Laboratory Work #27

Using Java Collections Framework, JCF

(since JDK5.0)

LEARN. GROW. SUCCEED.

 Department: <Software of Information Systems and Technologies>
Faculty of Information Technology and Robotics
Belarusian National Technical University
by Viktor Ivanchenko / ivanvikvik@bntu.by / Minsk

ЛАБОРАТОРНАЯ РАБОТА #27

Использование контейнеров в Java. Java Collections Framework, JCF

Цель работы 

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

Требования 

1) При разработке программ рекомендуется использовать шаблон MVC и SOLID (GRASP) принципы, особенно принцип единственной ответственности (*Single Responsibility Principle*), т.к. классы рекомендуется проектироваться и реализовываться таким образом, чтобы они были слабо зависимы от других классов.
2) При выполнении каждого задания необходимо по максимуму пытаться разрабатывать универсальный код.
3) Программа должна обязательно быть снабжена комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать комментариями.
4) Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.
5) Обязательно покрыть тестами новое задание, а также новый функционал, который был добавлен при рефакторинге предыдущего проекта. Запустить тесты и проверить проекты на их работоспособность.
6) Модифицируйте свою UML-диаграмму классов с учётом внесённых изменения в архитектуру своего приложения.
7) Упаковать всю программу в jar-архив, который должен быть запускаемым.

8) При разработке программ придерживайтесь соглашений по написанию кода на JAVA (Java Code-Convention).

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 JCF;
- добавить возможность сортировки 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 – The Strategy Pattern);
- 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).

Дополнительное задание 

1) В кругу стоят N человек, пронумерованных от 1 до N. При ведении счета по кругу вычеркивается каждый второй человек, пока не останется один. Составить две программы, моделирующие процесс. Одна из программ должна использовать класс ArrayList, а вторая – LinkedList. Проанализируйте, какая из двух программ работает быстрее? Почему?
2) Задан список целых чисел и число X. Не используя вспомогательных объектов и не изменяя размера списка, переставить элементы списка так, чтобы сначала шли числа, не превосходящие X, а затем числа, большие X.

3) Написать программу, осуществляющую сжатие английского текста. Построить для каждого слова в тексте оптимальный префиксный код по алгоритму Хаффмена. Использовать класс PriorityQueue.
4) На плоскости задано N точек. Вывести в файл описания всех прямых, которые проходят более чем через одну точку из заданных. Для каждой прямой указать, через сколько точек она проходит. Использовать класс HashMap.
5) На клетчатой бумаге нарисован круг. Вывести в файл описания всех клеток, целиком лежащих внутри круга, в порядке возрастания расстояния от клетки до центра круга. Использовать класс PriorityQueue.
6) Один из способов шифрования данных, называемый «двойным шифрованием», заключается в том, что исходные данные при помощи некоторого преобразования последовательно шифруются на некоторые два ключа K1 и K2. Разработать и реализовать эффективный алгоритм, позволяющий находить ключи K1 и K2 по исходной строке и ее зашифрованному варианту. Проверить, оказался ли разработанный способ действительно эффективным, протестировав программу для случая, когда оба ключа K1 и K2 являются 20-битными (время ее работы не должно превосходить одной минуты).
7) На плоскости задано N отрезков. Найти точку пересечения двух отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.
8) На клетчатом листе бумаги закрашена часть клеток. Выделить все различные фигуры, которые образовались при этом. Фигурой считается набор закрашенных клеток, достижимых друг из друга при движении в четырёх направлениях. Две фигуры являются различными, если их нельзя совместить поворотом на угол, кратный 90 градусам, и параллельным переносом. Используйте класс HashSet.
9) Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу, состоящую из максимального количества одинаковых элементов. Использовать класс Stack.
10) Реализовать структуру "черный ящик", хранящую множество чисел и имеющую внутренний счетчик K, изначально равный нулю. Структура должна поддерживать операции добавления числа в множество и возвращение K-го по минимальности числа из множества.
11) На прямой гоночной трассе стоит N автомобилей, для каждого из которых известны начальное положение и скорость. Определить, сколько произойдет обгонов.
12) На прямой гоночной трассе стоит N автомобилей, для каждого из которых известны начальное положение и скорость. Вывести первые K обгонов.

13) На базе коллекций реализовать класс Graph, представляющий собой неориентированный граф. В конструкторе класса передается количество вершин в графе. Методы должны поддерживать быстрое добавление и удаление ребер.

14) На базе коллекций реализовать структуру хранения чисел с поддержкой следующих операций: добавление/удаление числа; поиск числа, наиболее близкого к заданному (т.е. модуль разницы минимален).

15) Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

16) Во входном файле хранятся две разреженные матрицы A и B. Построить циклически связанные списки CA и CB, содержащие ненулевые элементы соответственно матриц A и B. Просматривая списки, вычислить: а) сумму S = A + B; б) произведение P = A * B.

17) Во входном файле хранятся наименования некоторых объектов. Построить список C1, элементы которого содержат наименования и шифры данных объектов, причем элементы списка должны быть упорядочены по возрастанию шифров. Затем “сжать” список C1, удаляя дублирующие наименования объектов.

18) Во входном файле расположены два набора положительных чисел; между наборами стоит отрицательное число. Построить два списка C1 и C2, элементы которых содержат соответственно числа 1-го и 2-го набора таким образом, чтобы внутри одного списка числа были упорядочены по возрастанию. Затем объединить списки C1 и C2 в один упорядоченный список, изменяя только значения полей ссылочного типа.

19) Во входном файле хранится информация о системе главных автодорог, связывающих г.Минск с другими городами Беларуси. Используя эту информацию, постройте дерево, отображающее систему дорог республики, а затем, продвигаясь по дереву, определить минимальный по длине путь из г.Минска в другой заданный город. Предусмотреть возможность для последующего сохранения дерева в виртуальной памяти.

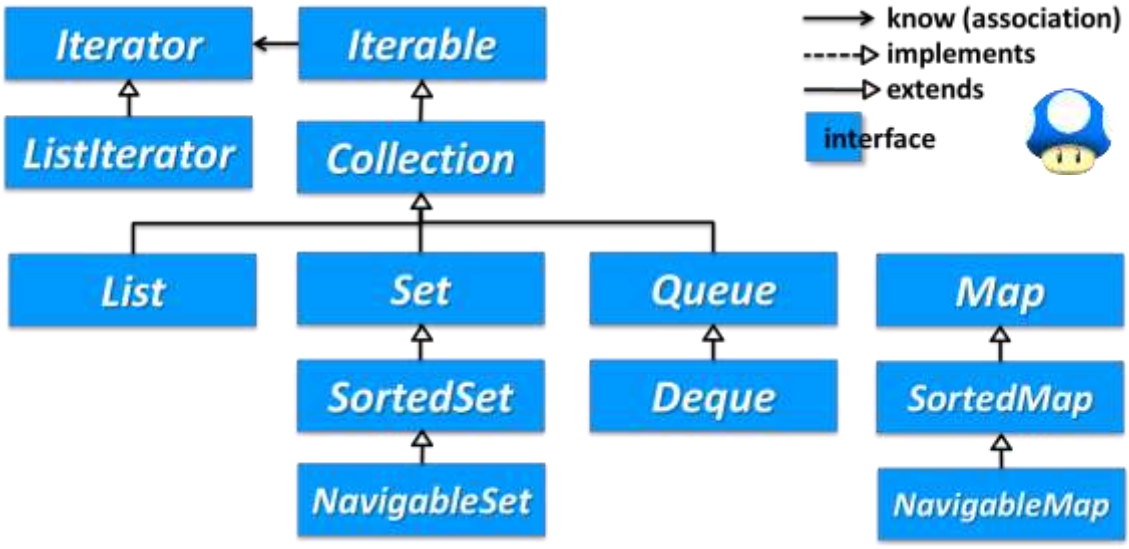
Best of LUCK with it, and remember to HAVE FUN while you’re learning :)

Victor Ivanchenko

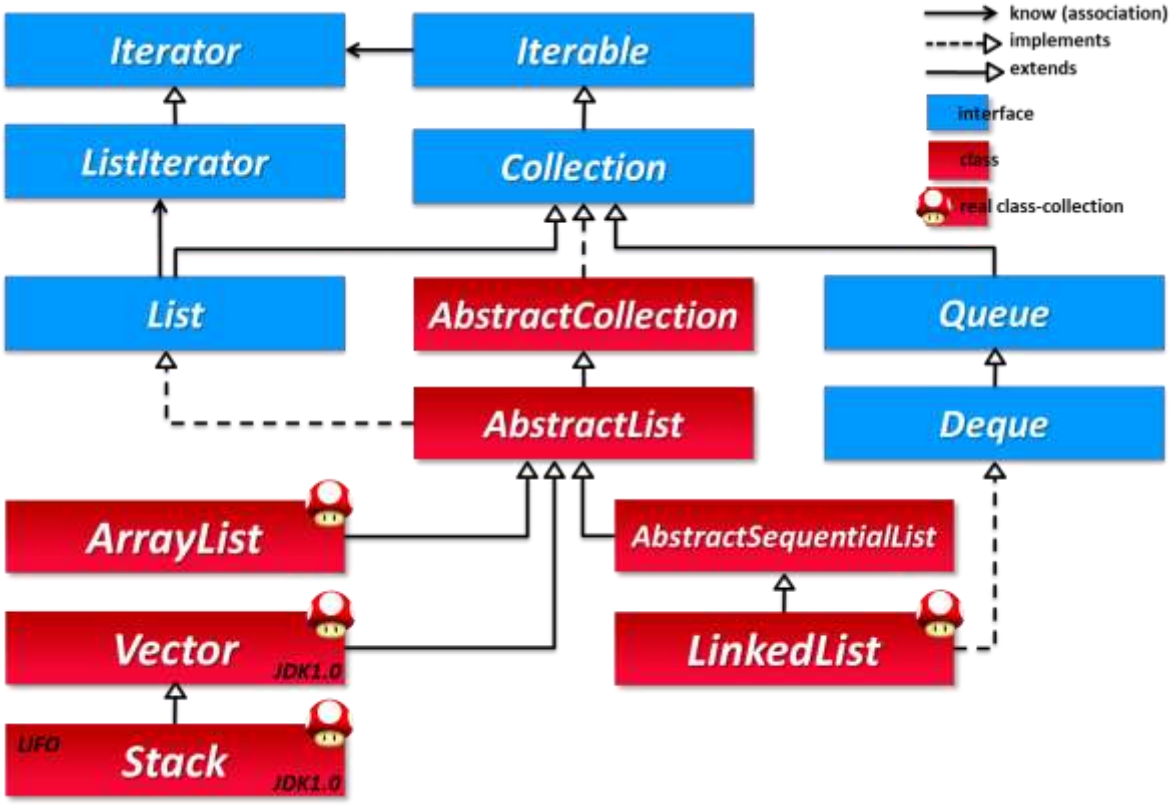
Что нужно запомнить (краткие тезисы)

Перед написанием очередного приложения, очень важно вспомнить (запомнить) следующие вещи о *Java Collections Framework (JCF)*.

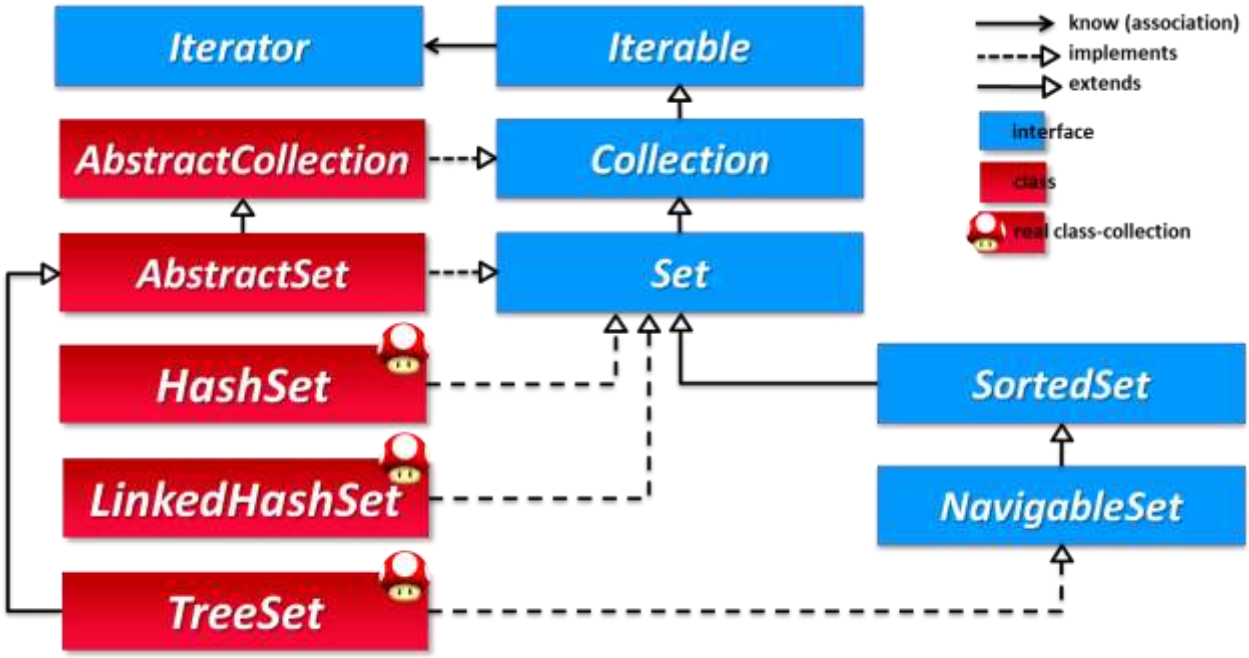
1) Библиотека *JCF* делиться на три части: раздел интерфейсов поведения контейнеров (интерфейсы), раздел реализации данного поведения (классы-имплементации) и раздел классов, предоставляющих готовые алгоритмы для обработки содержимого контейнеров.
2) Общая UML-диаграмма основного поведения (основных интерфейсов) библиотеки *JCF* представлена ниже на рисунке:



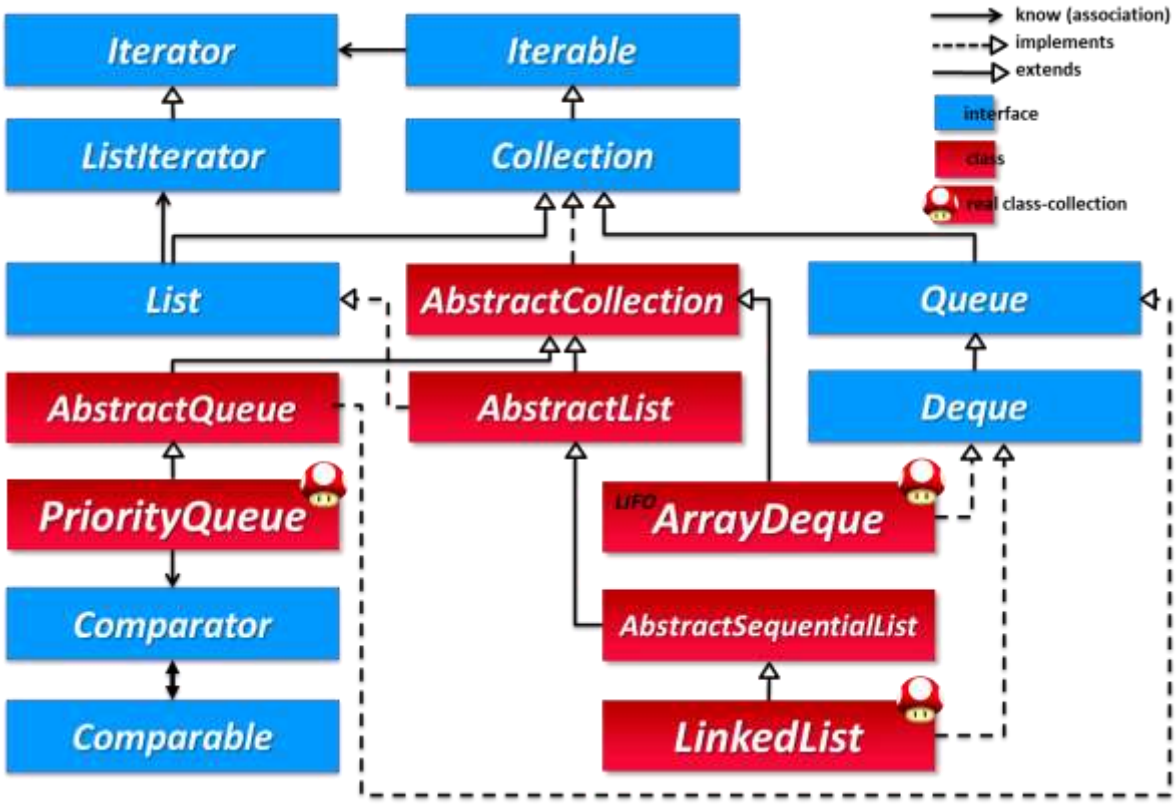
3) Общая *UML*-диаграмма основного поведения (интерфейсов) списков и их классы-имплементации представлены ниже на рисунке:



4) Общая *UML*-диаграмма основного поведения (интерфейсов) множеств и их классы-имплементации представлены ниже на рисунке:



5) Общая *UML*-диаграмма основного поведения очередей и их классы-имплементации представлены ниже на рисунке:



6) ...

Контрольные вопросы

1) Дайте определения Java-контейнерам (коллекциям) и перечислите основные концепции их использования, а также преимущества по сравнению с другими типами данных.
2) Какие данные могут хранить контейнеры?
3) Опишите общую архитектуру (картину) и иерархию библиотеки *Java Collections Framework (JCF)*: раздел интерфейсов, раздел абстрактных и конкретных классов-реализаций, раздел классов-алгоритмов.
4) В каких пакетах размещается библиотека *JCF*?
5) С помощью какой абстракции в *JCF* можно перебрать элементы любой коллекции?
6) Где и как используется паттерн *Iterator*, какова его реализация в *JCF*?
7) Зачем нужны интерфейсы *Iterable* и *Iterator*?
8) Какова специфика работы объекта, реализующий интерфейс *Iterator*?
9) В каких случаях может быть выброшено *ConcurrentModificationException*?
10) Какой интерфейс необходимо реализовать в пользовательском классе и как это сделать, чтобы объект данного класса можно было использовать с модифицированным в *JDK5.0* циклом *for* (в простонародий, *foreach*)?
11) Опишите основное поведение любой коллекции (перечислите все методы, которые декларирует интерфейс *Collection*).
12) Опишите стандартные интерфейсы-поведения *JCF*: *List, Set (SortedSet, NavigableSet), Queue, Deque, Map (SortedMap, NavigableSet)* и др.
13) Для чего нужны абстрактные классы *JFC*: *AbstractCollection, AbstractSequentialList, AbstractList, AbstractMap, AbstractSet, AbstractQueue* и др.
14) Опишите особенности и функционал основных классов-реализаций поведения списков (реализации интерфейса *List*), их внутреннюю архитектуру, а также их преимущества и недостатки: *ArrayList*, *LinkedList* и др.
15) Когда лучше использовать *ArrayList* и *LinkedList*, а в каких случаях разумно использовать массив?
16) Перечислите доступные способы перебора элементов списка. Зачем для списков нужен дополнительный интерфейс *ListIterator*?

17) Какая разница между объектами, реализующий интерфейс *Iterator,* и объектами, реализующий интерфейс *ListIterator*?

18) Опишите особенности и функционал основных классов-реализаций поведения множества (расширение и реализации интерфейса *Set*), их внутреннюю архитектуру, а также их преимущества и недостатки: *HashSet*, *LinkedHashSet*, *TreeSet* и др.

19) Как и при помощи какой логики поддерживается уникальность объектов соответствующими реализациями интерфейса *Set*?

20) В какой из реализаций поведения *Set* не допускается в качестве значения добавлять *null*-ссылку?

21) Опишите особенности и функционал основных классов-реализаций алгоритмов абстрактных структур данных типа очереди, стека, дека и т.д. (расширение и реализации интерфейса *Queue*), их внутреннюю архитектуру, а также их преимущества и недостатки: *PriorityQueue*, *ArrayDeque* и др.

22) Опишите особенности и функционал основных классов-реализаций поведения контейнеров, поддерживающих отображение, т.е. контейнеры, которые хранят элемент в виде пары «ключ-значение» (расширение и реализации интерфейса *Map*), их внутреннюю архитектуру, а также их преимущества и недостатки: *HashMap*, *LinkedHashMap*, *TreeMap* и др.

23) Что будет, если в *Map* положить две пары элементов с одинаковым значением ключей?

24) Охарактеризуйте потокобезопасные контейнеры, которые существовали с *JDK 1.0* ещё до выхода *JCF* и основное предназначение каждого: *Vector*, *Stack*, *BitSet*, *Dictionary*, *Hashtable*, *Properties* и др.;

25) Чем отличается коллекция *ArrayList* от коллекций *Vector* и *Stack*?

26) Чем контейнер *Hashtable* отличается от контейнера *HashMap*? На сегодняшний день контейнер *Hashtable* является не рекомендованных для использования (*deprecated)*, как все-таки использовать функциональность данного контейнера?

27) Опишите утилитные классы для работы с контейнерами (а также с массивами), т.е. классы раздела алгоритмов библиотеки *JCF*: *Collections*, *Arrays* и др.

28) Как задается порядок следования объектов в контейнерах? Как отсортировать коллекцию? Как перетасовать элементы коллекции?

29) Как получить не модифицируемую (только для чтения) коллекцию?

30) Как получить типобезопасную (все элементы имеют один и тот же тип) коллекцию?

31) Какой необходимо реализовать интерфейс для того, чтобы объекты пользовательского класса могли быть, к примеру, отсортированы в своём естественном виде (*natural ordering*)?

32) В чем разница между интерфейсами *Comparable* и *Comparator*?

33) Опишите архитектуру и поведение поведенческого шаблона проектирования Стратегия (*The Strategy Pattern*). Когда и где его применяют?

34) На каких принципах базируется поведенческий шаблон проектирования Стратегия?

35) На какие области *JCF* повлиял поведенческий шаблон проектирования Стратегия?