Information System

Краткое описание

В данной задаче рассматривается структура данных, описывающая некую реальную систему, объект или процесс. На основе заданной структуры данных реализовать справочную систему, отвечающую за манипулирование данными. Данные справочной системы хранятся на сервере, а доступ к ним осуществляется удаленно.

Детальное описание

Данные разделены на два (или более) типа, связанные между собой. Обработка каждого типа данных осуществляется в отдельном классе (Model class). После обработки информация сохраняется на диск в определенном формате. Каждый класс должен уметь создавать, удалять и модифицировать свой тип данных, при этом корректно взаимодействуя с классом, занимающимся обработкой зависимых типов. Управление справочной системой осуществляется с удаленной машины, на которой установлен клиент с графическим интерфейсом, реализованный с помощью Swing компонентов. Реализация конкретной справочной системы строится на модели клиент-сервер. В этом случае сервер полностью отвечает за обработку данных, а клиентская часть только за их отображение и обработку команд пользователя.

Требования

Звездочками * отмечены обязательные требования. Набор обязательных требований может быть изменен или дополнен куратором группы.

1. * Реализация Справочной системы должна соответствовать модели клиент-сервер с использованием сокетов.
2. * Возможность подсоединения к серверу более одного клиента.
3. * Графический интерфейс должен состоять из таблицы, отображающей данные, и функциональными кнопками.
4. * Наличие функций добавления, удаления, изменения и просмотра данных.
5. Реализация поиска данных в соответствии с некоторым шаблоном, который вводится пользователем (в окошке графического интерфейса). Шаблон включает в себя все разрешенные символы с точки зрения хранимых данных и символы заменяющие один и несколько любых символов (* и ?).
6. * Вывод данных в табличку в отсортированном виде с помощью использования компаратора (java.util.Comparator).
7. * Сохранение загрузка данных через Serialization/XML, а также передача их между сервером и клиентом.
8. Возможность добавления всех данных с одного сервера на другой с проверкой на наличие дубликатов, т.е. не должно быть абсолютно одинаковых данных.
9. * Программный код должен удовлетворять Java Code Conventions и быть снабженным JavaDoc.
10. * Логирование в коде организовать с помощью библиотеки Log4j.
11. * Для сборки приложения использовать Maven.
12. * Дизайн приложения. Дизайн приложения должен создаваться **до написания кода** и представляться группой куратору для обсуждения и утверждения. Дизайн должен состоять из:
    a. Диаграмма классов (Class diagram) приложения
    b. Диаграмма прецедентов (Use Case диаграмма)

Модификации задания

В информационной системе могут храниться следующие наборы данных. Окончательный выбор предметной области должен быть согласован с куратором.

1. Группы и студенты: Студент (ФИО, группа, дата зачисления), Группа (Номер, Факультет).
2. Библиотека: Экземпляр книги (инвентарный номер, книга, выдана или нет), Книга (авторы, название, год издания, число страниц).
3. Отдел кадров: Сотрудник (ФИО, отдел, телефон, зарплата), Отдел (название, начальник).
4. Отдел поставок: Сырье (название, поставщик, цена), Поставщик (название, расчетный счет, ФИО контактного лица).
5. Отдел продаж: Заказ (номер, заказчик, дата, сумма заказа), Заказчик (название, телефон, адрес).
6. Ресторан: Блюдо (название, категория, цена), Категория блюд (название).
7. Анализ публикаций: Публикация (название, тип, источник, дата), Источник (название, город, телефон).
8. Авиарейсы: Рейс (Номер рейса, аэробус, маршрут, время вылета, путевое время), Маршрут (Пункт вылета, пункт прибытия).
9. Расписание электричек: Электропоезд (Номер состава, маршрут, время отправления, путевое время), Маршрут (Начальная станция, конечная станция).
10. Собственный вариант, согласованный с куратором.

Дополнительная информация

Изучаемые темы

Основы ООП, синтаксис Java, Collections, I/O, XML, Serialization, Sockets, Threads.

Справочные материалы:

1. http://logging.apache.org/log4j/2.x/
2. http://maven.apache.org/guides/getting-started/index.html
3. http://www.info-system.ru/designing/methodology/uml/theory/class_diagram_theory.html
4. http://www.info-system.ru/designing/methodology/uml/theory/use_case_diagram_theory.html