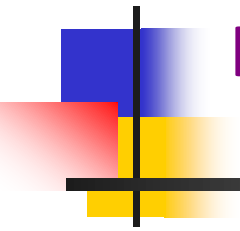
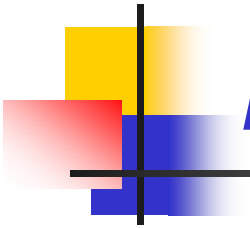


Generics

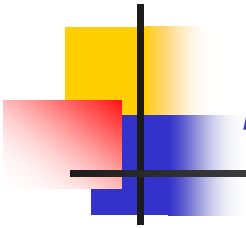
настраиваемые типы

JDK 1.5



Полезные ссылки

- http://www.quizful.net/post/java-generics-tutorial
- http://www.ibm.com/developerworks/ru/library/j-jtp04298/



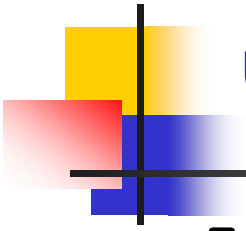
Как было

```
List myIntList = new LinkedList();
myIntList.add(new Integer(0));
myIntList.add(new Double(3.14));
….
Integer x = (Integer)
  myIntList.iterator().next();
```

Возможные проблемы

– *Runtime error*

– *Или выяснять тип объекта*



Что дает *generic*?

```
List<Integer> myIntList = new
  LinkedList<Integer>();
myIntList.add(new Integer(0));
. . .
Integer x =
  myIntList.iterator().next();
```

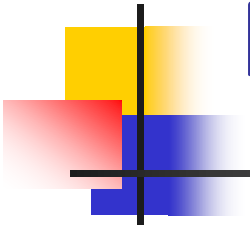
В случае неправильного использования

```
myIntList.add(new Double(3.14));
```

- Ошибка компиляции

Пример описания

```
class Gen<T> {
   T ob;
Gen(T o) {
     ob = o;
   }
```



Пример описания

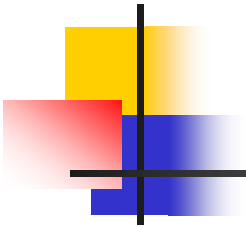
```
T getob() {
    return ob;
  }
void showType() {
 System.out.println("Type of T is "
  + ob.getClass().getName());
  }
}
```

Пример использования

```
class GenDemo {
  public static void main(String args[])
  {
     Gen<Integer> iOb;
     iOb = new Gen<Integer>(88);
     iOb.showType(); //!!!!!!!!!!
     int v = iOb.getob();
     System.out.println("value: " + v);
```

Пример использования

```
Gen<String> strOb =
        new Gen<String>("Generics Test");
        strOb.showType(); //!!!!!!!!!!
        String str = strOb.getob();
        System.out.println("value: " + str);
    }
}
```



Параметризация интерфейса

```
public interface List<E> {
void add(E x);
Iterator<E> iterator();
}
public interface Iterator<E> {
E next();
boolean hasNext();
}
```

E- формальный параметр типа

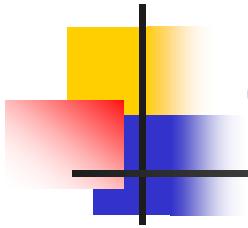
Общий вид объявления параметризованного класса

```
class   имя-класса  <список-параметров-типа>
  {//...
}
```

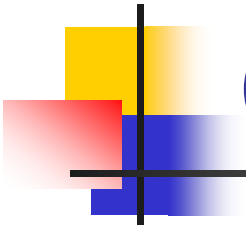
Объявление ссылки на объект параметризованного класса

```
class   имя-класса<список-аргументов-типа>
 имя-переменной =
   new  имя-класса <список-аргументов-типа>
   (...);
```

Ограниченные типы (bounded types)



```
class Stats<T> {
  T[] nums;
  Stats(T[] o) {
    nums = o;
  }
```



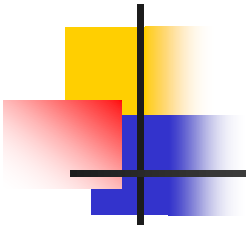
Ограниченные типы

```
double average() {
    double sum = 0.0;
for(int i=0; i < nums.length; i++)
     sum += nums[i].doubleValue();
   return sum / nums.length;
  }
}
```

Из-за стирания типа массив nums состоит из Object

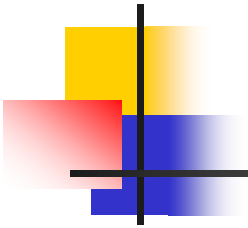
Ограниченные типы

```
//аргумент типа для
// T должен быть Number, или производный
// от Number класс

class Stats<T extends Number> {
  T[] nums;

// массив типа Number или его подкласса
```



Пример использования

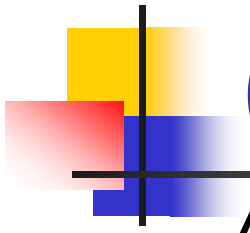
```
Integer inums[] =  {1, 2, 3, 4, 5 };
Double dnums[]  =   {1.1, 2.2, 3.3, 4.4,
  5.5 };

Stats<Integer> intob = new
  Stats<Integer>(inums);
Stats<Double> dob = new
  Stats<Double>(dnums);
```



Пример использования

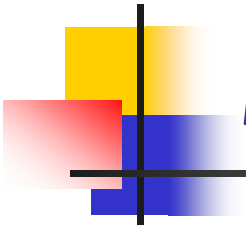
```
double v = intob.average();
    System.out.println("intob average is
  " + v);
double w = dob.average();
    System.out.println("dob average is "
  + w);
```

Метасимвольные аргументы (wildcard argument)

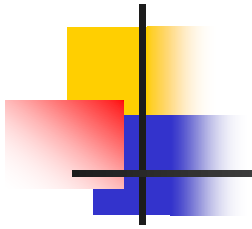


```
// Определяет, равны ли средние
// арифметические.
boolean sameAvg(Stats<?> ob) {
  if ((average()) == ob.average())
    return true;
  return false;
}
//метасимвол «смягчает» контроль,
// позволяя сравнить intob.sameAvg(dob)
```

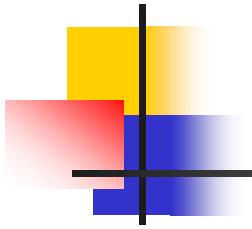
Ограниченные метасимвольные аргументы



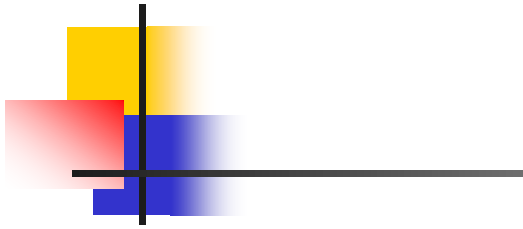
```
class TwoD {
  int x, y;
  TwoD(int a, int b) {
    x = a;
    y = b;
  }
}
```



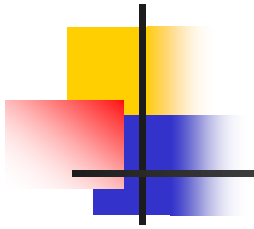
```
class ThreeD extends TwoD {
  int z;
  ThreeD(int a, int b, int c) {
    super(a, b);
    z = c;
  }
}
```



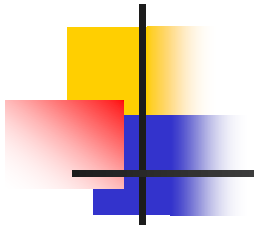
```
class FourD extends ThreeD {
  int t;
  FourD(int a, int b, int c, int d) {
    super(a, b,c);
    t = d;
  }
}
```



```
class Coords<T extends TwoD> {
  T[] coords;

  Coords(T[] o) { coords = o; }
}
```



```
static void showXY(Coords<?> c)    {
System.out.println("X Y Coordinates;");
for(int i=0; i < c.coords.length; i++)
  System.out.println(c.coords[i].x + "    "
     +c.coords[i].y);
   System.out.println() ;
}
```



```
static void showXYZ(Coords<? extends ThreeD> c)
{
  System.out.println("X Y Z Coordinates:");
    for(int i=0; i < c.coords.length; i++)
      System.out.println(c.coords[i].x +  "   "  +
                         c.coords[i].y +  "   "  +
                          c.coords[i].z);
  System.out.println() ;
}
```

Метасимволы супертипов

```
<? super T>
<  >
```