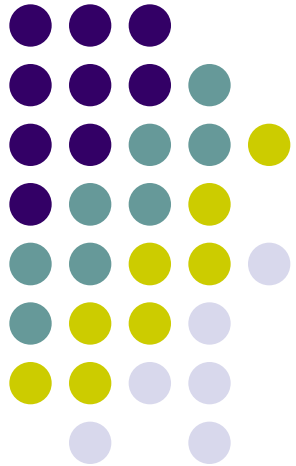
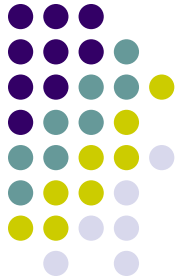
Коллекции в Java

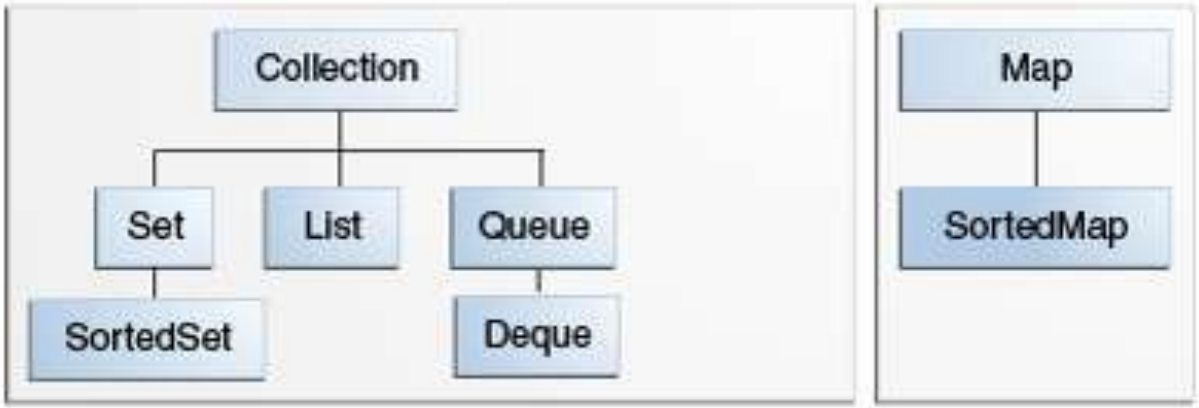


Java 1



- Классы
  - Vector, Stack, Hashtable, BitSet
- Интерфейс
  - Enumeration

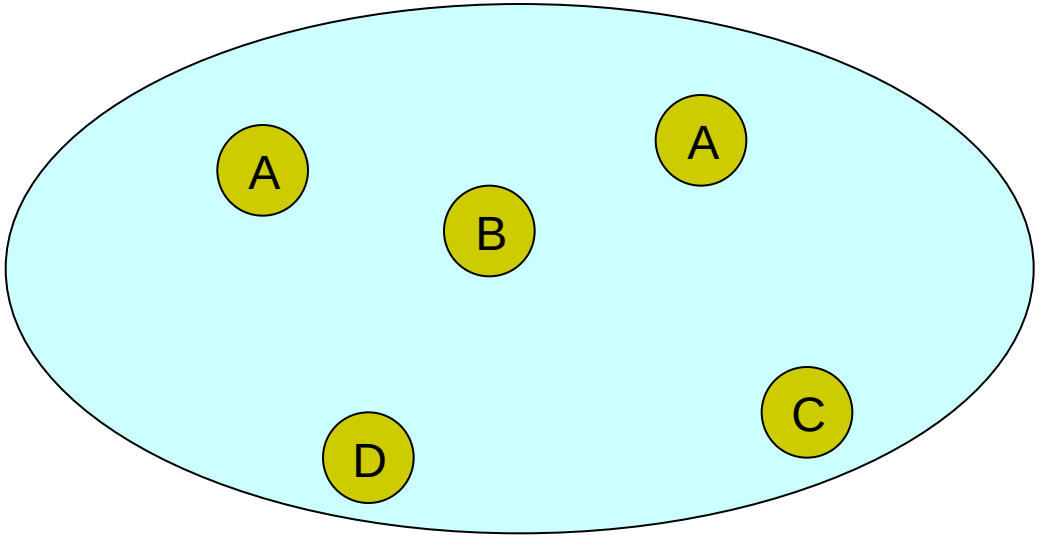
Начиная с Java 1.2



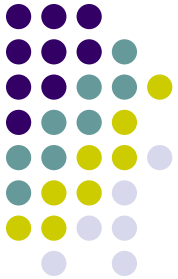
Базовые интерфейсы коллекций в пакете java.util.*

Коллекции

- Коллекция — неупорядоченный набор элементов
- Интерфейс Collection <E>

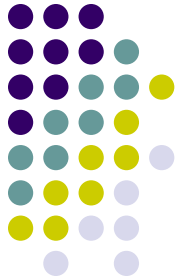


Интерфейс Collection <E> некоторые методы



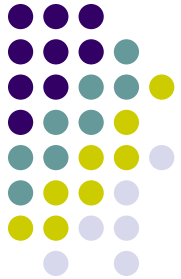
`boolean add(E element)`	Добавляет элемент в набор данных. Возвращает `false`, если не может добавить аргумент.
`void clear( )`	Удаляет все элементы.
`boolean contains(Object obj)`	`true`, если набор данных содержит элемент `obj`.
`boolean isEmpty( )`	`true`, если набор данных не имеет элементов.
`Iterator<E> iterator( )`	Возвращает итератор, используемый для обращения к элементам.
`boolean remove(Object obj)`	Если аргумент присутствует в наборе данных, один экземпляр этого элемента будет удален. Возвращает `true`, если произошло удаление.
`int size( )`	Возвращает текущее количество элементов в наборе данных.
`Object[] toArray( )`	Возвращает массив, содержащий все элементы.

Интерфейс Collection<E>



- Немодифицирующие операции
  - size()
  - isEmpty()
  - contains(Object o)
  - containsAll(Collection<?> c)
  - equals (Object o)
  - iterator()
  - toArray()
  - toArray(E[ ])

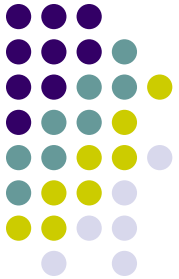
Интерфейс Collection<E>



- Модифицирующие операции
  - add(E e)
  - addAll(Collection <? Extends E> c)
  - remove(Object e)
  - removeAll (Collection <?> c)
  - retainAll (Collection <?> c)
  - clear()

✓ Исключения
  - UnsupportedOperationException

Перебор элементов коллекции

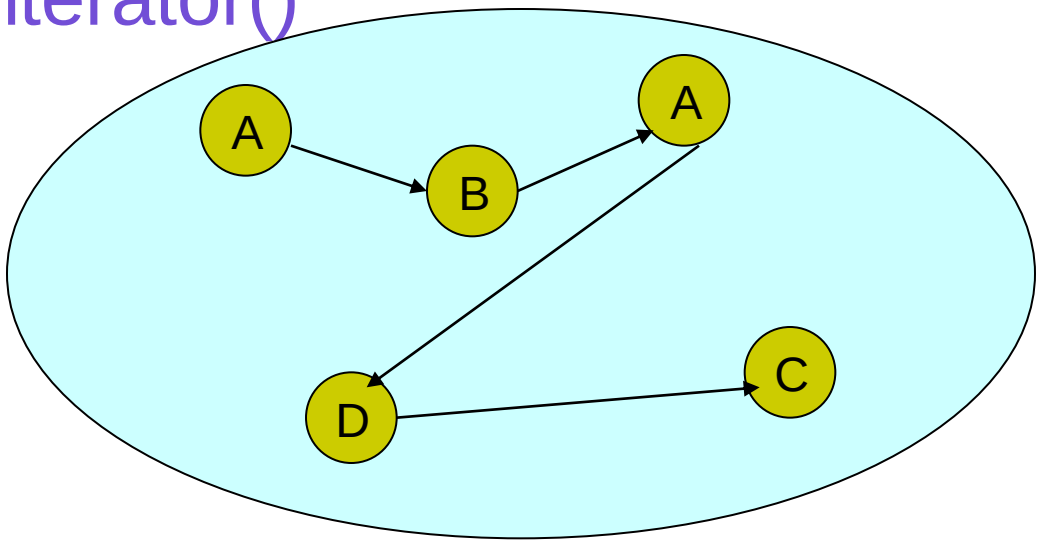


- Итератор
- Цикл for-each

Итераторы

- Итератор — для обхода коллекции
- Интерфейс Iterator <E>
- Метод коллекции

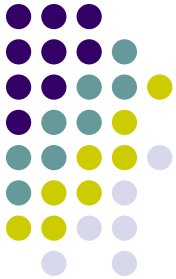
Iterator<E> iterator()



Интерфейс Iterator
java.util.Iterator<E>

boolean hasNext()	Возвращает `true`, если существует следующий элемент, к которому можно обратиться.
E next()	Возвращает следующий элемент. Генерируется исключение **`NoSuchElementException`**, если достигнут конец контейнера.
void remove()	Удаляет последний просмотренный элемент. Этот метод должен вызываться сразу после метода `next()`. Если этого не сделать генерируется исключение **`IllegalStateException.`** Если после чтения элемента набор данных изменился, данный метод генерирует исключение **`ConcurrentModificationException`**

Пример использования



```
import  java.util.*;
public class SimpleCollection {
  public static void main(String[ ] args) {
//Используем интерфейс, т.к. просто будем
// работать с особенностями Collection
Collection<String> c = new ArrayList<String>();
for(int i = 0; i < 10; i++)
    c.add(Integer.toString(i));
Iterator<String> it = c.iterator();
while(it.hasNext())
      System.out.println(it.next());
}
}
```

В JDK 5.0 цикл “for each”

```
for (String el : c){
 System.out.println(el);
}
```

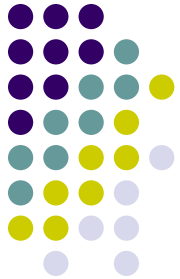
Компилятор преобразует такой цикл в цикл с итератором

Особенности использования метода remove()

```
Iterator<String> it = c.iterator();
it.next(); //пропускаем первый элемент
it.remove(); //удаляем его
```

При вызове метода next() итератор “перескакивает” через следующий элемент и возвращает ссылку на него

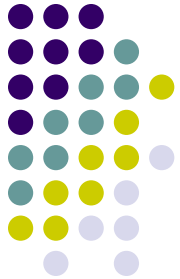
Интерфейс List<E>



- Добавляет методы, использующие позицию элемента в коллекции (индекс)

```
E          get(int index)
void       add(int index, E element)
E          set(int index, E element)
boolean    addAll(int index, Collection<? extends E> c)
E          remove(int index)
```

Интерфейс List<E>



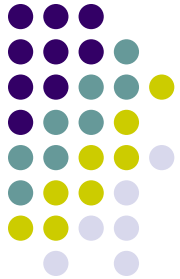
- Поиск возвращает позицию

```
int         indexOf(Object o)
int         lastIndexOf(Object o)
```

- Выделение подсписка

```
List<E>     subList(int fromIndex, int toIndex)
```

Интерфейс List<E>



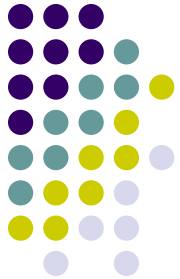
- Добавлены методы, возвращающие итератор с интерфейсом ListIterator<E>

ListIterator<E> listIterator()

ListIterator<E> listIterator(int index)

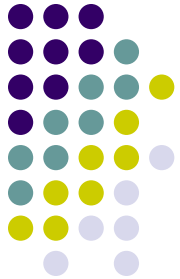
- ListIterator<E> позволяет навигацию в двух направлениях и изменение элемента по позиции итератора

Implementations

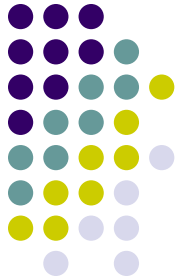


Interfaces	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Deque		ArrayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

ArrayList <E> и LinkedList<E>



- ArrayList <E>
  - имеет начальную емкость, которая может увеличиваться
  - добавление – в конец и по позиции
  - удаление по позиции, по значению и через итератор
- LinkedList <E>
  - добавление/удаление в конце и в начале
  - есть обратный итератор
  - есть операции, чтобы работать как со стеком/деком



CopyOnWriteArrayList <E>

List<String> list = Arrays.asList(new String[size]);

Класс AbstractCollection<E>

- Абстрактный
- Позволяет быстро реализовывать коллекции
- Реализация неизменяемых коллекций (абстрактные методы – реализация обязательно)
  - iterator()
  - size()
- Реализация изменяемых коллекций - дополнительно
  - add(Object o)
  - iterator.remove()

Устаревшие коллекции

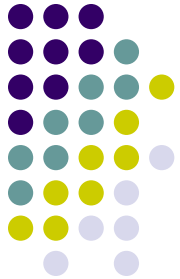
Устаревшие коллекции являются синхронизированными

- Vector <E>
- Stack <E>
- Dictionary <K,V>
- Hashtable <K,V>
- interface Enumeration <K>

Устаревшие коллекции

Класс BitSet предназначен для работы с последовательностями битов. Каждый компонент этой коллекции может принимать булево значение, которое обозначает, установлен бит или нет. Содержимое BitSet может быть модифицировано содержимым другого BitSet с использованием операций AND, OR или XOR

Класс Properties



Класс Properties - предназначен для хранения множества свойств (параметров).

Это карты особого вида:

- ключ и значение являются строками
- карту можно сохранить в файле и загрузить из файла
- для используемых по умолчанию значений создается вторая карта