи, открыв файл другим методом**

**Классы:**

1. Order – модель (Id, ProductName, Quantity).
2. OrderFileWriter – записывает Order в file.data.
3. OrderFileReader – проверяет корректность данных.

**Методы:**

* WriteOrders(List<Order> orders) – записывает заказы.
* VerifyFile() – проверяет, корректно ли записаны данные.

**11. Запись данных с заголовком (название колонок)**

**Классы:**

1. Employee – модель (Name, Position, Salary).
2. EmployeeFileWriter – записывает объекты Employee в file.data.

**Методы EmployeeFileWriter:**

* WriteEmployeesWithHeader(List<Employee> employees) – записывает заголовок перед данными.

**12. Защита файла от перезаписи после записи данных**

**Классы:**

1. Document – модель (Title, Content).
2. DocumentFileWriter – записывает Document в file.data.

**Методы DocumentFileWriter:**

* WriteAndProtect(Document document) – записывает и делает файл ReadOnly.

**13. Запись объектов в JSON-формате (без сериализации)**

**Классы:**

1. Book – модель (Title, Author, Year).
2. BookJsonWriter – записывает книги в file.data.

**Методы BookJsonWriter:**

* WriteBooksAsJson(List<Book> books) – записывает данные в JSON-формате вручную (без сериализации).

**14. Сортировка данных перед записью**

**Классы:**

1. Student – модель (Name, Score).
2. StudentFileWriter – записывает Student в file.data.

**Методы StudentFileWriter:**

* WriteSortedStudents(List<Student> students) – сортирует студентов по баллам перед записью.

**15. Запись только объектов, удовлетворяющих условию**

**Классы:**

1. Car – модель (Brand, Year).
2. CarFileWriter – записывает Car в file.data.

**Методы CarFileWriter:**

* WriteFilteredCars(List<Car> cars, int minYear) – записывает только автомобили, новее minYear.

**Задание 3: Чтение данных из файла "file.data" и их обработка (сложность 4)**

**Описание**

В этом задании необходимо реализовать:

* **Класс для чтения данных из файла** file.data (заполненного в Задании 2).
* **Класс для преобразования данных в коллекцию объектов**.
* **Класс для поиска, фильтрации и сортировки объектов**.
* **Нельзя использовать десериализацию!**

**1. Чтение списка пользователей и поиск по email**

**Классы:**

1. User – модель (Name, Age, Email).
2. UserFileReader – читает пользователей из file.data.
3. UserProcessor – поиск по Email.

**Методы:**

* ReadUsers() – загружает пользователей из файла в коллекцию.
* FindUserByEmail(string email) – ищет пользователя по email.

**2. Чтение списка товаров и сортировка по цене**

**Классы:**

1. Product – модель (Name, Price, Category).
2. ProductFileReader – читает товары из file.data.
3. ProductProcessor – сортировка.

**Методы:**

* ReadProducts() – загружает товары в список.
* SortByPrice(bool ascending) – сортирует товары по цене.

**3. Фильтрация логов по дате**

**Классы:**

1. LogEntry – модель (Date, Message).
2. LogFileReader – читает логи из file.data.
3. LogProcessor – фильтрация.

**Методы:**

* ReadLogs() – загружает логи.
* FilterLogsByDate(DateTime fromDate, DateTime toDate) – фильтрует по диапазону дат.

**4. Поиск сотрудников по отделу**

**Классы:**

1. Employee – модель (Name, Department, Salary).
2. EmployeeFileReader – читает сотрудников.
3. EmployeeProcessor – поиск по отделу.

**Методы:**

* ReadEmployees() – загружает сотрудников.
* FindByDepartment(string department) – ищет сотрудников по отделу.

**5. Группировка людей по возрасту**

**Классы:**

1. Person – модель (Name, Age).
2. PersonFileReader – читает данные.
3. PersonProcessor – группировка.

**Методы:**

* ReadPeople() – загружает людей.
* GroupByAge() – группирует по возрасту (моложе 18, от 18 до 40, старше 40).

**6. Поиск клиентов с отрицательным балансом**

**Классы:**

1. Client – модель (Name, Balance).
2. ClientFileReader – читает клиентов.
3. ClientProcessor – поиск должников.

**Методы:**

* ReadClients() – загружает клиентов.
* FindDebtors() – ищет клиентов с балансом < 0.

**7. Фильтрация транзакций по сумме**

**Классы:**

1. Transaction – модель (Id, Amount).
2. TransactionFileReader – читает транзакции.
3. TransactionProcessor – фильтрация.

**Методы:**

* ReadTransactions() – загружает транзакции.
* FilterByAmount(decimal minAmount) – фильтрует по минимальной сумме.

**8. Определение средних показателей датчиков**

**Классы:**

1. SensorData – модель (Timestamp, Value).
2. SensorFileReader – читает данные.
3. SensorProcessor – анализ данных.

**Методы:**

* ReadSensorData() – загружает данные.
* CalculateAverageValue() – вычисляет среднее значение.

**9. Проверка уникальности клиентов**

**Классы:**

1. Customer – модель (Id, Name).
2. CustomerFileReader – читает клиентов.
3. CustomerProcessor – поиск дубликатов.

**Методы:**

* ReadCustomers() – загружает клиентов.
* FindDuplicates() – ищет дубликаты по Id.

**10. Определение самых популярных товаров**

**Классы:**

1. Order – модель (Id, ProductName, Quantity).
2. OrderFileReader – читает заказы.
3. OrderProcessor – определяет популярные товары.

**Методы:**

* ReadOrders() – загружает заказы.
* GetMostPopularProducts(int topN) – находит топ-N товаров.

**11. Чтение сотрудников и поиск по зарплате**

**Классы:**

1. Employee – модель (Name, Position, Salary).
2. EmployeeFileReader – читает сотрудников.
3. EmployeeProcessor – фильтрация по зарплате.

**Методы:**

* ReadEmployees() – загружает сотрудников.
* FilterBySalary(decimal minSalary) – фильтрует сотрудников.

**12. Фильтрация автомобилей по году выпуска**

**Классы:**

1. Car – модель (Brand, Year).
2. CarFileReader – читает автомобили.
3. CarProcessor – фильтрует машины.

**Методы:**

* ReadCars() – загружает автомобили.
* FilterByYear(int minYear) – оставляет только новые авто.

**13. Определение самой прибыльной сделки**

**Классы:**

1. Deal – модель (Id, Revenue).
2. DealFileReader – читает сделки.
3. DealProcessor – анализирует прибыль.

**Методы:**

* ReadDeals() – загружает сделки.
* FindMostProfitableDeal() – находит самую выгодную.

**14. Подсчет количества записей по категориям**

**Классы:**

1. CategoryItem – модель (Category, Name).
2. CategoryFileReader – читает данные.
3. CategoryProcessor – подсчитывает записи.

**Методы:**

* ReadItems() – загружает записи.
* CountByCategory() – считает количество записей по категориям.

**15. Анализ студентов по среднему баллу**

**Классы:**

1. Student – модель (Name, Score).
2. StudentFileReader – читает студентов.
3. StudentProcessor – анализирует баллы.

**Методы:**

* ReadStudents() – загружает студентов.
* AnalyzePerformance() – рассчитывает средний балл и определяет лучших.

**Задание 4: FileSystemWatcher – отслеживание изменений в файлах (сложность 3)**

**Описание**

Необходимо реализовать класс, который отслеживает изменения в указанной папке с помощью FileSystemWatcher и выполняет определенные действия.

Каждая задача должна включать:

* **Класс FileWatcher** – основной класс, который использует FileSystemWatcher.
* **Методы для обработки событий** (Created, Deleted, Changed, Renamed).
* **Дополнительные действия** (логирование, копирование, анализ и др.).

**1. Отслеживание создания и удаления файлов**

* Реализовать отслеживание создания и удаления файлов в указанной папке.
* При создании файла – выводить в консоль сообщение "Файл {name} создан".
* При удалении файла – выводить сообщение "Файл {name} удален".

**2. Автоматическое создание резервной копии при изменении файла**

* При изменении файла копировать его в папку backup.
* Название копии должно включать дату и время (file\_YYYYMMDD\_HHMMSS.bak).

**3. Логирование всех изменений в файлах**

* Реализовать логирование всех изменений (Created, Deleted, Changed, Renamed).
* Логировать события в файл log.txt с указанием времени и типа события.

**4. Реакция на переименование файла**

* При переименовании файла записывать старое и новое имя в log.txt.
* Выводить в консоль сообщение: "Файл {oldName} переименован в {newName}".

**5. Уведомление о новых файлах по email**

* При создании нового файла отправлять email-уведомление (в консоль вместо реального email).

**6. Контроль изменений конфигурационных файлов**

* Отслеживать изменения в .config файлах.
* При изменении – выводить предупреждение: "Конфигурация изменена!".

**7. Автоматическое удаление временных файлов**

* Если в папке появляется файл с расширением .tmp, он автоматически удаляется.

**8. Перемещение старых файлов в архив**

* Если файл не изменялся более 30 дней, перемещать его в папку archive.

**9. Подсчет общего количества файлов в папке**

* При любом изменении (создание, удаление) выводить количество файлов в папке.

**10. Поиск файлов с определенным расширением**

* Если добавляется файл .txt, .csv, .xml, записывать его имя в files\_list.txt.

**11. Контроль дубликатов**

* Если в папке появляется файл с таким же именем, как существующий, добавлять \_copy к названию.

**12. Реализация блокировки удаления важных файлов**

* Защитить файлы important.txt и config.json от удаления (восстанавливать при удалении).

**13. Уведомление при изменении больших файлов**

* Если изменился файл размером >100 MB, выводить предупреждение "Файл {name} слишком большой!".

**14. Интеграция с базой данных**

* При изменении файлов записывать информацию о событии в файл CSV.

**15. Синхронизация изменений в двух папках**

* Если файл изменяется в FolderA, автоматически копировать обновление в FolderB.