Консалтинговая фирма использует специализированное автоматизированное рабочее место (АРМ) для работы своих сотрудников. Данный АРМ также хранит всю информацию о клиентах фирмы. Карточка клиента имеет специальный формат и специалисты фирмы уже привыкли работать с этим форматом. Данный АРМ хранит все данные локально и  не использует никаких СУБД. На общем сетевом диске создана папка, в которой хранятся файлы вида client.out с информацией о клиенте. При интеграции новой CRM у владельцев фирмы возникла необходимость в создании плагина для CRM, который бы осуществил импорт данных  в СРМ с сохранением форматов данных, удобных для сотрудников.

Старый формат данных был описан публичным, нестатическим классом **Clients** c приватными полями доступными для чтения и записи:

* name - полное наименование организации с указанием организационно-правовой формы
* **contactPerson** - контактное лицо, объект класса **ContactPerson**
* requisites - список реквизитов (известная информация о клиенте, например ИНН, КПП и тд), список объектов класса Requisites, у каждого клиента свой список заполненных реквизитов

Список реквизитов описан публичным, нестатическим классом Requisites с приватными полями доступными для чтения и записи:

* name - наименование реквизита
* value - значение реквизита

**ContactPerson** - также публичный, нестатический класс, который описывает контрагентов консалтинговой фирмы, в частности клиентов. Данный класс содержит приватные поля, доступные для чтения и записи:

* name - имя человека, через которого происходит коммуникация , имеет строковый тип данных, может быть в любом формате, не только ФИО
* email - контактный адрес электронной почты, имеет строковый тип данных, может быть пустым
* phone  - контактный телефон, хранится без форматирования, имеет строковый тип данных, может быть пустым

Все вышеуказанные классы имеют стандартные публичные конструкторы.

Проведя небольшое исследование технические сотрудники фирмы обнаружили, что файлы карточек клиентов не зашифрованы, а представляют собой сохраненные сериализованные объекты класса **Clients**.

Таким образом все указанные выше классы реализуют интерфейс **Serializable**

Для осуществления импорта тебе потребуется десериализовать данные, хранящиеся в архиве Clients.zip и осуществить их простую проверку.

В данном задании тебе необходимо реализовать публичный, нестатический класс **ImportClients**, который осуществляет импорт данных, а также все вспомогательные классы, которые необходимы для описания форматов, используемых в карточке клиента.

Класс **ImportClients** содержит следующие методы:

* публичный статический метод **cardsToClients()** - на входе получает путь к папке с данными клиентов (папка, в которую ты распакуешь архив Clients.zip ), на выходе формирует список объектов класса Clients, считанных из файлов с расширением **\*.out**  в указанной папке (один файл соответствует одному клиенту), при возникновении исключительных ситуаций возвращает пустой список объектов класса Clients
* публичный статический метод **checkImport()** - на входе получает список объектов типа **Clients**, на выходе выводит в консоль информации о клиенте, с самым коротким названием  в формате:

“<Наименование фирмы>, <Имя контактного лица>, <ИНН>, <ОКВЭД>”

При получении пустого списка выводит в консоль сообщение “import failed”

Для успешного выполнения задания тебе необходимо реализовать все указанные классы и их методы,а также предусмотреть их безотказную работу в случае загрузки неправильных или отсутствующих данных по указанному пути.

Под безотказной работой понимается корректное завершение программы без генерации исключений.