Оглавление

[Тема занятия 1](#_Toc465801616)

[Длительность 1](#_Toc465801617)

[Цели 1](#_Toc465801618)

[Наполнение занятий 1](#_Toc465801619)

[Т1 1](#_Toc465801620)

[Введение 1](#_Toc465801621)

[Списки (List) 1](#_Toc465801622)

[Ассоциативный массив (Map) 2](#_Toc465801623)

[Множества (Set) 3](#_Toc465801624)

[Iterator 3](#_Toc465801625)

[П1 4](#_Toc465801626)

[Заключение 4](#_Toc465801627)

# Тема занятия

Collection framework

# Длительность

Теоретическая часть – 1 час

Практическая часть – 1 час

Цели занятия

Получение первоначальных знаний о структуре и API по работе с основными коллекциями, особенностях их использования и категориях задач, в которых применима та или иная коллекция.

Наполнение занятий

## Т1

### Введение

Основное предназначение – работа с динамическими наборами объектов. Главное отличие от массивов – динамическая размерность.

Расположение – пакет java.util, а значит требуется указывать подключаемые типы в разделе описания импортируемых типов.

*Какой пакет не требует указания в разделе import? (Ответ – java.lang)*

Наиболее часто используемые типы и их иерархия:

Interface Collection

Interface Set

Interface List

Interface Map

### Списки (List)

Предназначение: работа с набором элементов

Наиболее часто используются 2 реализации

ArrayList

Структура основана на массиве. Как только в текущем массиве заканчивается место, выделяется новый массив (правила изменения размерности – см. capacity) и вызывается метод System.arrayCopy. Этот статический метод реализован нативно и копирует массивы на уровне ява машины, следовательно, копирование происходит быстро.

LinkedList

Структура, основанная на классическом двусвязном списке. При добавлении элемента создается новый элемент, в котором хранится ссылка на следующий и предыдущий элемент.

*В каких категориях задач эффективнее использовать ArrayList? (Ответ – когда требуется доступ к произвольным элементам списка)*

*Какова оценка сложности доступа к произвольному элементу ArrayList (Ответ – О(1))*

*В каких категориях задач эффективнее использовать LinkedList? (Ответ – когда требуется вставка или удаление из середины списка)*

*Какова оценка сложности доступа к произвольному элементу LinkedList (Ответ – О(n))*

### Ассоциативный массив (Map)

Предназначение: работа с набором элементов

*Приведите пример задачи, требующей использования ассоциативного массива (Пример ответа – словарь (английские слова – ключи, русские – значения), катологи, реестры и т.д.)*

HashMap

*Что такое хеш? (Ответ – это односторонняя функция. Невзаимооднозначное отображение бесконечного множества на конечное. В случае java – отображение множества объектов в целое число типа int)*

Основан на хранении пар ключ-значение и поиску значений в 2 этапа.

Структура представляет собой набор корзин, в каждой корзине хранится список элементов Map.Entry. Примерное описание структуры Map.Entry:

MapEntry{

Object key

Object value

Map.Entry next

}

Хеш-код ключей каждой из пар, находящихся в корзине одинаковый.

Алгоритм добавления ассоциации:

1. Вызывается метод put(key, value)
2. У аргумента key вызывается функция hashCode, которая возвращает число типа int
3. По полученному значению вычисляется корзина, содержащая цепочку пар
4. Аргумент key сравнивается с ключом каждой пары в цепочке с помощью вызова метода equals
   1. Если сравнение с ключом какой-то пары выдает true, значит в структуре уже есть ассоциация с аргументом key. В этом случае значение пары подменяется аргументом value, метод put возвращает старое значение
   2. Если ни одно сравнение не дало положительного результата, значит произошла коллизия. Создается новая пара и добавляется в конец цепочки. Метод put возвращает null

Алгоритм поиска значения по ключу:

1. Вызывается метод get(key)
2. У аргумента key вызывается функция hashCode, которая возвращает число типа int
3. По полученному значению вычисляется корзина, содержащая цепочку пар
4. Аргумент key сравнивается с ключом каждой пары в цепочке с помощью вызова метода equals
   1. Если сравнение с ключом какой-то пары выдает true, значит в структуре есть ассоциация с аргументом key. В этом случае метод get возвращает значение этой пары
   2. Если ни одно сравнение не дало положительного результата, значит структура не содержит ассоциации с аргументом key. Метод get возвращает null

TreeMap

Структура основана на бинарном красно-черном дереве.

*Что такое бинарное дерево? (Ответ – дерево, в котором у каждого узла не более 2 потомков).*

*Какое основное свойство красно-черных деревьев? (Ответ – это дерево самобалансирующееся. Это означает, что вес любых двух путей не может отличаться более, чем на единицу)*

*Какая операция должна быть определена для построения бинарного дерева? (Ответ – операция сравнения)*

Для построения структуры необходимо либо использовать ключи, реализующие интерфейс Comparable, либо при создании дерева передать ему объект, расширяющий тип Comparator.

Весь функционал тривиален и далее не рассматривается.

### Множества (Set)

Предназначение: работа с набором уникальных элементов

HashSet

Структура, содержащая в себе HashMap в качестве хранилища. Т.к. в HashMap не могут встретиться два одинаковых ключа, то элементы, добавляемые в Set используются в качестве ключей в HashMap. В качестве значения выступает заглушка – простой объект.

TreeSet

Структура, основанная на TreeMap.

*Мы рассмотрели все основные структуры. Есть ли вопросы по API работы с коллекциями (Ответ – не ясно, как получить доступ к элементам множеств)*

### Iterator

Предназначение: позволяет поэлементно проходить по коллекциям (ассоциативный массив не реализует Collection).

Методы:

hasNext() – возвращает true, если есть следующий элемент

next() – возвращает следующий элемент

remove() – удаляет последний возвращенный элемент

Варианты использования итераторов:

Цикл while:

Iterator itr = someSet.iterator();

Object element;

while(itr.hasNext()){

element = itr.next();

…

}

Цикл for:

for(Object element: someSet){

…

}

## П1

Основная задача – создать реестр студентов и их знаний.

Ход исполнения:

1. Создать новый проект в Idea
2. Создать package
3. Создать сущность студент (POJO), содержащий набор свойств. В первом варианте реализации не переопределять методы hashCode и equals
4. Создать сущность умение – Skill, содержащий набор свойств.
5. Создать основной класс с точкой входа main
6. В методе main создать map – реестр. В качестве ключей использовать студентов, в качестве значения – список умений.
7. Наполнить реестр ассоциациями.
8. Показать обязательность переопределения hashCode и equals – создать еще один объект студент, аналогичный тому, который уже есть в реестре. Попытаться получить его умения – результат должен быть null
9. Переопределить метод hashCode и equals в типе Student. После этого еще раз выполнить программу. Результат предыдущего пункта должен возвращать значение из реестра
10. Написать код, пробегающий по списку умений студента и выводящий их на экран
11. Показать недостатки использования Mutable объектов в качестве ключа. Сохранить прямую ссылку на любого студента, после того, как он будет добавлен в реестр, изменить свойство, участвующее в вычислении hashCode. После это попробовать получить список его умений. Результат должен быть null – изменение хеш-кода привело к тому, что ассоциация ищется в другой корзине.
12. Сделать тип Student immutable, показать, что теперь такая ситуация невозможна програмно.
13. Написать код, выводящий все содержимое мап на печать

## Заключение

Ответ на задаваемые вопросы, анонс следующего занятия.

# Материалы для подготовки:

Вопросы по коллекциям

https://habrahabr.ru/post/162017/