НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Объектно-ориентированное программирование на языке Java

Часть 6. Коллекции (1/2): Списки.

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

Прежде чем мы начнем...

Generics…

- Начиная с JDK 1.5, в Java появляются новые возможности для программирования. Одним из таких нововведений являются Generics **(дженерики)**. Generics являются аналогией с конструкцией "Шаблонов"(template) в C++, но имеет свои нюансы.
- С generics вы пишете код для одного типа (например, T), который применим для всех типов, вместо написания отдельных классов для каждого типа.

Пример

```
class BoxPrinter<T> {
    private T val;
    public BoxPrinter(T arg) {
        val = arg;
    }
    public String toString() {
        return "[" + val + "]";
    }
}
```

```
BoxPrinter<Integer> value1 =
        new BoxPrinter<Integer>(new Integer(10));
System.out.println(value1);
BoxPrinter<String> value2 =
        new BoxPrinter<String>("Hello world");
System.out.println(value2);
```

Пояснение

1. Рассмотрим объявление BoxPrinter:

   ```
   class BoxPrinter<T>
   ```

   После имени класса в угловых скобках "<" и ">" указано имя типа "T", которое может использоваться внутри класса.

Фактически T – это тип, который должен быть определён позже (при создании объекта класса).

Пояснение

2. Внутри класса вы сначала используете T при описании поля:

```
private T val;
```

Вы объявляете val универсального типа - фактический тип будет указан позже, когда вы будете использовать BoxPrinter.

В main () вы объявляете переменную типа BoxPrinter для Integer следующим образом:

```
BoxPrinter<Integer> value1
```

Здесь вы указываете, что T является типом Integer-идентификатор T (владелец места) заменяется типом Integer.

Таким образом, val внутри BoxPrinter становится Integer, потому что T заменяется на Integer.

Пояснение

3. Вот еще, где используется T:

```
public BoxPrinter(T arg) {
    val = arg;
}
```

Подобно объявлению val с типом T, вы говорите, что аргумент для конструктора BoxPrinter имеет тип T.

Позже в методе main (), когда конструктор вызывается в new, вы указываете, что T имеет тип Integer:

```
new BoxPrinter<Integer>(new Integer(10));
```

- Теперь, внутри конструктора BoxPrinter, arg и val должны быть одного типа, так как оба имеют тип T. Например следующее изменение конструктора:
- new BoxPrinter<String>(new Integer(10));
  приведёт к ошибке компиляции.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Пример

```
class Pair<T1, T2> {
    T1 object1;
    T2 object2;
    Pair(T1 one, T2 two) {
        object1 = one;
        object2 = two;
    }
    public T1 getFirst() {
        return object1;
    }
    public T2 getSecond() {
        return object2;
    }
}
```

```
Pair<Integer, String> worldCup =
        new Pair<Integer, String>(2018, "Kazakhstan");
System.out.println("World Chess Championship" +
worldCup.getFirst() +
        " in " + worldCup.getSecond());
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Алмазный синтаксис

Чтобы упростить жизнь программистам в Java 7 был введён алмазный синтаксис (diamond syntax), в котором можно опустить параметры типа. Т.е. можно предоставить компилятору определение типов при создании объекта. Вид упрощённого объявления:

```
Pair<Integer, String> worldCup =
        new Pair<>(2018, "Kazakhstan ");
```

Следует обратить внимание, что возможны ошибки связанные с отсутствием "<>" при использовании алмазного синтаксиса

Pair<Integer, String> pair = **new** Pair(6, " Apr");

ArrayList

- Эта лекция охватывает только один класс из API Java Collection: ArrayList. Остальные классы из API Java Collection рассмотрим позже.
- Одной из причин такого внимания к этому классу является то, как часто этот класс используется всеми Java-программистами.
- **ArrayList** является одним из наиболее широко используемых классов из Collections. Он предлагает наилучшую комбинацию возможностей, предоставляемых массивом и списком данных.
- Наиболее часто используемые операции со списком: • **добавлять** (**add)** элементы в список, **изменять** (**modify)** элементы в списке, **удалять** (**delete)** элементы из списка и **перебирать** (**iterate)** элементы

Использование ArrayList

- Один из часто задаваемых вопросов разработчиками Java: «Зачем мне заморачиваться с ArrayList, когда я уже могу хранить объекты такого же типа в массиве?»
- Ответ заключается в простоте использования ArrayList.
- Вы можете сравнить ArrayList с изменяемым размером. Как вы знаете, как только массив будет создан, вы не сможете увеличить или уменьшить размер массива.
- С другой стороны, ArrayList автоматически увеличивается и уменьшается по мере того, как элементы добавляются или удаляются из него.
- Кроме того, в отличие от массивов, вам не нужно указывать начальный размер для создания ArrayList.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Важные свойства ArrayList

- Он реализует интерфейс ***List***.
- Он позволяет добавлять к нему ***null***.
- Он реализует все операции со списками (добавляет, изменяет и удаляет значения).
- Он позволяет добавлять к нему дублирующиеся значения.
- Он поддерживает свой порядок вставки.
- Вы можете использовать либо **Iterator**, либо **ListIterator** для перебора элементов ArrayList.
- Он поддерживает generics, делая использование типов безопасным. (Вы должны объявить тип элементов, которые должны быть добавлены в ArrayList при объявлении.)

Создание ArrayList

Создание :

```
ArrayList<String> myArrList =
        new ArrayList<String>();
```

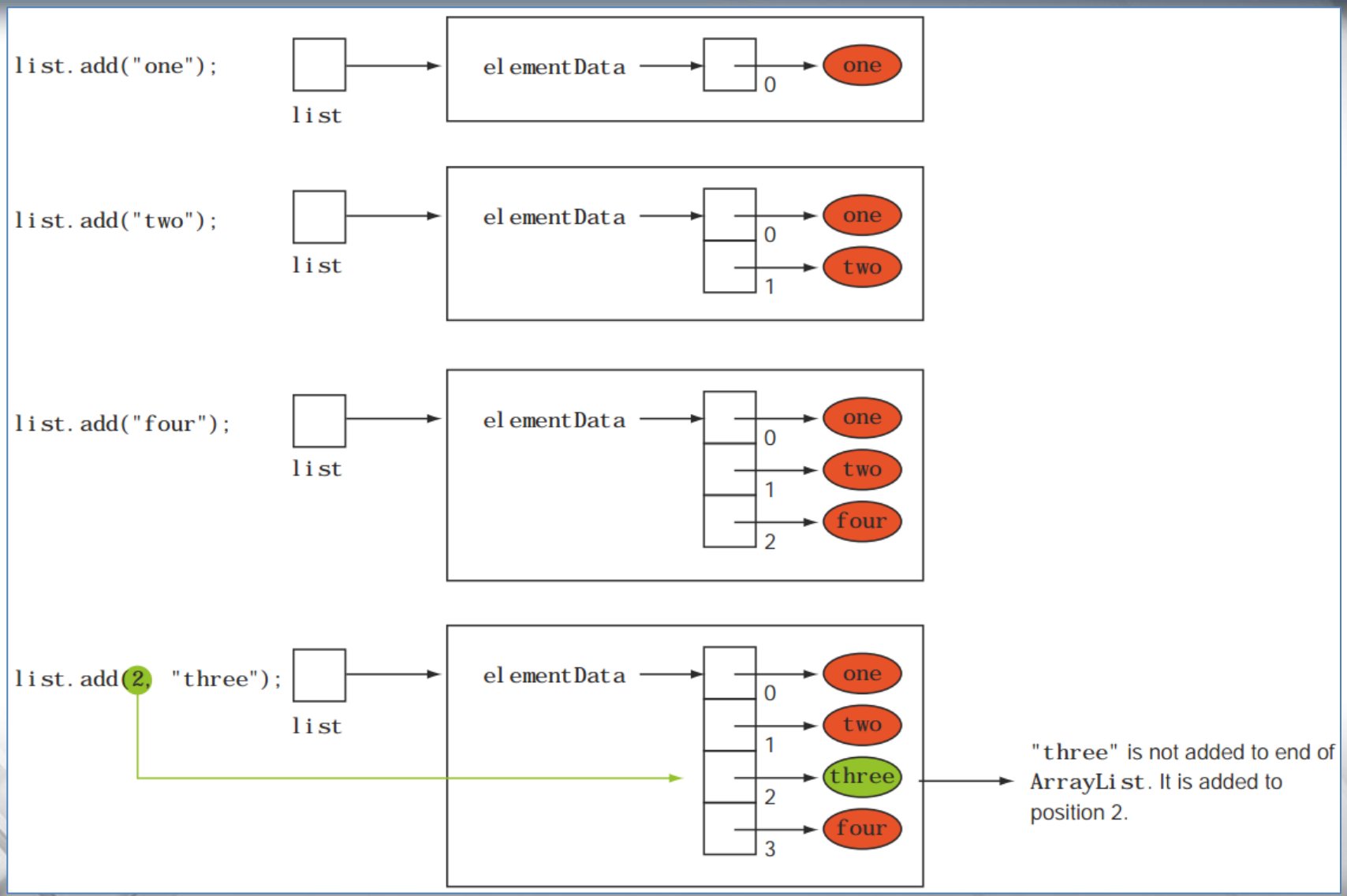
Начиная с версии Java версии 7, вы можете опустить тип объекта в правой части и создать ArrayList следующим образом :

```
ArrayList<String> myArrList = new ArrayList<>();
```

Добавление элементов к ArrayList

```
ArrayList<String> list = new ArrayList<>();
list.add("one");
list.add("two");
list.add("four");
list.add(2, "three");
```

Добавление элементов к ArrayList



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Доступ к элементам ArrayList

Использование расширенного цикла :

```
ArrayList<String> myArrList = new ArrayList<>();
myArrList.add("One");
myArrList.add("Two");
myArrList.add("Four");
myArrList.add(2, "Three");
for (String element : myArrList) {
    System.out.println(element);
}
```

Использование ListIterator:

```
ListIterator<String> iterator =
        myArrList.listIterator();
while (iterator.hasNext()) {
    System.out.println(iterator.next());
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Изменение элементов ArrayList

```
ArrayList<String> myArrList = new ArrayList<>();
myArrList.add("One");
myArrList.add("Two");
myArrList.add("Three");
myArrList.set(1, "One and Half");
for (String element:myArrList) {
    System.out.println(element);
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Удаление элементов ArrayList

```
ArrayList<String> myArrList = new ArrayList<>();
String s1 = "One";
String s2 = "Two";
String s3 = "Three";
String s4 = "Four";
myArrList.add(s1);
myArrList.add(s2);
myArrList.add(s3);
myArrList.add(s4);
myArrList.remove(1);
for (String element:myArrList) {
    System.out.println(element);
}
myArrList.remove(s3);
myArrList.remove("Four");
System.out.println();
for (String element : myArrList) {
    System.out.println(element);
}
```

Другие методы ArrayList

Добавление сложных (множественных) объектов в ArrayList

Вы можете добавить несколько элементов в ArrayList из другого ArrayList или любого другого класса, который является подклассом Collection, используя следующие перегруженные версии метода addAll :

- `addAll(Collection<? extends E> c)`
- `addAll(int index, Collection<? extends E> c)`

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Другие методы ArrayList

Добавление сложных (множественных) объектов в ArrayList

```
ArrayList<String> myArrList = new ArrayList<>();
myArrList.add("One");
myArrList.add("Two");
ArrayList<String> yourArrList = new ArrayList<>();
yourArrList.add("Three");
yourArrList.add("Four");
myArrList.addAll(1, yourArrList);
for (String val : myArrList) {
    System.out.println(val);
}
```

Другие методы ArrayList

Очистка от элементов ARRAYLIST

Вы можете удалить все элементы ArrayList, вызвав `clear`

```
ArrayList<String> myArrList = new ArrayList<>();
myArrList.add("One");
myArrList.add("Two");
myArrList.clear();
for (String val:myArrList) {
    System.out.println(val);
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Другие методы ArrayList

Поэлементный доступ к ARRAYLIST

- get(int index) — -возвращает элемент по указанной индексу в этом списке .
- size() — возвращает количество элементов в этом списке.
- contains(Object o) — возвращает true, если этот список содержит указанный элемент.
- indexOf(Object o) - возвращает индекс первого вхождения указанного элемента в этот список или -1, если этот список не содержит элемент.
- lastIndexOf(Object o) — возвращает индекс последнего появление указанного элемента в этом списке или -1, если этот список не содержит элемент.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Другие методы ArrayList

Поэлементный доступ к ARRAYLIST

- Три метода : **contains**, **indexOf**, и **lastIndexOf**— требуют от вас однозначного и глубокого понимания механизма сравнения объектов.
- ArrayList хранит объекты, и эти три метода будут сравнивать значения, которые вы передаете этим методам, со всеми элементами массива ArrayList

Сравнение объектов

- По умолчанию объекты считаются равными, если они имеют один и тот же адрес (класс String является исключением с пулом объектов String).
- Если вы хотите сравнивать объекты по их состоянию (значения переменной экземпляра), переопределите метод equals в этом классе.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Сравнение объектов

Метод equals реализует отношение эквивалентности для ненулевых ссылок на объекты:

Симметричность: для любых двух объектов x и y, x.equals(y) должен вернуть *true,* тогда и только тогда, когда y.equals(x) возвращает *true*.

Рефлексивность: для любых не-null объектов x, x.equals(x) должен возвращать *true*

Транзитивность: Для любых ненулевых объектов x, y, и z, если x.equals(y) и y.equals(z) возвращает *true*, то тогда x.equals(z) должен вернуть *true.*

Постоянство: для любых объектов *x* и *y* x.equals(y) возвращает одно и тоже, если информация, используемая в сравнениях, не меняется;

Для любых не-null объектов x, x.equals(null) должно возвращать false.

Другие методы ArrayList

СОЗДАНИЕ МАССИВА ИЗ ARRAYLIST

- Вы можете использовать метод toArray для получения массива, содержащего все элементы в ArrayList, в последовательности от первого до последнего элемента.
- Метод toArray создает новый массив, копирует на него элементы ArrayList и затем возвращает его.