Параметризованные типы

Пример:

```
public int abs(int x) {return x>0?x:-x;}
```

Вопросы:

- Что делает этот метод?

- Является ли следующий код ошибочным:

```
double d; double m = abs(d);
```

Определение и свойства параметризованных типов

Параметризация позволяет создавать классы, интерфейсы и методы, в которых тип обрабатываемых данных задается как параметр.

Параметром может быть только ссылочный тип

Параметром может быть шаблон

Синтаксис параметров:

<Тип1 или Шаблон, Тип2 или Шаблон, ...>

Пример класса

```
public class Subject <T1, T2> {
   private T1 name;
   private T2 id;
   public Subject() {}
   public Subject(T1 names, T2 ids ) {
      id = ids;
      name = names;
   } }

Subject<String,Integer> sub =new Subject<String,Integer>();

char ch[] = {‘J’,’a’,’v’,’a’};
Subject<char[],Double> sub2 = new Subject<char[],Double>(ch, 71.0 );
```

Пример интерфейса

```
public interface Generator<T> { T next(); }
public class Coffee {}

public class CoffeeGenerator implements
                                        Generator<Coffee> {
  private Coffee[] types = {…}
  public Coffee next() {
      Random rand = new Random();
      return types[rand.nextInt(types.length)]
  } }
```

Пример библиотечных интерфейсов

```
Interface Comparable<T> {
    int compareTo(T o)
}
java.util.function
Interface Function<T,R> {
    R apply(T t)
}
```

Шаблоны типов

? extends Type : семейство наследников типа Type.

? super Type : семейство предков типа Type.

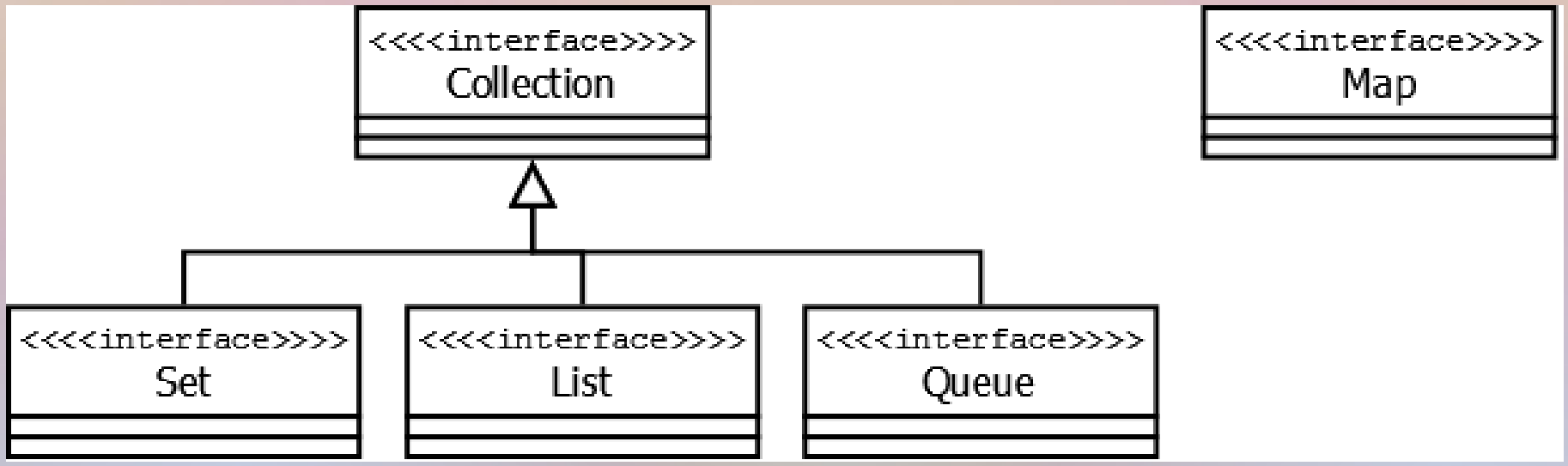
? : набор любых типов.

Пример:

```
public interface Collection<E> {
boolean   addAll (Collection<? extends E> c);
}
```

Контейнеры

- Массивы
- Коллекции
- Ассоциативные массивы



Коллекции

- Interface Iterator<E>
  - boolean hasNext()
  - E next()
  - void remove()
- Interface Collection<E>

Перебор элементов коллекции

```
ArrayList <MyClass> collection =  new
                                  ArrayList<MyClass>();
for (Iterator<MyClass>  it = collection.iterator();
                                  it.hasNext(); )
    System.out.println(it.next());
```

Перебор элементов коллекции в стиле foreach

```
for(MyClass e : collection)
    System.out.println(e);
```

Списки и динамические массивы

- Interface List<E>
- Interface ListIterator<E>
- Class LinkedList<E>
- Class ArrayList<E>

LinkedList

- Двусвязный список
- Быстрая вставка элемента
- Быстрое удаление элемента
- Медленный доступ к произвольному элементу

ArrayList

- Массив
- Медленная вставка элемента
- Медленное удаление элемента
- Быстрый доступ к произвольному элементу

Множества и упорядоченные множества

- Interface Set<E>
- Class HashSet<E>
- Interface SortedSet<E>
- Class TreeSet<E>

HashSet

- Представляет собой неупорядоченную коллекцию, которая не может содержать дублирующиеся данные. Является программной моделью математического понятия «множество».
- Для хранения объектов используется хеширование (хеш-таблицу)
- Объекты должны реализовывать методы hashCode() и equals()
- Быстрые добавление, удаление и проверка вхождения элемента в множество (не зависят от количества хранимых элементов)

Реализация методов equals() и hashCode()

- реализуются как переопределение унаследованных методов boolean equals(Object obj) и int hashCode() класса Object;
- если поля объектов равны, то и объекты равны;
- если два объекта тождественны в соответствии с методом equals(), они должны иметь одно значение hashCode() (обратное не верно);
- наличие грамотной реализации equals() и hashCode() объектом позволяет хранить и извлекать их с помощью коллекций на базе хеширования.

Хеш-код

- хеш-код — это целочисленный результат работы метода, которому в качестве входного параметра передан объект;
- для одного и того-же объекта, хеш-код всегда будет одинаковым;
- если объекты одинаковые, то и хеш-коды одинаковые (в Java по умолчанию не выполняется)
- если хеш-коды равны, то объекты не всегда равны (коллизия);
- если хеш-коды разные, то объекты гарантированно разные.

TreeSet

- Для хранения объектов используется сбалансированное дерево
- Объекты должны реализовывать интерфейс Comparable<E>
- Гарантируется обход итератором в порядке возрастания
- Добавление, удаление и проверка вхождения элемента в множество за время порядка log(N)

Упорядочение объектов

- Естественное упорядочение – реализация интерфейса Comparable<E>
  - int compareTo(E obj)
- Внешнее упорядочение – определении внешнего класса компаратора, реализующего интерфейс Comparator<E>
  - int compare(E o1, E o2)

Реализация алгоритмов обработки данных

Класс Collections содержит статические методы для работы с коллекциями данных:

- поиск элементов, максимальных, минимальных;
- сортировка;
- замена элементов.

Ассоциативные массивы

- Interface Map<K,V> (вместо Collection<E>)
- Хранение данных парами – (ключ, объект)
- Для обхода используются итераторы их ключей
- Class HashMap<K,V>
- Class TreeMap<K,V>