ORACLE®

Коллекции в Java

начальный набор практических знаний

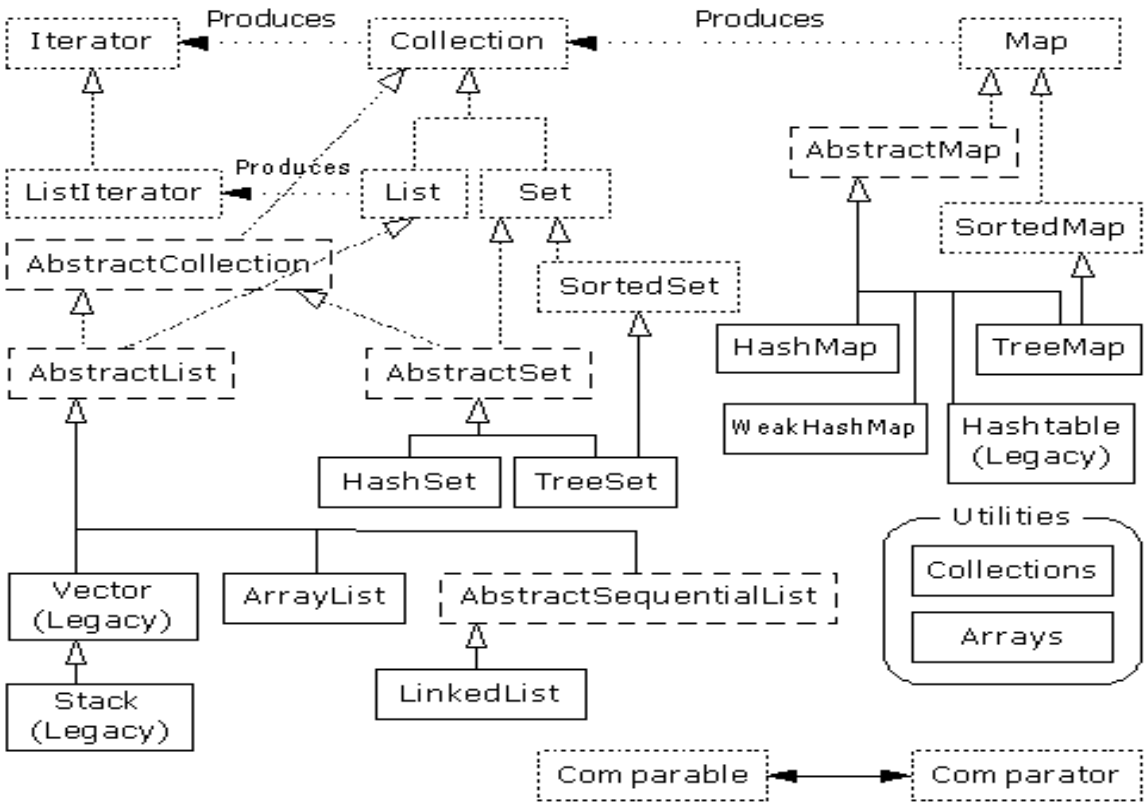
Дмитрий Бессонов

dmitry.bessonov@oracle.com

Материалы, почитать на досуге самостоятельно

- http://www.bruceeckel.by.ru/tij/Chapter09.html
- http://ru.wikipedia.org/wiki/Хеширование
- Исходный код некоторых классов и интерфейсов пакета java.util
  (поставляется вместе с Oracle JDK)
  - **`java.util.Collection`**
  - **`java.util.ArrayList`**
  - **`java.util.HashMap`**
  - **`java.util.HashSet`**

Коллекции, взгляд с высоты



Сегодня игрок в StarCraft2 – наш ключевой объект

```
public class SC2Player {
    String nickName;
    int points;
    League league;
    Race race;
}

public enum Race {TERRAN, PROTOSS, ZERG}

public enum League { PRO, DIAMOND, PLATINUM,
                     GOLD, SILVER, BRONZE, PRACTICE }
```

Массивы

```
// игра на троих

SC2Player[] battle = new SC2Player[3]

battle[0] = new SC2Player("SuperMax",
                          League.SILVER, 57);
battle[1] = new SC2Player("IvanZergling",
                          League.GOLD, 100);
battle[2] = new SC2Player("killer13fx",
                          League.PRACTICE, 0);

// четвертый - лишний, жди следующей !
```

Массивы

```
// игра на четверых,
// другой вариант инициализации массива
SC2Player[] battle = {
                new SC2Player("MaxRusher",
                        League.SILVER, 57),
                new SC2Player("IvanZergling",
                        League.GOLD, 100),
                new SC2Player("killer13fx",
                        League.PRACTICE, 0),
                new SC2Player("sc_master66",
                        League.DIAMOND, 1500),
};
// но пятого все равно уже не добавить
```

Массив игроков

```
// но можно отсортировать!

Arrays.sort(battle, new Comparator() {
  public int compare(Object p1, Object p2) {
      return Integer.compare(
               ((SC2Player)p1).points,
               ((SC2Player)p2).points );
  }
});

// игроки в массиве battle будут
// отсортированы по возрастанию
// количества очков
```

Список игроков – гораздо удобнее массива

```
// похож на массив по названию, но удобнее
java.util.List battleList = new ArrayList();

// вначале было трое
battleList.add(new SC2Player("player1"));
battleList.add(new SC2Player("player2"));
battleList.add(new SC2Player("player3"));

// потом первый дал слабину, осталось двое
battleList.remove(0);

// но ему быстро нашли замену
battleList.add(0, new SC2Player("player4"));
```

java.util.Collection – интерфейс определяющий коллекции объектов

- `int size()`
- `boolean isEmpty()`
- `boolean contains(Object o)`
- `Iterator<E> iterator()`
- `Object[] toArray()`
- `boolean add(Object e)`
- `boolean remove(Object o)`
- `...`

Список – интерфейс java.util.List, потомок Collection

- Упорядоченный набор объектов
- Размер списка может меняться после его создания
- Список способен содержать одинаковые объекты

Списки – потомки/имплементоры java.util.List

- Все остальное – индивидуально, выбирайте в зависимости от ваших задач
  - "Умный" массив - **`ArrayList`**
  - Связный список – быстрое добавление/удаление элементов, поиск соседей – **`LinkedList`**
  - Пакет **`java.util`** содержит много других видов списков, которые покрывают 99% потребностей разработчика

Продолжаем манипулировать игроками

```
java.util.LinkedList battleList
    = new LinkedList();

SC2Player p_1, p_2, p_3;
battleList.add(p_1 = new SC2Player("p_1"));
battleList.add(p_2 = new SC2Player("p_2"));
battleList.add(p_2 = new SC2Player("p_3"));

battleList.remove(new SC2Player("p_1"));
battleList.remove(p_2);

// сколько игроков осталось?
```

Осталось двое

```
// метод remove() использует Object.equals()
// для поиска элемента списка
// у SC2Player унаследована реализация метода
// Object.equals() сравнивающая ссылки

// результата не будет,
// передаваемого значения как будто нет в списке
battleList.remove(new SC2Player("p_1"));

// но вот это сработает:
battleList.remove(p_2);

// что сделать, чтобы сработали оба подхода?
```

Реализовать метод SC2Player.equals()

```
// современные IDE позволяют его сгенерировать

public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    SC2Player sc2Player = (SC2Player) o;

    if (points != sc2Player.points) return false;
    if (league != sc2Player.league) return false;
    if (nickName != null ? !nickName.equals(sc2Player.nickName)
            : sc2Player.nickName != null) return false;
    if (race != sc2Player.race) return false;
    return true;
}

// и этот метод сейчас нам пригодится
```

Множество – интерфейс java.util.Set, потомок Collection

- Основная особенность – не содержит одинаковых элементов
- Одинаковых с точки зрения метода **`equals()`**:
  - элемент1.`equals`(элемент2) должен возвращать false для любых элементов множества

Неупорядоченная лига уникальных игроков

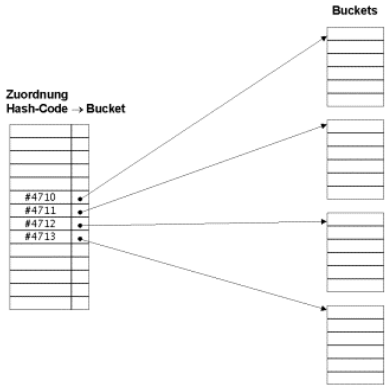
```
java.util.Set customLeague
= new HashSet(); // про hash-код чуть позже

customLeague.add(new SC2Player("VasyaX"));
customLeague.add(new SC2Player("PetyaF"));
customLeague.add(new SC2Player("AntonK"));

// Вася там уже есть,
// попытка зарегистрировать дважды под одним ником
// будет пресечена
boolean result =
    customLeague.add(new SC2Player("VasyaX"));

// все благодаря тому, что у нас есть свой метод
// SC2Player.equals()
System.out.println("result = " + result); // напечатает false
```

Вернемся к истокам

```
public class Object {

// при переопределении нужно соблюдать контракт:
// рефлексивность – объект эквивалентен себе
// симметричность
// транзитивность
// консистентность – сколько ни вызывай, результат один
public boolean equals(Object obj) {
  return (this == obj);
}

// при переопределении нужно соблюдать контракт:
// - не меняется внутри запущенной программы для объекта
// - равенство хэшкодов - необходимое условие эквивалентности
//   но не достаточное !!!
public native int hashCode();

}
```

java.util.HashSet - потомок java.util.Set

- Обеспечивает одинаковое время всех базовых операций - **`add()`, `remove()`, `contains()`, `size()`**
- Но для этого нужно “хорошо” реализовать метод **`hashCode()`**, чтобы класс **`HashSet`** мог на него опираться например при определении неэквивалентных объектов



Как создать упорядоченное множество, которое будет сортировать игроков по количеству очков?

```
// java.util.SortedSet отсортирует по возрастанию

java.util.SortedSet league = new TreeSet();
league.add(new SC2Player("PetyaF", 67));
league.add(new SC2Player("AntonK", 88));

// к сожалению, так просто не отсортирует
// потому, что не знает, как сравнивать игроков
// Мы получим
// java.lang.ClassCastException:

org.school30.javalesson.SC2Player cannot be cast to
java.lang.Comparable
```

java.lang.Comparable ?

```
// позволяет сравнивать два объекта
// давая ответы "<", ">" или "=" в их отношении,
// возвращая отрицательное, положительное значение или ноль
public interface Comparable {
    public int compareTo(Object o);
}

public class SC2Player implements Comparable {
    ...
    // теперь мы сможем четко понимать, кто сильнее
    public int compareTo(Object o) {
      SC2Player otherPlayer = (SC2Player)o;
      if (this.getPoints() <  otherPlayer.getPoints()) return -1;
      if (this.getPoints() > otherPlayer.getPoints()) return 1;
      return 0;
    }
}
```

Теперь игроки могут быть отсортированы по количеству очков

```
java.util.SortedSet league = new TreeSet();
league.add(new SC2Player("VanyaB", 92));
league.add(new SC2Player("AntonK", 88));
league.add(new SC2Player("PetyaF", 67));
league.add(new SC2Player("RusherX", 165));

for (Iterator iterator = league.iterator(); iterator.hasNext();) {
    System.out.println(
            ((SC2Player) iterator.next()).getNickName());
}
```

```
PetyaF
AntonK
VanyaB
RusherX
```

Как отсортировать игроков по-другому, принимая во внимание только лигу, к которой игрок принадлежит?

```
// предоставить свою реализацию интерфейса
// Comparator
// которая будет использована
// сортирующей коллекцией

public interface Comparator {

    // метод должен вернуть отрицательное,
    // положительное значение или ноль
    // если объекты “<”, “>” или равны
    // соответвтвенно
    int compare(Object o1, Object o2)
    ...
}
```

Отсортируем по League, и попробуем вывести на консоль. Что будет напечатано?

```
TreeSet ladder = new TreeSet(new Comparator() {
    public int compare(Object o1, Object o2) {
        League league1 = ((SC2Player) o1).getLeague();
        League league2 = ((SC2Player) o2).getLeague();
        return league1.compareTo(league2);
    }
});

ladder.add(new SC2Player("Anton", League.GOLD, 765));
ladder.add(new SC2Player("Vasya", League.PLATINUM, 988));
ladder.add(new SC2Player("Dima", League.PRACTICE, 34));
ladder.add(new SC2Player("Nick", League.SILVER, 980));
ladder.add(new SC2Player("Stepan", League.BRONZE, 800));

for (Iterator iterator = ladder.iterator(); iterator.hasNext();) {
    System.out.println(iterator.next());
}
```

Получим не совсем то, что мы ожидали...

```
org.school30.javalesson.SC2Player@d4f2586f
org.school30.javalesson.SC2Player@fe8bea79
org.school30.javalesson.SC2Player@f20e0c20
org.school30.javalesson.SC2Player@f2bb1a20
org.school30.javalesson.SC2Player@1c68853

// как сделать вывод списка
// удобочитаемым ?
```

Нужно переопределить метод `toString()`, отнаследованный от класса `java.lang.Object`

```
public class SC2Player {
    ...
     public String toString() {
         return "SC2Player{" + "league=" + league + ", nickName='" +
                nickName + '\'' + ", points=" + points + ", race=" +
                Race + '}';
     }
}

// тогда мы сможем проверить сортировку по лигам,
// к которой принадлежит игрок, распечатав содержимое ladder
for (Iterator iterator = ladder.iterator(); iterator.hasNext();) {
     System.out.println(iterator.next());
}
```

```
SC2Player{league=PLATINUM, nickName='Vasya', points=988, race=null}
SC2Player{league=GOLD, nickName='Anton', points=765, race=null}
SC2Player{league=SILVER, nickName='Nick', points=980, race=null}
SC2Player{league=BRONZE, nickName='Stepan', points=800, race=null}
SC2Player{league=PRACTICE, nickName='Dima', points=34, race=null}
```

Как узнать и напечатать количество очков у... Васи?

```
java.util.Set league = new HashSet();

league.add(new SC2Player("PetyaF", 34));
league.add(new SC2Player("VasyaX", 838));
league.add(new SC2Player("AntonK", 74));
league.add(new SC2Player("MishaV", 165));
league.add(new SC2Player("DimaR",  90));
```

Не очень удобным и читабельным способом :(

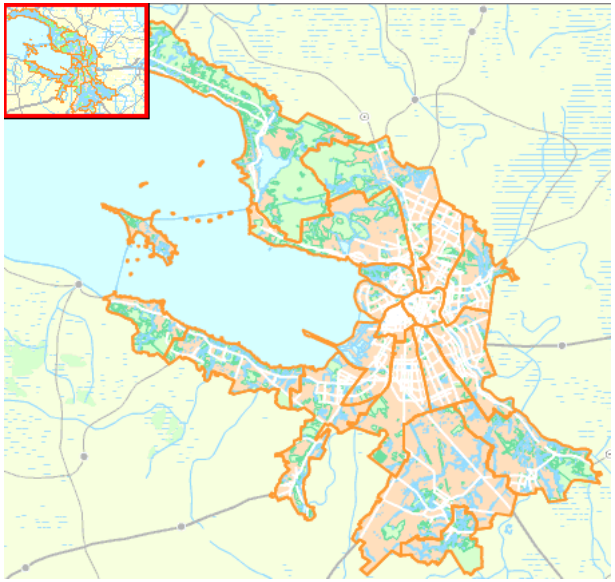
```
for (Iterator it = league.iterator(); it.hasNext();) {
    SC2Player player = (SC2Player) it.next();
    if (player.getNickName().equals("VasyaX")) {
        System.out.println(
            "VasyaX has "
            + player.getPoints() + " points");
        break;
    }
}

// есть ли в Java структора данных
// которая позволит упростить код ?
```

Есть ли структура, которая позволит упростить код до:

```
SC2Player player =
        (SC2Player)league.get("VasyaX");

System.out.println(
    "VasyaX has " +
    player.getPoints()
    + " points");
```

Есть, интерфейс java.util.Map!

- Карта, позволяющая находить реальные объекты по ключу (“координатам”)
- Ключи уникальны
- Разные реализации этого интерфейса служат разным целям
  - **`TreeMap`**
  - **`HashMap`**
  - **`...`**



Тогда игроков нужно организовать по-другому

```
java.util.Map league = new HashMap();

league.put("PetyaF", new SC2Player("PetyaF", 34));
league.put("VasyaX", new SC2Player("VasyaX", 838));
league.put("AntonK", new SC2Player("AntonK", 74));
league.put("MishaV", new SC2Player("MishaV", 165));
league.put("DimaR",  new SC2Player("DimaR",  90));


// при работе реального приложения
// наполнение лиги будет выглядеть:

league.put(player.getNickName(), player);
```

После этого можно будет легко искать игрока по имени

```
SC2Player player =
        (SC2Player)league.get("VasyaX");

System.out.println(
    "VasyaX has " +
    player.getPoints() +
    " points");
```

Домашнее задание

1. Написать систему управления игроками,
   разработав класс,
   реализующий следующий интерфейс:

```
public interface LeagueManager {

    public void addPlayer(SC2Player player);
    public void removePlayer(SC2Player player);
    public SC2Player getPlayer(String name);
    public SC2Player[] getAllPlayers();
    public SC2Player[] getPlayers(League league);
    public SC2Player[] getPlayers(Race race);
    public void addPoints(String name, int points);
}
```

2.* Написать программу, проверяющую корректность реализации интерфейса `LeagueManager`

2. Или юнит-тесты используя JUnit или TestNG**

Спасибо за внимание!