БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И РОБОТОТЕХНИКИ

ИНСТРУКЦИЯ

для выполнения лабораторных работ по дисциплине «КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Учебный модуль:
«Фундаментальные основы разработки программного обеспечения»
(Программная инженерия)

Разработал старший преподаватель кафедры «Программное обеспечение информационных систем и технологий», Станкевич Сергей Николаевич.

Минск, 2023 год.

КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Цели освоения дисциплины

Изучение и освоение подходов и способов конструирования современного программного обеспечения, получение навыков создания и тестирования программ с использованием современных языков, методик, технологий и инструментальных средств разработки программного обеспечения.

Средства выполнения

1. ***Система контроля версий*** Git и т.п., в также удаленный репозиторий GitHub и т.п.
2. Объектно-ориентированный, *компилируемый* ***язык программирования*** высокого уровня, например: C#, Java, C++.
3. ***Интегрированные системы разработки*** программного обеспечения, например: Visual Studio, IntelliJ IDEA, Eclipse и т.п.
4. Инструменты для разработки ***UML-диаграмм*** и генерации исходного кода с помощью этих диаграмм, например: Диаграммы классов (как плагин Visual Studio), Umple (как плагин Eclipse) и т.п.

Все средства выполнения выбираются студентом самостоятельно по его усмотрению.

Требования по защите лабораторных работ

1. Наличие работающей программы и практического выполнения задания.
2. Оформленный документированный отчет.
3. Личная защита лабораторной работы.

Требования для документированного отчета

1. Титульный лист.
2. Перечень *требований* и *заданий* к лабораторной работе.
3. Описание проделанной работы, скриншоты выполнения и результатов, ключевые фрагменты кодов программы.
4. Ответы на *контрольные вопросы*.
5. *Глоссарий* по ключевым определениям и понятиям.
6. *Ссылки* на использованные источники и ресурсы.

ниже

Общее содержание инструкции

Оглавление

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Глобальные, общие требования к разрабатываемому ПО

1. При проектировании и разработке системы необходимо полностью использовать своё объектно-ориентированное **воображение** и по максимум использовать возможности, которые предоставляет язык программирования для реализации ООП-методологии.
2. Разработайте ***консольное*** приложение.
3. Для взаимодействия с пользователем в программе должен быть **дружелюбный и интуитивно понятный интерфейс.**
4. Программа должна быть **интерактивна**. В приложении разработайте **меню** выбора действий пользователя. Должен быть предусмотрен **корректный выход из программы**.
5. В соответствующих компонентах бизнес-логики необходимо предусмотреть «**защиту от дурака**». Предусмотрите проверку корректного ввода данных.
6. Программа должна обязательно быть снабжена ***описанием***, в котором необходимо указать краткое предназначение программы, *номер лабораторной работы и её название, версию программы, ФИО разработчиков, название бригады (если есть), номер группы и дату разработки*.
7. Интерфейс программы и комментарии желательно, должны быть на английском языке.
8. Необходимо спроектировать и реализовать ***UML-диаграмму взаимодействия классов и объектов*** разрабатываемой программной системы с отображением всех связей (отношений) между классами и объектами.
9. Основные классы системы должны быть самодостаточными, т.е. не зависеть, к примеру, от консоли! Любые типы отношений между классами должны применяться обосновано и лишь тогда, когда это имеет смысл.
10. При выполнении задания необходимо по максимуму пытаться разрабатывать ***универсальный***, ***масштабируемый***, легко ***поддерживаемый*** и ***читаемый*** код.
11. Рекомендуется придерживаться принципов SOLID (GRASP), особенно ***принципа единственной ответственности*** (Single Responsibility Principle, SRP): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода. классы рекомендуется проектироваться и реализовываться таким образом, чтобы они были слабо зависимы от других классов.
12. При разработке программ придерживайтесь ***соглашений по написанию*** кода на языке программирования разрабатываемого проекта.
13. При разработке используйте ***архитектурный шаблон проектирования,*** рекомендуется ***Model-View-Controller*** (MVC).
14. Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты.
15. Использовать ***системы контроля версий*** (за базовую систему принять **Git**), и в процессе разработки приложения наиболее полно использовать их возможности и руководствоваться их ***стратегиями***.
16. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями(?). **Лучше делайте коммиты в Git**.

above

ЛАБОРАТОРНАЯ РАБОТА #11
ВЫБОР СРЕДСТВ ВЫПОЛНЕНИЯ И НАЧАЛО ПРОЕКТА

Цель

Научиться *анализировать* предметную область и с помощью абстракции выделять существенные детали, на базе которых в дальнейшем проектируются классы и объекты будущей программной системы согласно методологии ООП, а также практически закрепить данные навыки при решении соответствующих задач (бизнес проблем).

Основные понятия: предметная область, бизнес-логика, классы, абстракция, интерфейсы.

Требования

Разработка приложения должна проводиться с учетом «глобальных» требований.

Основное задание

Нужно разработать приложение с использованием методологии ООП. Для чего необходимо подобрать самостоятельно соответствующую ***проблемную*** (предметную/доменную) ***область***, которая базируется на объектах и событиях реального мира (примеры соответствующих предметных областей приведены ниже в Приложение А

Примеры предметных областей).

Прежде чем начать *проектирование* и *конструирование* приложения исследуйте выбранную предметную область и определите основные *требования* к программе. Для этого используйте возможности интернета и посетите два, три сайта по соответствующей теме. Скриншоты сайтов и ссылки на них представить в отчете.

Определите основные сущности предметной области и поведение программы. Кратко опишите это в отчете.

В процессе разработки требования к программе могут меняться и дополняться. Версии отчета по требованиям к программе сохраняйте в репозиториях систем контроля версий.

Спроектировать классы (собственные пользовательские типы данных) в языке программирования для программного представления данных объектов и основной логики будущей программной системы.

Система должна решать, как минимум, два полезных действия и иметь дополнительно следующие опции:

- не менее **3 разнообразных классов** предметной области;
- не менее **5 атрибутов** (состояния) и 3-х методов (поведения) в **классе-сущности**;
- не менее **3 методов**, которые реализуют **бизнес-логику** программы, в соответствующих *функциональных* классах;
- хранить **глобальные характеристики** системы или **характеристики уровня *отдельных классов***.

На базе спроектированной программной системы реализовать программу и продемонстрировать её работоспособность.

Необходимо спроектировать и реализовать ***UML-диаграмму взаимодействия классов и объектов*** разрабатываемой программной системы с отображением всех связей (отношений) между классами и объектами.

При разработке проекта **должны быть** сделаны **соответствующие коммиты в Git.**

Реферат к лабораторной работе #11

1. Пояснить почему были выбраны те или иные средства выполнения работы (язык программирования, интегрированная система разработки ПО, система контроля версий). Указать их преимущества и недостатки. Провести их сравнительный анализ с аналогичными средствами.
2. Опишите **соглашении о написании кода** выбранного языка программирования.

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #21 Объектно-Ориентированное Программирование на языке Java. Абстракция. Автор – Иванченко В.В.

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #12
ИНИЦИАЛИЗАЦИЯ СОСТОЯНИЯ ОБЪЕКТА И СТАТИЧЕСКИЕ КОМПОНЕНТЫ КЛАССА

Цель

Научиться использовать соответствующие средства объектно-ориентированного языка программирования для первоначальной *инициализации состояния объекта*, а также изучить истинное предназначение *статических компонентов класса*.

Основные понятия: объект как экземпляр класса, конструктор класса, инициализация объекта, статические компоненты класса.

Требования

Учитывайте все требования, указанные в предыдущих лабораторных работах.

Основное задание

Необходимо в проект, который был спроектирован и разработан в предыдущей лабораторной работе, внести следующие дополнения:

- для грамотной инициализации состояния объектов соответствующей предметной области добавить всевозможные средства инициализации, которые предоставляет язык Java (блоки инициализации, конструктор по умолчанию, конструкторы с параметрами, конструктор-копирования и т.д.);
- проанализировав соответствующую предметную область добавить в проект статические компоненты класса и возможность их первоначальной инициализации с помощью средств, который предоставляет язык программирования.

Дополнительно необходимо проанализировать стадии и способы инициализации как состояния объектов, так и состояния соответствующих объектов классов (объектов класса Class), а также их очередность вызова JVM.

Привести анализ результатов и соответствующие выводы в отчёте.

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #22 Инициализация состояния объекта. Статические компоненты класса. Автор – Иванченко В.В.

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #13
ИНКАПСУЛЯЦИЯ

Цель

Углубить фундаментальные знания в использовании методологии ООП, а также научиться практически применять инкапсуляцию с использованием средств, которые предоставляет объектно-ориентированный язык программирования.

Основные понятия: инкапсуляция как парадигма ООП, средства и методы доступа к состоянию объекта, порождающие шаблоны проектирования (паттерны-креаторы).

Требования

1. Необходимо ***скорректировать UML-диаграмму*** взаимодействия классов и объектов программной системы с учётом вносимых дополнений и изменений.
2. Необходимо скрыть реализацию компонентов и структур хранения данных с помощью грамотного применения инкапсуляции.
3. Каждый пользовательский тип должен иметь адекватное осмысленное имя и информативный состав (соответствующие конструкторы: по умолчанию, с параметрами, конструктор-копирования; get- и set-методы для доступа к состоянию объекта; корректно **переопределённые** методы базового класса Object: toString(), equals(), hashCode() и др.).

Основное задание

Необходимо в проект, который был спроектирован и разработан в предыдущей лабораторной работе, внести ***следующие изменения и дополнения***:

- скрыть реализацию всех компонентов и структур данных проекта, т.е. **инкапсулировать** все поля классов и методы, которые предназначены для внутреннего использования, с использованием модификаторов доступа языка Java, и предоставить только интерфейсную часть для внешнего взаимодействия;
- ввести, где это необходимо, высокоуровневые **объекты-контейнеры**, которые инкапсулируют структуру хранения множества объектов предметной области;
- убрать из класса-контроллера код по инициализации объектов предметной области и ввести соответствующие программные компоненты, которые и будут предназначены для создания и инициализации объектов предметной области, т.е. использовать компоненты в виде шаблонов проектирования: ***фабрик*** или ***строителей*** («креаторов»).

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #23 ООП в Java. Инкапсуляция. Автор – Иванченко В.В.

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #14
КОЛЛЕКЦИИ И КЛАССЫ-КОНТЕЙНЕРЫ

Цель

Ознакомиться с содержимым и архитектурой библиотеки классов-контейнеров, коллекций, а также получить практические навыки использования интерфейсов и классов данного фреймворка для решения задач, связанных с хранением и обработкой набора данных.

Основные понятия: коллекции, классы-контейнеры, сортировка объектов, контейнеры (например, в Java – Java Collections Framework, JCF), поведенческие шаблоны проектирования.

Требования

Учитывайте все требования, указанные в предыдущих лабораторных работах.

Основное задание

Модифицировать программу, разработанную в предыдущей лабораторной работе, следующим образом:

- во всех классах заменить работу с контейнерами пользовательских типов и(или) массивами на работу с соответствующими контейнерами из библиотеки;

- используя поведенческий шаблон проектирования, «Стратегия» (The Strategy Pattern) внедрить логику по поиску соответствующих данных в предметную область,
- добавить различные *типы сортировок* (сортировки по разным критериям сущностей из предметной области), создать сортировку бизнес-объектов по нескольким критериям с использованием интерфейса Comparable для указания естественной сортировки и Comparator для реализации остальной логики сортировки бизнес-объектов (пример реализации шаблона «Стратегия»);
- выбор объекта, который содержит логику сортировки, должен выбираться из контейнера типа Map.
- добавить возможность поиска бизнес-объектов по разным параметрам;
- добавить возможность использования итераторов в пользовательских контейнерных классах (т.е. необходимо описать класс-контейнер таким образом, чтобы объект данного класса можно было использовать с модифицированным в JDK5.0 циклом for).

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:

ЛАБОРАТОРНАЯ РАБОТА #27 Использование контейнеров в Java. Java Collections Framework, JCF. Автор – Иванченко В.В.

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #15
НАСЛЕДОВАНИЕ

Цель

Углубить фундаментальные знания в использовании методологии ООП, научившись применять на практике инструменты объектно-ориентированного языка программирования для повторного использования кода в виде ассоциации, наследования, агрегации, композиции и делегирования.

Основные понятия: наследование как парадигма ООП, повторное использование кода, наследование базовых конструкторов, ассоциация, наследование, агрегация, композиция и делегирование.

Требования

Обеспечить масштабирование программы средствами наследования.

Создать классы с **иерархической структурой наследования**. Произвести, где это необходимо, классификацию типов, *например*, в предыдущей лабораторной работе была только сущность автомобиль/автотранспорт, а теперь должна быть иерархия автотранспорта с соответствующими характеристиками: легковой автомобиль (седан, универсал, хэтчбэк, …), грузовой автомобиль (фура, самосвал, бетономешалка, …), пассажирский автомобиль (автобус, микроавтобус, минивэн, …) и т.д., никто никого не ограничивает в фантазиях.

Использовать наследование базовых конструкторов, ассоциация, наследование, агрегация, композиция и делегирование.

Основное задание

Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

- **классы**, описывающие объекты соответствующей предметной области (бизнес-объекты), должны быть **сведены в иерархическую структуру,** создать минимум один класс предметной области **с иерархией в трех уровнях,** при этом применить наследование конструкторов базовых классов;
- логика системы должна быть реализована внутри соответствующих функциональных классов;
- логика системы и большинство других компонентов должны зависеть преимущественно только от абстракции, а не от реализаций;
- необходимо дополнительно для безопасности выполнения кода добавить по возможности в методы бизнес-логики проверку входящих объектов на соответствие типа, с которым должна взаимодействовать данная логика.

Модифицируйте UML-диаграмму классов с учётом, внесённых изменения архитектуры своего приложения.

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Рисунок 1 - Пример иерархии классов

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #24 Наследование. Повторное использование кода. Автор – Иванченко В.В.

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #16
ПОЛИМОРФИЗМ

Цель

Углубить фундаментальные знания в использовании методологии ООП, научившись применять на практике динамический полиморфизм с использованием средств, которые предоставляет объектно-ориентированный язык программирования.

Основные понятия: полиморфизм как парадигма ООП, поведенческий шаблон проектирования, перегрузка методов.

Требования

Учитывайте все требования, указанные в предыдущих лабораторных работах.

Основное задание

Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

- наделить сущности (бизнес-объекты), которыми манипулирует бизнес-логика, соответствующим полиморфным поведением таким образом, чтобы не пришлось переписывать саму бизнес-логику;
- на уровне всей программы добавить глобальные характеристики, на базе которых реализуется бизнес-логика (на пример, механизм отслеживания количества создаваемых объектов предметной области, которыми манипулирует система при выполнении основных расчётов программной системы).

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:

ЛАБОРАТОРНАЯ РАБОТА #25 ООП в Java. Полиморфизм. Автор – Иванченко В.В.

«Easy things should be easy and hard things should be possible»

Best of LUCK with it, and remember to HAVE FUN while you're learning :)

РАЗДЕЛ 1. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Глобальные, общие требования к разрабатываемому ПО

1. Перестройте ранее разработанное консольное приложение на приложение с оконным интерфейсом. При этом используйте современные технологии: WPF (Microsoft), или аналогичную технологию Java или MacOC.
2. Необходимо проверить все ***тестовые случаи*** работы основной бизнес-логики программы.

выше

ЛАБОРАТОРНАЯ РАБОТА #21
МЕТОДОЛОГИЯ РАЗРАБОТКИ TEST DRIVEN DEVELOPMENT

Цель

Освоить методологию разработки ПО через тестирование (TDD), а также изучить соответствующие тестовые фреймворки для модульного тестирования (например, в Java это jUnit и TestNG) и закрепить их практически.

Основные понятия: модульное тестирование, Unit-тесты.

Требования

Весь код модели системы должен быть покрыт модульными тестами, подтверждающие качество и работоспособность ядра системы. Необходимо проверить все тестовые случаи. Для реализации модульных тестов рекомендуется использовать jUnit фреймворк.

Основное задание

Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

- создать отдельный тестовый подпроект в рамках текущего программного решения;
- максимально покрыть основную бизнес-логику модульными тестами;
- реализовать тестовые планы, которые будут включать как методы, которые будут вызываться перед и после каждого тестового случая, так и методы, которые должны вызываться на уровне всего класса;

- предусмотреть также тестирование методов на исключительные ситуации и временные интервалы.

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #26 Методология разработки Test Driven Development. Использование тестовых фреймворков jUnit и TestNG для проведения модульного тестирования. Автор – Иванченко В.В.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #22
БИБЛИОТЕКИ ПОЛЬЗОВАТЕЛЬСКИЕ

Цель

Изучить и закрепить на практике создание и использование *статически* и *динамически* подключаемых *пользовательских* библиотек в среде вашей операционной системы.

Основные понятия: статические и динамические библиотеки, (например, в Microsoft это DLL), ад DLL, статическое и динамическое подключение динамически подключаемых библиотек.

Требования

Создать программные модули приложения в виде *статических* и *динамических* пользовательских *библиотек*. Выполнение данной работы можно совместить с выполнением последующих работ.

Основное задание

Необходимо произвести рефакторинг программной системы следующим образом:

- Вынесите код функций программы в отдельную *статическую* библиотеку. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат?
- Вынесите код функций программы в отдельную *динамическую* библиотеку со *статическим* вызовом. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.
- Вынесите код функций программы в отдельную *динамическую* библиотеку с *динамическим* вызовом. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.
- Выполнение второго и третьего задания можно совместить.

- Сравните результаты выполняемых заданий, сделайте вывод.

При разработке проекта **должны быть** сделаны соответствующие **коммиты в Git.**

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #26. Автор – Станкевич С.Н.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #23
ПОТОКИ ВВОДА-ВЫВОДА

Цель

Познакомиться с основами ввода-вывода, структурным шаблоном проектирования «Декоратор», а также практически закрепить данные знания на примере разработки интерактивного приложения.

Основные понятия: библиотека потоков ввода-вывода, спроектированная и реализованная на базе паттерна «Декоратор», структурные шаблоны проектирования, паттерн «Декоратор».

Требования

Для построения необходимых объектов с целью создания бизнес-объектов программной системы и объектов-принтеров рекомендуется воспользоваться шаблонами Абстрактная фабрика (The Abstract Factory Pattern), Фабричный метод (The Factory Method Pattern) или Строитель (The Builder Pattern).

Основное задание

В программную систему, разработанную на предшествующем этапе, необходимо внести следующие, расширяющие функционал возможности:

- возможность создавать объекты, которыми манипулирует бизнес логика, различными способами (с использованием генератора случайных чисел и «хардкода») и из различных источников данных (из текстового и бинарного файла) – для этого рекомендуется реализовать иерархию компонентов-создателей и сделать из взаимозаменяемыми в тестовом классе (контроллере);
- возможность выводить результирующие данные не только в консоль, но и ***сохранять их в файл (текстовый)*** – для этого рекомендуется также создать иерархию принтеров и сделать их взаимозаменяемыми в классе-контроллере.

По сделанным изменениям должны быть сделаны соответствующие коммиты в Git.

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #28 (part 1) Потоки ввода-вывода в Java. Структурный шаблон проектирования Декоратор. Автор – Иванченко В.В.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #25
ЖУРНАЛИРОВАНИЕ

Цель
Изучить технологию журналирования в Java на примере использования широко распространённой библиотеки де-факто Apache Log4j.

Основные понятия: журналирование, лог-файлы.

Требования

В качестве логгера необходимо использовать ставшую уже стандартом де-факто библиотеку логгирования Apache Log4j (ресурс для загрузки соответствующей библиотеки: http://logging.apache.org/log4j/).

Основное задание

- Необходимо произвести рефакторинг приложения предыдущей лабораторной работы таким образом, чтобы результирующие данные сохранялись в лог-файл, а также вести журналирование любых исключительных ситуации, которые могут возникнуть в процессе работы программы.
- Дополнительно необходимо расширить иерархию принтеров ещё одним компонентом, вывод которого будет базироваться на базе библиотеки log4j.

По сделанным изменениям должны быть сделаны соответствующие коммиты в Git.

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #28 (part 3) Использование библиотеки де-факто для журналирования Apache Log4j. Автор – Иванченко В.В.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #24
СЕРИАЛИЗАЦИЯ И ДЕСЕРИАЛИЗАЦИЯ ОБЪЕКТОВ

Цель

Изучить концепцию и аспекты сериализации и десериализации и закрепить знания и навыки на примере разработки интерактивных приложений.

Основные понятия: сериализация и десериализация объектов, типы сериализации … … … .

Требования

Учитывайте все требования, указанные в предыдущих лабораторных работах.

Основное задание

В программе, разработанной в предыдущей лабораторной работе, необходимо:

- добавить дополнительно два способа инициализации объектов с использованием стандартной и пользовательской («кастомной») сериализации;
- попробовать сделать рефакторинг слоя View таким образом, чтобы в контроллере объект-принтер возвращался с помощью объекта-креатора, метод которого брал бы уже готовые объекты из контейнера типа Map. Ключом в данном контейнере может быть тип принтера, а значением – конкретный объект-принтер.

По сделанным изменениям должны быть сделаны соответствующие ***коммиты в Git***.

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА #28 (part 2) Сериализация и десериализация объектов в Java. Автор – Иванченко В.В.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #26
ЛОКАЛИЗАЦИЯ ИНТЕРНАЦИОНАЛЬНЫХ ДАННЫХ

Цель

Изучить технологии *локализации* и *интернационализации* данных и применить полученные знания при проектировании и разработки интерактивных приложений.

Основные понятия: локализация и интернационализация данных.

Требования

1. Предусмотреть поддержку нескольких языков для локализации приложения.
2. Для реализации локализации данных используйте соответствующие библиотеки и классы предусмотренные в выбранной вами технологии программирования. Например, в Java рекомендуется использовать связку классов java.util.Locale и java.util.ResourceBundle, а также специальные properties-файлы для хранения соответствующих локализованных данных, а на C#, это реализации IStringLocalizer<T> и IStringLocalizerFactory пакетов NuGet Microsoft.Extensions.Localization и Microsoft.Extensions.Hosting и файлы ресурсов.

Основное задание

Необходимо адаптировать видимый интерфейс программы предыдущей лабораторной работы, а также отдельные её программные элементы (текст, даты, денежные единицы и т.п.) под соответствующую локаль пользователя.

Методические рекомендации:
ЛАБОРАТОРНАЯ РАБОТА # 29 (part 1) Локализация данных в Java. Автор – Иванченко В.В.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ЛАБОРАТОРНАЯ РАБОТА #27
ОБРАБОТКА ТЕКСТА И РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Цель

Закрепить навыки работы со строками на примере разработки интерактивных приложений для обработки текстовой информации.

Основные понятия: анализ текста, парсер, регулярные выражения.

Требование

1. Необходимо создать интерактивное приложение для анализа текста по программированию согласно *индивидуальному* заданию (необходимо выполнить минимум два задания).
2. Анализируемый текст в процессе работы программы должен быть представлен в виде соответствующих бизнес-объектов (сущностей). К примеру, объект-текст содержит в себе объекты, ассоциированы с блоками текста и программного кода. Блоки текста, в свою очередь, представляют собой объекты-контейнеры, в которых содержатся предложения. А сами предложения являются хранилищами простых слов, знаков препинаний и других частей предложения. Классы, на базе которых будут создаваться вышеописанные объекты, являются классами-сущностями. Они должны содержать в себе только определённую текстовую информации. У них не должно быть никаких методов бизнес-логики.

3. Анализ (разбор) текста должен осуществлять специальный утилитный класс-парсер. Именно данный класс должен возвращать бизнес-объект (объекты), с которыми и должна работать бизнес логика приложения.
4. Для разбиения тестовой информации необходимо использовать регулярные выражения.
5. Предусмотреть восстановление текста в первоначальный (оригинальный) вид из программных объектов-сущностей. Оригинальный текст и текст, который получается в результате восстановления, должны полностью совпадать. Это и будет показателем высокого качества разработанного класса-парсера.

Индивидуальное задание

1. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.
2. Вывести все предложения заданного текста в порядке возрастания количества слов в каждом из них.
3. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.
4. Во всех вопросительных предложениях текста найти и напечатать без повторений слова заданной длины.
5. В каждом предложении текста поменять местами первое слово с последним, не изменяя длины предложения.
6. Напечатать слова текста в алфавитном порядке по первой букве. Слова, начинающиеся с новой буквы, печатать с красной строки.
7. Рассортировать слова текста по возрастанию доли гласных букв (отношение количества гласных к общему количеству букв в слове).
8. Слова текста, начинающиеся с гласных букв, рассортировать в алфавитном порядке по первой согласной букве слова.
9. Все слова текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством расположить в алфавитном порядке.
10. Существует текст и список слов. Для каждого слова из заданного списка найти, сколько раз оно встречается в каждом предложении, и рассортировать слова по убыванию общего количества вхождений.
11. В каждом предложении текста исключить подстроку максимальной длины, начинающуюся и заканчивающуюся заданными символами.
12. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.
13. Отсортировать слова в тексте по убыванию количества вхождений заданного символа, а в случае равенства – по алфавиту.
14. В заданном тексте найти подстроку максимальной длины, являющуюся палиндромом, т.е. читающуюся слева направо и справа налево одинаково.
15. Преобразовать каждое слово в тексте, удалив из него все последующие вхождения первой буквы этого слова.
16. Преобразовать каждое слово в тексте, удалив из него все предыдущие вхождения последней буквы этого слова.
17. В некотором предложении текста слова заданной длины заменить указанной подстрокой, длина которой может не совпадать с длиной слова.

Методические рекомендации:

ЛАБОРАТОРНАЯ РАБОТА # 29 (part 2) Обработка текста в Java. Использование регулярных выражений. Автор – Иванченко В.В.

«Простые вещи должны быть простыми, а сложные вещи должны быть возможными»

«Easy things should be easy and hard things should be possible»

Best of LUCK with it, and remember to HAVE FUN while you're learning :)

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
выше

ГЛОССАРИЙ

Интегрированная система разработки программного обеспечения (Integrated Development Environment, IDE) – это интегрированная, **единая** среда разработки, которая используется разработчиками для создания различного программного обеспечения.

> IDE представляет собой комплекс из нескольких инструментов, а именно: текстового редактора, компилятора либо интерпретатора, средств автоматизации сборки и отладчика. Помимо этого, IDE может содержать инструменты для интеграции с системами управления версиями и другие полезные утилиты.
>
> Есть IDE, которые предназначены для работы только с одним языком программирования, однако большинство современных IDE позволяет работать сразу с несколькими.

UML расшифровывается как *унифицированный язык моделирования*. Это стандарт, который в основном используется для создания объектно-ориентированных, значимых моделей документации для любой программной системы, представленной в реальном мире.

> Он предлагает богатые модели, которые описывают работу любых программных / аппаратных систем.

https://coderlessons.com/tutorials/kompiuternoe-programmirovanie/uchebnik-uml/13-luchshie-instrumenty-uml#:~:text=UML%20расшифровывается%20как%20унифицированный%20язык,системы%2C%20представленной%20в%20реальном%20мире.

РАЗДЕЛ 2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

выше

Приложение А

Примеры предметных областей

1. **Цветочница или магазин цветов** (Flower Shop). В магазине цветов можно собрать букет из соответствующих цветов. Необходимо определить вес букета, его стоимость и самый дорогой/недорогой цветок (или цветы, если они одинаковы по стоимости).
2. **Продуктовый магазин** (Grocery Store). Есть магазин, в который каждый день приходят покупатели за продуктами. Необходимо подсчитать среднюю выручку магазина за сутки (или за месяц, или год и т.д.), а также найти покупателя, который оставил в магазине само больше (меньше) денег за отчётный период.
3. **Новогодняя ёлка** (Christmas tree). Есть новогодняя ёлка, которую можно украсить соответствующими новогодними игрушками. Необходимо подсчитать вес всей ёлки вместе с игрушками, её общую стоимость, а также найти самую дорогую игрушку.
4. **Кредиты** (Credits). В банке можно сформировать набор предложений клиенту по соответствующим целевым кредитам. Необходимо подсчитать общую сумму по всем кредитам клиента и найти максимальную ставку по кредиту и самый дорогой кредит клиента.
5. **Жилищно-Коммунальное Хозяйство, ЖКХ** (Housing and Communal Services). В ЖКХ предлагают набор соответствующих услуг по обслуживанию и эксплуатации жилищного хозяйства клиента. Необходимо подсчитать общую стоимость услуг, которые были оказаны клиенту ЖКХ и найти самую дорогую (недорогую) услугу.
6. **Турагентство и Туристические путевки** (Tourist trips). В турфирме можно сформировать набор предложений клиенту по выбору туристической путевки различного типа. Необходимо подсчитать общую сумму, которую клиент должен заплатить за выбранные им путёвки и найти самую выгодную по стоимости путёвки, исходя из расчёта стоимости за один день.
7. **Ресторан** (Restaurant). Есть ресторан соответствующей кухни, который своим посетителям предоставляет разнообразное меню блюд. Необходимо подсчитать общую сумму заказа соответствующего столика или общее количества денег, который ресторан получил за отчётный период (день, месяц и т.д.). Также необходимо найти чек (столик) с максимальной (минимальной) суммой заказа.
8. **Шеф-повар** (Chef). Необходимо приготовить основное блюдо (или десерт), овощной и фруктовый салаты и определить вес салата и его калорийность, а также найти самый калорийных продукт в блюде и салате.
9. **Налоги** (Taxes). Определить множество и сумму налоговых выплат физического лица за год с учетом доходов с основного и дополнительного мест работы, авторских вознаграждений, продажи имущества, получения в подарок денежных сумм и имущества, переводов из-за границы, льгот на детей и материальную помощь.
10. **Игровая комната** (Game room). Есть игровая комната с соответствующими игрушками. Необходимо провести подсчёт стоимости всех игрушек, их общий вес и найти самую дорогую игрушку.
11. **Камни** (Stones). В ювелирном магазине можно отобрать соответствующие драгоценные камни для ожерелья. Необходимо подсчитать общий вес (в каратах) и стоимость изделия. Определить самый дорогой камень в ожерелье.
12. **Звукозапись** (Sound Recording). В звукозаписывающей студии можно записать на диск сборку музыки. Необходимо подсчитать продолжительность всего диска и найти самую короткую (длинную) песню.

13. **Компьютерный герой** (Game Hero). Определить сильные стороны героя (его способности: ум, сила, ловкость и т.д.) на базе собранных им артефактов. Подсчитать общую стоимость артефактов, а также величину его соответствующих способностей.
14. **Новогодний подарок** (New Year gift). Есть новогодний сладкий подарок, который состоит из конфет и прочих сладостей. Необходимо подсчитать общий вес и стоимость подарка, а также самую дорогую (недорогую) сладость.
15. **Страховое агентство** (Insurance Company). Есть страховая фирма, которая предлагает страховые услуги и обязательства своим клиентам. Необходимо подсчитать общую стоимость страховых услуг, оказываемых конкретному клиенту, а также найти самую дорогую (недорогую) услугу.
16. **Видео/Компьютерная игра** (Video/Computer Game). Есть крутая игра, направленная на зарабатывание игроками в процессе игры определённого количества «качества» – это может быть соответствующие баллы, время, уровни, виртуальные деньги, алмазы, брильянты, золото, другие виртуальные ресурсы и т.д. Необходимо определить максимальное количество «качества», заработанное всеми игроками в данной видео игре, а также выявить победителя (– игрок с максимальным количеством «качества») и аутсайдера (– игрок с минимальным количеством «качества») на данный момент.
17. **Авиакомпания (Airline).** Есть авиакомпания, которая состоит из соответствующих самолётов. Необходимо подсчитать общее количество мест для пассажиров и грузоподъемность всех самолётов авиакомпании, а также найти самый вместительный самолёт авиакомпании.
18. **Железнодорожный транспорт** (Railway Transport). Имеются поезда, которые состоят из соответствующих вагонов. Необходимо подсчитать общую длину конкретного поезда, а также найти самый длинный (короткий) вагон.
19. **Автосалон** (Car Center). Есть автосалон, который состоит из соответствующих машин. Необходимо подсчитать общую стоимость машин автосалона и найти самую дорогую (недорогую) машину.
20. **Таксопарк** (Taxi Station). Есть таксопарк, который состоит из соответствующего автотранспорта. Необходимо подсчитать стоимость автопарка и найти самый дорогой автотранспорт по тарифам поездки.
21. **Автостоянка** (Parking). В городе есть автостоянка (или несколько автостоянок), стоимость машиномест на которой тарифицируется по часам. Необходимо подсчитать общую сумму, которую получает автостоянка за сутки (или любой другой отчётный период), а также автовладельца, который суммарно заплатил больше (меньше) всего денег за парковку.
22. **Грузоперевозки** (Cargo Transportation). Есть транспортная компания, которая занимается грузовыми перевозками. Необходимо подсчитать максимальное количество грузов, которое за один раз может осуществить компания, исходя из имеющего собственного грузового транспорта, а также найти транспорт, который перевозит максимальное (минимальное) количество груза.
23. **Пассажирские перевозки** (Passenger Operations). Есть транспортная компания, которая занимается пассажирскими перевозками. Необходимо подсчитать максимальное количество пассажиров, которых за один раз может перевезти компания, исходя из имеющего собственного пассажирского транспорта, а также найти транспорт, который перевозит максимальное (минимальное) количество пассажиров.
24. **Фургон кофе** (Coffee Car). Есть фургон определенного объема, в который можно загрузить на определенную сумму соответствующие упаковки кофе. Необходимо определить, сколько упаковок может влезть в фургон заданного объёма и на какую общую сумму.

25. **Морской порт** (Sea Port). Есть морской порт с пирсами (причалами), на которых каждый день происходит выгрузка и загрузка соответствующих грузов. Необходимо подсчитать общий грузопоток, который проходит через данный порт за сутки (или любой другой период), а также определить максимально (минимально) загруженный пирс (причал).
26. **Рыцарь** (Knight). Есть рыцаря, который экипирован соответствующей амуницией. Необходимо определить общую стоимость всей амуниции, а также найти самую дорогую (недорогую) амуницию рыцаря.
27. **Мобильная связь** (Mobile Communication). Компания-оператор мобильной связи предлагает своим клиентам различные тарифные планы использования мобильной связи. Необходимо определить общую сумму дохода компании в месяц, которую она получает от клиентов за использования соответствующих тарифных планов, а также найти клиента, который заплатил больше (меньше) других.
28. **Футбольный Менеджер** (Football Manager System). Есть футбольные клубы, у которых есть соответствующие характеристики, влияющие на исход встречи с противником. Необходимо определить шансы футбольной команды на победу против команды-соперника, а также общие шансы на победу в кубке и чемпионате соответствующей страны.
29. **Вклады или кредиты** (Deposits or Credits). Банки предлагают своим клиентам соответствующие вклады или кредиты с различными процентами. Необходимо определить общую сумму дохода соответствующего банка в месяц, которую он получает согласно соответствующей марже. Найти самые дорогие (недорогие) кредиты с учётом всех платежей или самые выгодные (невыгодные) вклады с учётом процентной ставки и доходности по ним.
30. **Зоопарк** (Zoo). Есть зоопарк животных, которые ОЧЕНЬ любят кушать. Необходимо определить общий суточный запас продуктов, необходимых зоопарку, чтобы прокормить всех своих животных. Также необходимо найти самых (менее) прожорливых животных зоопарка.
31. **Спортивная рыбалка** (Sports Fishing). В соревнованиях по спортивной рыбалке собираются команды из нескольких рыбаков. В процессе соревнований каждая из команд за определённое время должна наловить максимальное количество рыбы. Необходимо определить общий улов всех команд, а также найти команду победителя (команда, которая имеет наибольший улов рыбы) и аутсайдера (команда, которая имеет самые скромные результаты по улову).
32. **Компания по разработки программного обеспечения, IT-компания** (Software Company). Есть IT-компании, которые состоят из соответствующих сотрудников. Каждый сотрудник за свою работу получает соответствующую зарплату. Необходимо найти общий фонд заработной платы компании на месяц, а также определить сотрудников с максимальной (минимальной) заработной платой. А также определите самого производительного и полезного сотрудника.
33. **Домашние электроприборы** (House Equipments). Подсчитать потребляемую мощность всех бытовых приборов или только тех, которые включенный в данный момент. Найти также бытовой прибор, который наиболее (наименее) энергозатратный.
34. **Пчелиная пасека** (Bee Apiary). На пасеке есть несколько домиков с ульями пчёл. Каждая пчела собирает за день определённое количество мёда и приносит его в свой улей. Необходимо подсчитать общее количество мёда в день, которая даёт вся пасека, а также определить улей, которые даёт больше (меньше) всего мёда.
35. **Поликлиника** (Clinic/Hospital). В городе есть несколько поликлиник (больниц), куда обращаются пациенты, у которых есть проблемы со здоровьем. Необходимо подсчитать общее количество обращений во все поликлиники (больницы) города, а также найти самую (менее) загруженную по посещению поликлинику (больницу).

36. **Библиотека** (Library). Библиотека имеет собственный книжный фонд и предоставляет своим читателям книги (и другую литературу), которые можно почитать как в самой библиотеки, так и взять с собой на определённый период. Необходимо подсчитать, сколько сейчас книг на руках у читателей, а также определить самую (менее) популярную книгу у читателей.
37. **Тюрьма** (Jail/Prison). Есть тюрьма, в которой содержаться заключённые, осужденные по разным статьям и на различный срок. Необходимо подсчитать суммарное количество дней (или месяцев, или лет, …) всех заключённых, а также найти самую (менее) популярную уголовную статью, за которую отбывают наказание осуждённые, или найти самый большой (короткий) срок заключённого.
38. **Суд или судебное делопроизводство** (Court). В городе есть суд, который каждый день рассматривает уголовные дела или другие правонарушения, касающиеся нарушения закона. Необходимо подсчитать: общее количество дел, которые были рассмотрены судом за отчётный период; сколько из общих дел были оправдательными, а сколько – с доказанной виной и повлёкшим к заключению под стражу виновных. Можно дополнительно подсчитать общее число лиц, которые были осуждены на соответствующие сроки и сейчас находятся в местах отбывания наказания, а также найти осуждённых, которым судья установил максимальный (минимальный) срок.
39. * **Придумайте** свою предметную область, которая вам интересна.

выше