Задания по курсу объектно-ориентированного программирования (ООП)

Для выполнения заданий необходимо создать аккаунт на публичном интернет ресурсе github.com и репозиторий «ООР».

Репозиторий должен быть доступен для чтения лектору(alexander-a-vlasov) и преподавателю ведущему семинарские занятия (alex-ks).

Задания выполняются на языке Java в программном средстве IntelliJ IDEA версии Community.

Все решения должны сопровождаться набором тестов проверяющего корректность выполненного задания.

Для выполнения первого задания «Пирамидальная сортировка» необходимо создать проект Task_1_1. Следующие задания именовать соответствующим образом.

Зеленым цветом обозначены примеры к заданиям, которые можно использовать в качестве первого теста.

Серым цветом обозначены дополнительные требования к задаче, которые не обязательны к реализации, но приносят дополнительные баллы

Желтым цветом отмечены комментарии к задаче. Дополнительные условия и ограничения.

1. Знакомство с языком программирования

1.1. Пирамидальная сортировка

Реализовать классический алгоритм пирамидальной сортировки с набором тестов.

вход:
{5,4,3,2,1}
выход:
{1,2,3,4,5}

1.2. Поиск кратчайшего пути в графе

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой или определить его отсутствие.

вход:
3 // Количество вершин в графе (вершина №1,№2,№3)
2 // Начальная вершина (нумерация вершин начинается с 1)
1 // Конечная вершина
0 1 1 // Матрица смежности (значение в матрице, длинна ребра)
4 0 1 // индекс по вертикале – стартовая вершина
2 1 0 // индекс по горизонтали – конечная вершина
выход:
3

1.3. Определить наличие подстроки в файле

На вход подаётся имя файла и строка, которую надо найти. Строка может содержать буквы любого алфавита в кодировке UTF-8.

Реализовать функцию определяющую индекс начала каждого вхождения заданной подстроки.

Размер входного файла может быть на много больше объема оперативной памяти вычислительного устройства. Размер искомой подстроки во много раз меньше доступного объема оперативной памяти.

Файл input.txt : Я хочу пирог!
вход:
input.txt
пирог
выход:
{7}

Файл input.txt : Я хочу сок!
вход:
input.txt
пирог
выход:
{}

2. Контейнеры

2.1. Стек (stack)

Реализовать класс стек с поддержкой операций push (добавить элемент), pop(взять элемент), и count(узнать количество элементов). Элементы стека могут быть разных типов. Должна быть возможность использовать стандартный итератор для обхода контейнера.

вход:
push(2)
push(7)
pop
count
выход:
{2}

2.2. Очередь с приоритетами (priority queue)

Реализовать класс очередь с приоритетами. Поддерживается две операции Insert(ключ, значение) — вставка в очередь и extrack_minimum(ключ, значение) — удаляет и возвращает значение с максимальным ключом. Тип данных ключа и значения могут быть произвольными. Должны поддерживаться стандартные механизмы итерирования контейнера.

реализовать поддержку Stream API

вход:
Insert 200, «собака»
Insert 10, «человек»
ExtractMax –> 200, «собака»
Insert 5, «пингвин»
Insert 500, «попугай»
ExtractMax -> 500, «попугай»
выход:
{{5,«пингвин»},{10,«человек»}}

2.3. Дерево (tree)

Реализовать класс контейнер «Tree»(дерево) с поддержкой следующих операций:

1) итератор для перемещения по дереву;
2) добавить/удалить элемент;
3) получить/добавить поддерево;
4) реализовать итераторы совместимые со стандартными реализующие обход в глубину и ширину

Реализовать поддержку ConcurrentModificationException.

вход:
setRoot «НГУ»
add «НГУ» «ММФ»
add «НГУ» «ФФ»
add«НГУ» «ФИТ»
add «НГУ» «ГГФ»
add «ФИТ» «СИ»
add «ФИТ» «КОИ»
выход:

Обход в ширину {«НГУ»,«ММФ», «ФФ», «ФИТ», «ГГФ», «СИ», «КОИ»}

~~2.4. Дерево квадрантов (quadtree)~~

Для множества начальных объектов на плоскости построить «Quadtree»(Дерево квадрантов) с размером XxY со следующими операциями:

1) добавить новый объект;
2) удалить существующий объект;
3) получения все объекты в заданном прямоугольнике;

реализовать поддержку Stream API

3. Типы данных

3.1. Григорианский календарь

Реализовать класс григорианского календаря:

1) високосный год (28/29 февраля)
2) завершающий год века, если он не делится на 400, не является високосным
3) день недели

Выполнить операции с календарем:

1) Какой день недели будет через 1024 дня?
2) Сколько лет, месяцев и дней назад был день победы 9 мая 1945 года?
3) В какой день недели вы родились?
4) Какой месяц будет через 17 недель?
5) Сколько дней до нового года?
6) Ближайшая пятница 13-го числа месяца?

Использование классов java.util.Date, java.util.Calendar из пакета java.time запрещено

3.2. Зачетная книжка

Реализовать класс электронная зачетной книжки студента ФИТ и обеспечить следующие функции:

1) текущий средний бал за все время обучения;
2) может ли студент получить «красный» диплом с отличием;
3) будет ли повышенная стипендия в этом семестре.

Для первого теста используйте данные из Вашей зачётной книжки

Требования для диплома с отличием:

- 75% оценок в приложении к диплому(последняя оценка) – «отлично»
- Нет в зачетной книжке итоговых оценок «удовлетворительно»
- Квалификационная работа защищена на «отлично»

3.3. Упорядоченные множества

Реализовать библиотеку для выполнения классических операций над объектами: множество с , частично-упорядоченное множество(ЧУМ), решёточно-упорядоченное множество(РУМ), линейно-упорядоченное множество(ЛУМ):

1) проверка выполнения транзитивность для операции сравнения между элементами массива;
2) топологическая сортировка массива по возрастанию (текущий элемент отсортированного массива больше или не сравним чем все предыдущие);
3) поиск максимальных элементов в массиве.

С помощью созданной библиотеки решите несколько прикладных задач:

1) Найти самых умных студентов в группе.
2) Зная функцию вероятности победы каждой команды любого футбольного матча на турнире. Составить турнирную таблицу по олимпийской системе так, чтобы заданная команда выиграла как можно больше матчей.

4. Консольные приложения

4.1. Калькулятор

Реализовать калькулятор инженера для вещественных чисел. Пользователь вводит на стандартный поток ввода выражение в префиксном виде. Калькулятор вычисляет значение и выводит на стандартный поток вывода. Кроме стандартных операций (+, -, *, /) есть набор функций (log, pow, sqrt, sin, cos).

С помощью динамической загрузки библиотеки(plugin) переопределить поведение базовых операций. Добавить поддержку работы с градусами и комплексными числами.

вход:
sin + - 1 2 1
выход:
0

4.2. Записная книжка

Сделать записную книжку с набором функций доступных с командной строки:

- Добавить запись
- Удалить запись
- Вывести все записи, отсортированные по времени добавления
- Вывести отсортированные по времени добавления записи из заданного интервала времени, содержащие в заголовке ключевые слова.

Данные записной книги сериализуются в файл формата JSON. Для работы с форматом JSON рекомендуется использовать библиотеки Jackson или Gson. Для анализа параметров коммандной сроки, так же рекомендуется использовать сторонние библиотеки.

Примеры:
notebook -add “Моя заметка” “Очень важная заметка”
notebook -rm “Моя заметка”
notebook -show
notebook -show “14.12.2019 7:00” “17.12.2019 13:00” “МоЯ” “Твоя” “мне”

5. Многопоточные вычисления

5.1. Простое число

У вас имеется массив целых чисел, необходимо определить есть ли в этом массиве хотя бы одно не простое (делится без остатка только на себя и единицу). Необходимо предоставить две реализации с последовательным и параллельным решением. Удалось ли ускорить решение задачи за счет применения параллельных вычислений?

вход:
{6,8,7,13,9,4}
выход:
True

вход:
6997901 6997927 6997937 6997967 6998009 6998029 6998039 6998051 6998053
выход:
False

Реализовать с помощью ThreadPool без использования StreamAPI

5.2. Биржа

Реализовать модель биржи. Биржа — место где «Брокеры» могут совершать операции по продаже и покупке «Товара». Цель брокера за счет проведения операций увеличить свое состояние. Каждый брокер в своем распоряжении имеет определенное количество денег и товара. Операция выполняется за счет того, что вы размещаете свои предложения по покупке и продаже товара на бирже и анализируя текущие предложения можете совершить сделку. В качестве брокера может выступать как человек управляющий командами с консоли, так и несколько типов искусственных интеллектов, реализующих свое поведение согласно стратегии: «Большой куш» — покупают и продают только если есть предложение превышающее стоимость затраченных денег на покупку заданный порог или выполняют минимизацию убытков если в течении длительного времени заработать не получилось. «Игрок» — покупает и продает как можно чаще. «Аналитик» — анализирует историю предложений на бирже и исходя из этой информации совершает сделки. За каждую минуту размещенного на бирже заказа, биржа забирает с брокера фиксированную сумму денег за свои услуги. В системе каждый брокер должен быть представлен независимой нитью исполнения в которой могут асинхронно приниматься решения.

~~5.3. Туннель~~

На загруженном участке двухколейной железной дороги имеется одноколейный туннель. С двух сторон туннель оборудован проходными светофорами, которые могут находится в двух состояниях: зеленый — проезд разрешен и красный — проезд запрещен. Так как длинна тормозного пути поезда большая, а поезд выехавший на одноколейный путь блокирует все движение, в настоящее время, все поезда останавливаются перед проходным светофором и дожидаются разрешительного сигнала светофора. Из-за того, что все поезда останавливаются перед проездом туннеля, пропускная способность всей железнодорожной

ветки остается низкой. Для увеличения пропускной способности предлагается поставить предупредительный светофор, увидев сигнал которого машинист гарантированно сможет остановить поезд к проходному светофору, а в случае если туннель свободный не уменьшая скорость движения проехать туннель. Схема туннеля приведена на рисунке.

Железнодорожные пути в районе туннеля оборудованы асинхронными датчика, которые сообщаю о наличие поезда в месте его установки. Необходимо реализовать программное средство для автоматического управления всеми светофорами туннеля на основании показаний датчиков наличия поезда. Каждый датчик должен работает в своей нити исполнения. Необходимо реализовать две стратегии управления светофорами:

1) каждый поезд останавливается перед проходным светофором;
2) поезд останавливается, только если туннель занят.