Параметризация типов в Java

Непараметризованный класс:

```
public class Box
{
        private Object object;
        public void add(Object object)
        { this.object = object; }
        public Object get()
        { return object;  }
}
```

Параметризация типов в Java

Ошибка, связанная с отсутствием параметризации:

```
public class BoxDemo
{public static void main(String[ ] args)
    {  Box integerBox = new Box();
       …
       integerBox.add("10");
       ...
       Integer someInteger = (Integer)integerBox.get();
       System.out.println(someInteger); }
}
```

Параметризация типов в Java

```
Exception in thread "main"
        java.lang.ClassCastException:
        java.lang.String cannot be cast to java.lang.Integer
          at BoxDemo.main(BoxDemo.java:6)
```

Параметризация типов в Java

Параметризованный класс (generic type):

```
public class Box<T>
{  private T t; // T stands for "Type"
    public void add(T t)
    {   this.t = t; }
    public T get()
    {  return t;   }
}
```

<T> - type variable, formal type parameter

Параметризация типов в Java

Создание объекта параметризованного класса (generic type invocation):

Box<Integer> integerBox = new Box<Integer>();

```
public class BoxDemo
{public static void main(String[ ] args)
    {  Box<Integer> integerBox = new Box<Integer>();
       integerBox.add(new Integer(10));
       Integer someInteger = integerBox.get(); //no cast!
       integerBox.add("10"); }
}
```

Параметризация типов в Java

```
BoxDemo3.java:5: add(java.lang.Integer) in
Box<java.lang.Integer> cannot be applied to
(java.lang.String) integerBox.add("10");
                  ^
   1 error
```

Таким образом вместо исключения (runtime error) получаем compilation error !!!

Параметризация типов в Java

Параметризованный класс может иметь несколько параметров, но они должны обозначаться разными буквами.

class Box <T, T> - error!

class Box <T,U> - ok!

Параметризация типов в Java

Параметризованные методы

```
public class Box<T>
{  private T t;
   public void add(T t)
    {   this.t = t; }
   public T get()
    {  return t;   }
   public <U> void inspect(U u)
   {System.out.println("T: " + t.getClass().getName());
    System.out.println("U: " + u.getClass().getName()); }
}
```

Параметризация типов в Java

Параметризованные методы

```
public static void main(String[ ] args)
{     Box<Integer> integerBox = new Box<Integer>();
      integerBox.add(new Integer(10));
      integerBox.inspect("some text");
 }
```

Вывод:

T: java.lang.Integer

U: java.lang.String

Параметризация типов в Java

Ограниченная параметризация (bounded type parameters)

```
public <U extends Number> void inspect(U u)
    {System.out.println("T: " + t.getClass().getName());
     System.out.println("U: " + u.getClass().getName()); }
```

```
public static void main(String[ ] args)
{      Box<Integer> integerBox = new Box<Integer>();
       integerBox.add(new Integer(10));
       integerBox.inspect("some text");   //error!!!
 }
```

Параметризация типов в Java

Ограниченная параметризация (bounded type parameters)

Extends в данном случае понимается и как extends и как implements:

```
public <U extends Number & MyInterface> void inspect(U u)
    {System.out.println("T: " + t.getClass().getName());
     System.out.println("U: " + u.getClass().getName()); }
```

Параметризация типов в Java

Подтипизация

```
Box<Number> box = new Box<Number>();
box.add(new Integer(10)); // OK
box.add(new Double(10.1)); // OK
…
public void boxTest (Box<Number> n)
{
    …
}
```

Параметризация типов в Java

Подтипизация

```
Box<Integer> IntegerBox = new Box<Integer>();
Box<Double> DoubleBox = new Box<Double>();
boxTest(IntegerBox );      //error!
boxTest(DoubleBox );     //error!
```

Причина ошибки - Box<Integer> и Box<Double> не являются подтипами Box<Number> !!!

Параметризация типов в Java

Wildcards

```
Box<? extends Number> someBox = ...;   //upper bound
Box<? super Number> someBox = ...;      //lower bound
Box<?> someBox = ...;                   //unbounded wildcard
Box<? extends Object> someBox = ...;    //unbounded wildcard
```

Box<Integer> и Box<Double> не являются подтипами

Box<Number>, но являются подтипами Box<? extends Number>

Параметризация типов в Java

Wildcards

```
Box<? extends Number> someBox = new Box<Number>();
Box<Integer> IntegerBox = new Box<Integer>();
Box<Double> DoubleBox = new Box<Double>();
someBox = IntegerBox;
someBox = DoubleBox;
```

Параметризация типов в Java

Wildcards

```
public void boxTest (Box<? extends Number> n)
{
    …
}
Box<Integer> IntegerBox = new Box<Integer>();
Box<Double> DoubleBox = new Box<Double>();
boxTest(IntegerBox );      //ok!
boxTest(DoubleBox );     //ok!
```

Параметризация типов в Java

Очистка типа (Type erasure)

На этапе компиляции вся информация о параметризованных типах удаляется.

Это позволяет сохранить совместимость на уровне байт-кода с обычными типами.

Следствие: нельзя получить информацию о параметризованном типе во время выполнения.

Параметризация типов в Java

Очистка типа (Type erasure)

```
public class MyClass<E> {
    public void myMethod(Object item) {

      if (item instanceof E)  //Compiler error
     { ...   }
      E item2 = new E();      //Compiler error
      E[ ] iArray = new E[10];    //Compiler error
      E obj = (E)new Object();   //Unchecked cast warning  }
}
```

Параметризация типов в Java

Очистка типа (Type erasure)

```
public class WarningDemo
{
    public static void main(String[ ] args)
     { Box<Integer> bi;
        bi = createBox();  }
     static Box createBox()
     { return new Box(); }
}
```

Параметризация типов в Java

Очистка типа (Type erasure)

```
WarningDemo.java:4: warning: [unchecked] unchecked
    conversion
found   : Box
required: Box<java.lang.Integer>
        bi = createBox();
                     ^
1 warning
```

Библиотека классов Java

Коллекции. Структура коллекций

Коллекция – это объект-контейнер, включающий группу, как правило, однотипных объектов. Структура коллекций (collections framework) Java стандартизирует способ, с помощью которого ваши программы хранят и обрабатывают группы объектов.

Библиотека классов Java

Коллекции. Структура коллекций

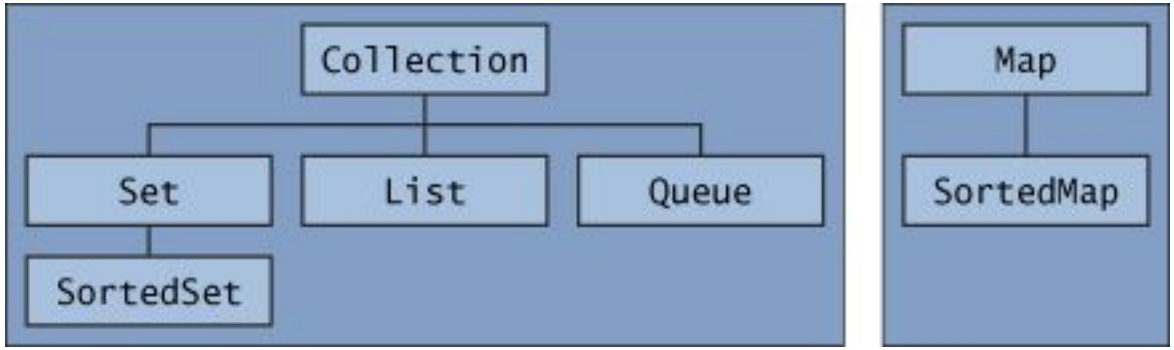
Преимущества использования структуры коллекций:

1. **Избавление от рутинных операций по кодированию стандартных структур данных и алгоритмов**
2. **Высокая эффективность реализации**
3. **Универсальность и простота изучения(различные типы коллекций работают похожим друг на друга образом и с высокой степенью способности к взаимодействию)**
4. **Расширяемость**

Структура коллекций находится в пакете java.util.*

Библиотека классов Java

Коллекции. Интерфейсы



Все коллекции в Java являются параметризованными

public interface Collection<E>...

Библиотека классов Java

Коллекции. Интерфейс Collection

Корень иерархии. Задает самые общие методы для работы с коллекциями.

```
public interface Collection<E> extends Iterable<E>
{   int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(E element);
    boolean remove(Object element);
    Iterator<E> iterator();
```

Библиотека классов Java

Коллекции. Интерфейс Collection

```
    …
    boolean containsAll(Collection<?> c);
    boolean addAll(Collection<? extends E> c);
    boolean removeAll(Collection<?> c);
    boolean retainAll(Collection<?> c);
    void clear();
   Object[ ] toArray();
}
```

Библиотека классов Java

Коллекции. Перемещение по коллекции

1. For-each

```
for (Object o : collection)
  System.out.println(o);
```

2. public interface Iterator<E>

```
{
  boolean hasNext();
       E next();
      void remove();
}
```

Библиотека классов Java

Коллекции. Перемещение по коллекции

```
Collection <String> cs = new ArrayList<String>();
cs.add("1");
cs.add("2");
cs.add("3");
for (String str : cs)
    System.out.println(str);
```

Библиотека классов Java

Коллекции. Перемещение по коллекции

```
Collection <String> cs = new ArrayList<String>();
cs.add("1");
cs.add("2");
cs.add("3");
Iterator it = cs.iterator();
while(it.hasNext())
    System.out.println(it.next());
```

Библиотека классов Java

Коллекции. Перемещение по коллекции

Метод remove() может быть вызван только один раз после вызова методы next(), иначе бросается исключение.

Метод remove() единственный безопасный способ модификации коллекции.

```
static void filter(Collection<?> c) {
    for (Iterator<?> it = c.iterator(); it.hasNext(); )
        if (!cond(it.next()))
            it.remove();
}
```

Библиотека классов Java

Коллекции. Интерфейс Set

Set – коллекция без повторяющихся элементов (математическое множество). Методы совпадают с Collection но add() вернет false, если элемент уже есть в коллекции.

Библиотека классов Java

Коллекции. Интерфейс Set

```
import java.util.*;
public class FindDups {
    public static void main(String[ ] args) {
        Set<String> s = new HashSet<String>();
        for (String a : args)
            if (!s.add(a))
                System.out.println("Duplicate detected: " + a);
        System.out.println(s.size() + " distinct words: " + s);
    }
}
```

Библиотека классов Java

Коллекции. Интерфейс SortedSet

Интерфейс SortedSet из пакета java.util, расширяющий интерфейс Set, описывает упорядоченное множество, отсортированное по естественному порядку возрастания его элементов или по порядку, заданному реализацией интерфейса Comparator. Элементы не нумеруются, но есть понятие первого, последнего, большего и меньшего элемента.

Библиотека классов Java

Коллекции. Интерфейс SortedSet

- Comparator Comparator() — возвращает способ упорядочения коллекции;
- Object first () — возвращает первый, меньший элемент коллекции;
- Object last() — возвращает последний, больший элемент коллекции;

Библиотека классов Java

Коллекции. Интерфейс SortedSet

- SortedSet headSet (Object toElement) — возвращает начальные, меньшие элементы до элемента toElement исключительно;
- SortedSet subSet(Object fromElement, Object toElement) — возвращает подмножество коллекции от элемента fromElement включительно до элемента toElement исключительно;
- SortedSet tailSet (Object fromElement) — возвращает последние, большие элементы коллекции от элемента fromElement включительно.

Библиотека классов Java

Коллекции. Интерфейс Comparator

- int compare (Object obj1, object obj2) — возвращает отрицательное число, если obj1 в каком-то смысле меньше obj2; нуль, если они считаются равными; положительное число, если objl больше obj2. С точки зрения теории множеств можно сказать, что этот метод сравнения обладает свойствами тождества, антисимметричности и транзитивности;
- boolean equals (Object obj) — сравнивает данный объект с объектом obj, возвращая true, если объекты совпадают в каком-либо смысле, заданном этим методом.

Библиотека классов Java

Коллекции. Интерфейс Comparator

```
class ComplexCompare implements Comparator
{public int compare(Object obj1, Object obj2)
{Complex z1 = (Complex)obj1, z2 = (Complex)obj2;
 double re1 = z1.getRe(), im1 = z1.getIm();
 double re2 = z2.getRe(), im2 = z2.getIm();
 if (re1 != re2) return (int)(re1 - re2);
 else if (im1 != im2) return (int)(im1 — im2) ;
 else return 0;}
public boolean equals(Object z)
{return compare (this, z) ==  0;}
}
```

Библиотека классов Java

Коллекции. Интерфейс Comparator

```
TreeSet ts = new TreeSet(new ComplexCompare());
ts.add(new Complex(1.2, 3.4));
ts.add(new Complex(-1.25, 33.4));
ts.add(new Complex(1.23, -3.45));
ts.add(new Complex(16.2, 23.4));

Iterator it = ts.iterator();
while(it.hasNext())
    ((Complex)it.next()).print();
```

Библиотека классов Java

Коллекции. Интерфейс List

Интерфейс List из пакета java.util, расширяющий интерфейс Collection, описывает методы работы с упорядоченными коллекциями. Иногда их называют последовательностями (sequence). Элементы такой коллекции пронумерованы, начиная от нуля, к ним можно обратиться по индексу. В отличие от коллекции Set элементы коллекции List могут повторяться.

Библиотека классов Java

Коллекции. Интерфейс List

- void add(int index, Object obj) — вставляет элемент obj в позицию index; старые элементы, начиная с позиции index, сдвигаются, их индексы увеличиваются на единицу;
- boolean addAll(int index, Collection coll) — вставляет все элементы коллекции coll;
- Object get(int index) — возвращает элемент, находящийся в позиции index;
- int indexOf (Object obj) — возвращает индекс первого появления элемента obj в коллекции;

Библиотека классов Java

Коллекции. Интерфейс List

- int lastIndexOf (Object obj) — возвращает индекс последнего появления элемента obj в коллекции;
- ListIterator listIterator() — возвращает итератор коллекции;
- ListIterator listIterator (int index) — возвращает итератор конца коллекции от позиции index;
- Object set (int index, Object obj) — заменяет элемент, находящийся в позиции index, элементом obj;
- List subList(int from, int to) — возвращает часть коллекции от позиции from включительно до позиции to исключительно.

Библиотека классов Java

Коллекции. Интерфейс ListIterator

- void add(Object element) — добавляет элемент element перед текущим элементом;
- boolean hasPrevious() — возвращает true, если в коллекции есть элементы, стоящие перед текущим элементом;
- int nextIndex() — возвращает индекс текущего элемента; если текущим является последний элемент коллекции, возвращает размер коллекции;

Библиотека классов Java

Коллекции. Интерфейс ListIterator

- Object previous() — возвращает предыдущий элемент и делает его текущим;
- int previousIndex() — возвращает индекс предыдущего элемента;
- void set(Object element) — заменяет текущий элемент элементом element; выполняется сразу после next() или previous().

Библиотека классов Java

Коллекции. Интерфейс Map

Интерфейс Map из пакета java.util описывает коллекцию, состоящую из пар "ключ — значение". У каждого ключа только одно значение, что соответствует математическому понятию однозначной функции или отображения. Такую коллекцию часто называют еще словарем (dictionary) или ассоциативным массивом (associative array).

Библиотека классов Java

Коллекции. Интерфейс Map

- boolean containsKey (Object key) — Проверяет наличие ключа key;
- boolean containsValue (Object value) — Проверяет наличие значения value;
- Set entrySet() — представляет коллекцию в виде множества, каждый элемент которого — пара из данного отображения, с которой можно работать методами вложенного интерфейса Map.Entry;
- Object get(Object key) — возвращает значение, отвечающее ключу key;

Библиотека классов Java

Коллекции. Интерфейс Map

- Set keySet() — представляет ключи коллекции в виде множества;
- Object put (Object key, Object value) — добавляет пару "key— value", если такой пары не было, и заменяет значение ключа key, если такой ключ уже есть в коллекции;
- void putAll (Map m) — добавляет к коллекции все пары из отображения m;
- Collection values() — представляет все значения в виде коллекции.

Библиотека классов Java

Коллекции. Интерфейс Map.Entry

- методы getKey() и getValue() позволяют получить ключ и значение пары;
- метод setValue (Object value) меняет значение в данной паре.

```
for (Iterator it=map.entrySet().iterator(); it.hasNext(); )
{
        Map.Entry entry = (Map.Entry)it.next();
        Object key = entry.getKey();
        Object value = entry.getValue();
}
```

Библиотека классов Java

Коллекции. Интерфейс SortedMap

Интерфейс SortedMap, расширяющий интерфейс Map, описывает упорядоченную по ключам коллекцию Map. Сортировка производится либо в естественном порядке возрастания ключей, либо в порядке, описываемом в интерфейсе Comparator.

Библиотека классов Java

Коллекции. Интерфейс SortedMap

- Comparator comparator() — возвращает способ упорядочения коллекции;
- Object firstKey() — возвращает первый, меньший элемент коллекции;
- SortedMap headMap (Object toKey) — Возвращает начало коллекции до элемента с ключом toKey исключительно;
- Object lastKey() — возвращает последний, больший ключ коллекции;

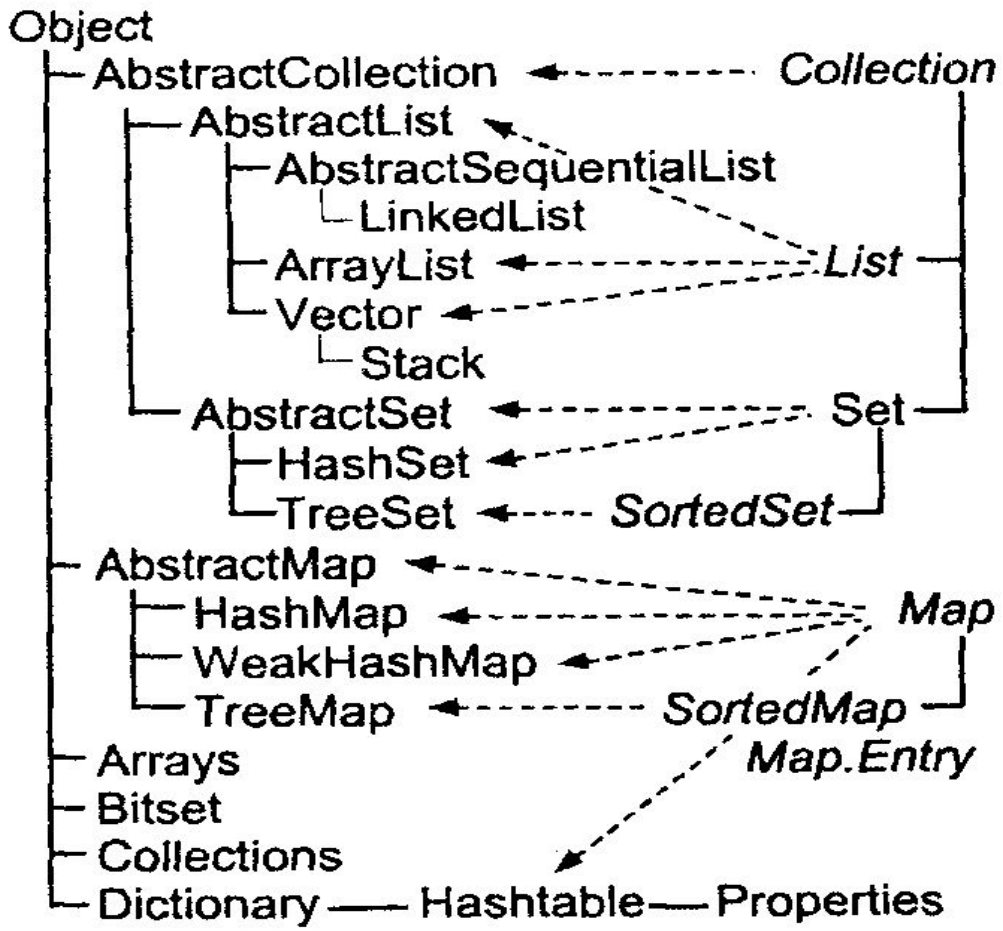
Библиотека классов Java

Коллекции. Интерфейс SortedMap

- SortedMap subMap (Object fromKey, Object toKey) — возвращает часть коллекции от элемента с ключом fromKey включительно до элемента с ключом toKey исключительно;
- SortedMap tailMap(Object fromKey) — возвращает остаток коллекции от элемента fromKey включительно.

Библиотека классов Java

Коллекции. Реализации



Библиотека классов Java

Коллекции. Реализации

General-purpose Implementations					
Interfaces	Implementations				
	Hashtable	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		Linked List	
Map	HashMap		TreeMap		LinkedHashMap

Библиотека классов Java

Коллекции. Алгоритмы сортировки

Сортировка может быть сделана только в упорядочиваемой коллекции, реализующей интерфейс List. Методы:

- static void sort (List coll) — сортирует в естественном порядке возрастания коллекцию coll, реализующую интерфейс List;
- static void sort (List coll, Comparator c) — сортирует коллекцию coll в порядке, заданном объектом c.

Библиотека классов Java

Коллекции. Алгоритмы сортировки

Сортировка является быстрой и стабильной:

1) гарантирована скорость не ниже n*log(n)
2) равные элементы не переупорядочиваются

```
public class Sort {
    public static void main(String[ ] args) {
        List<String> list = Arrays.asList(args);
        Collections.sort(list);
        System.out.println(list);
    }
}
```

Библиотека классов Java

Коллекции. Алгоритмы поиска

- static int binarySearch(List coll, Object element) — отыскивает элемент element в отсортированной в естественном порядке возрастания коллекции coll и возвращает индекс элемента или отрицательное число, если элемент не найден; отрицательное число показывает индекс, с которым элемент element был бы вставлен в коллекцию, с обратным знаком;
- static int binarySearch(List coll, Object element. Comparator c) — то же, но коллекция отсортирована в порядке, определенном объектом c.

Библиотека классов Java
Коллекции. Алгоритмы «перемешивания»

- static void shuffle (List coll) — случайные числа задаются по умолчанию;
- static void shuffle (List coll, Random r) — случайные числа определяются объектом r.

Библиотека классов Java
Коллекции. Алгоритмы манипуляции с данными

- static void reverse(List coll) меняет порядок расположения элементов на обратный.
- static void copy(List from, List to) копирует коллекцию from в коллекцию to.
- static void fill(List coll, Object element) заменяет все элементы существующей коллекции coll элементом element.
- static void swap(List coll, int i1, int i2) меняет местами элементы

Библиотека классов Java
Коллекции. Алгоритмы экстремумов

- static Object max (Collection coll) – возвращает наибольший в естественном порядке элемент коллекции coll;
- static Object max (Collection coll, Comparator c) — то же в порядке, заданном объектом c;
- static Object min (Collection coll) — возвращает наименьший в естественном порядке элемент коллекции coll;
- static Object min (Collection coil, Comparator c) — то же в порядке, заданном объектом c.

Библиотека классов Java
Коллекции. Алгоритмы объединения

- static int frequency(Collection coll, Object element) — считает кол-во появлений указанного элемента в коллекции
- static boolean disjoint(Collection coll1, Collections coll2) — определяет пересекаются ли две коллекции (возвращает true, если не пересекаются)