

JavaMUG May 2005 Meeting

Implementing Generics from JDK 5.0

Presented by
Jim Bender





Agenda

Recursion Software, Inc.

- **Generics Overview**
- **Sun's Approach to Implementing Generics**
- **Getting Started with Generics**
- **IDEs**
- **Information Sources**
- **Initial Approach to Parameterizing**
- **Lessons Learned**
- **Constraints with Generics**
- **Autoboxing with Parameterized Number Classes**
- **Benefits from Using Generics**
- **Summary**





Generics Overview

Recursion Software, Inc.

- **Java™ has traditionally allowed heterogeneous collections**
- **Heterogeneous collections complicated the use of objects extracted from collections, as they were (of course) Objects**
- **Homogeneous collections still required casting objects to their expected class**
- **Generics removes the need for casting from homogeneous collections**
- **A generic collection is parameterized like this: `List<T>`, `List<T extends Integer>`, or `List<Integer>`, for example**
- **With generics, we know the class of an extracted object, and no cast is required**

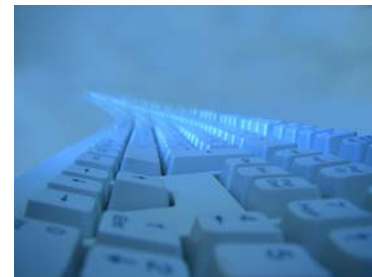




Sun's Approach to Implementing Generics

Recursion Software, Inc.

- **Java™ generic implementation**
 - **Backward compatibility with existing code, including JAR files**
 - ◆ **Generics are only visible during compile time**
 - ◆ **Runtime code has generics removed**
 - ◆ **Generics are implemented by the combination of erasure (of generic type information) and “bridging” method generation**
 - **JDK 5.0 code will cause class version errors when loaded into earlier JDK's.**

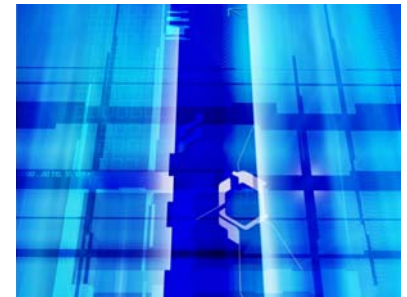




Sun's Approach to Implementing Generics

Recursion Software, Inc.

- Erasure follows this process*:
 - When a class doesn't have generic type parameters, the definition is unaffected by erasure
 - Generic type parameters are removed
 - When a type parameter is erased, the parameter is mapped to a type (possibly Object, where the parameter is "<T>")
 - Casts are generated where needed to allow the code to compile



* William Grosso, "Explorations: Generics, Erasure, and Bridging, Java.net, December 2, 2003.



Sun's Approach to Implementing Generics

Recursion Software, Inc.

■ Runtime consequences:

- All generic type parameters are gone
- The erased code is similar to “raw Java”
- Any unchecked conversions that could not be removed might cause runtime exceptions





Sun's Approach to Implementing Generics

Recursion Software, Inc.

- **Reducing runtime problems:**
 - **What this means: parameterizing completely is important**
 - **Angelika Langer suggests using wildcard question mark if don't know proper generic type parameter to use**
 - **By parameterizing, you allow compiler to assure you have consistent types**
 - **Not absolute assurance, but much better than what we had, where we had to hope that design was consistently putting objects of a certain type into a collection**
 - **Ordinary Java programmers (non-guru) who write normal code, are helped by maximal generic parameterization, as class cast exceptions will be rarer**



Sun's Approach to Implementing Generics

Recursion Software, Inc.

■ Backward compatibility

- Legacy code, either in source form or JAR files can be imported into JDK 5.0
- Code compiled in JDK 5.0 can't be loaded into an older environment such as the JDK 1.4.2





Getting Started with Generics

Recursion Software, Inc.

- **How much penetration has JDK 5.0 made in business and industry?**
 - **How many people are currently using JDK 5.0 in their work?**
 - ◆ **Probably not many**
- **Issues:**
 - **Issues such as which JDK version required by your application server**
 - ◆ **During 2003-2004, slow to move forward to JDK 1.4.2 on server**
 - **Using 1.4.2 in Swing client and 1.3.1 on server**
 - **Only able to move forward to 1.4.2 when corporate mandate made to move to newest app server version**



Getting Started with Generics

Recursion Software, Inc.

- **The natural approach to adding generics:**
 - **“Automatically” adding generic parameters and using them as types in the code**
 - **Issues regarding method visibility on generic types were immediately encountered**
 - **Solution: Have generic parameters extend types so that right methods would be available on generic types**





Getting Started with Generics

Recursion Software, Inc.

■ JGL Toolkit:

- Implements STL-like generic programming with collections, functions, predicates, and algorithms
- Developed starting in 1996 and inspired the eventual Java collections framework
- JGL Toolkit has kept moving forward to keep pace with the latest versions of the JDK





Getting Started with Generics

Recursion Software, Inc.

■ JDK 5.0 and Generics

- **JDK 5.0 is a fairly radical reshaping of Java**
- **Result has disappointed people who wanted what they considered to be “real” generics, similar to Ada generics or C++ templates**
- **Driving force behind what we actually got was the business need to achieve some degree of backward compatibility for legacy code**
- **Unparameterized code can be used in JDK 5.0, although will cause warning messages**
- **Can import older JAR files or source files**
- **Jars built in JDK 5.0 can't be used in older environments because class version issue immediately causes problems**



IDEs

Recursion Software, Inc.

- IDE's, "free IDEs", and their support for JDK 5.0
- Two main players in the "free IDE" market:
 - Eclipse
 - NetBeans from Sun
- Eclipse still dominant player
 - Other IDEs: IntelliJ IDEA IDE, Borland JBuilder





IDEs

Recursion Software, Inc.

IDE	JDK 5.0 Support	Cost	Characteristics
Eclipse	Yes (only in milestone builds (ie. 3.1M4, etc.))	Free Open source	<ul style="list-style-type: none">■ Backed by IBM and heavily promoted■ Core development team previously developed Visual Age Java■ Tightly integrated with ANT■ Well-integrated with CVS
Netbeans	Yes	Free	<ul style="list-style-type: none">■ Backed by Sun and included in JDK 5.0 downloads■ Better support for GUI development than Eclipse■ Integrated with CVS■ Promotes use of layout managers for GUIs
IDEA	Yes	<ul style="list-style-type: none">■ Free to open source projects■ \$499 for commercial development	<ul style="list-style-type: none">■ Promoted by Java gurus outside of Sun■ Has big “buzz” factor
JBuilder	Yes	<ul style="list-style-type: none">■ Foundation: \$0■ Developer: \$500■ Enterprise \$3,500	<ul style="list-style-type: none">■ Promotes style of GUI development that doesn't use layout managers



Information Sources

Recursion Software, Inc.

Sources:

- **Google search – JDK 1.5, JDK 5.0**
- **Sun on-line documentation and blogs**
 - **Gilad Bracha's Generics tutorial from Sun**
 - ◆ <http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf>
 - **J2SE 5.0 in a Nutshell article**
 - ◆ <http://java.sun.com/developer/technicalArticles/releases/j2se15/>
- **Angelika Langer's website - Java Generics FAQs**
 - <http://www.langer.camelot.de/GenericsFAQ/JavaGenericsFAQ.html>
- **David Hall's website - JDK 1.5 and JGA**
 - <http://jga.sourceforge.net/docs/AddingAlgorithms.shtml>
- **Articles and posting on forums**
 - **If written prior to JDK 5.0 release**
 - ◆ **See "Tiger" pre-release of JDK**



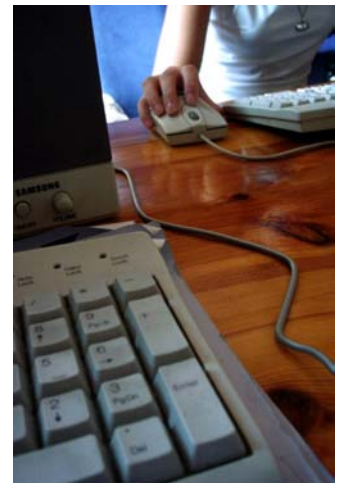


Getting Started

Recursion Software, Inc.

Steps for getting started

- Look for materials that you can easily absorb
- Programmers relate to different writing styles, so useful to do broad search
- Pick articles and tutorials with good “impedance” match
- Dive in and start parameterizing

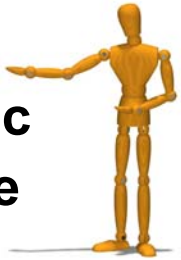




Initial Approach to Parameterizing

Recursion Software, Inc.

- **My approach:**
 - **When parameterizing a class, substitute generic parameter for every type that could be inserted into a collection**
 - **Collection would have that type as parameter**
 - **If compiler complains about result, investigate issues involved and experiment**
 - **If unresolved**
 - ◆ **Consult Angelika Langer's FAQs**
 - ◆ **Google to find specific example to match**
 - **Once collections parameterized, it drives how generic types are applied to method parameters and instance variables**

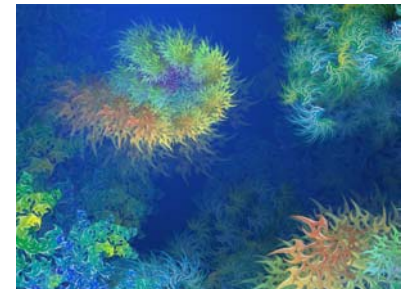




Initial Approach to Parameterizing

Recursion Software, Inc.

- **Java interfaces and generics**
 - **Problematic cases:**
 - ◆ **Classes that implement interfaces**
 - **True where interface has Object arguments**
 - ◆ **Interfaces that extend another interface**
 - ◆ **Clone methods**
 - **Cause of unchecked conversion warnings**
 - **Return Object from clone method which had to be cast to desired type**





Initial Approach to Parameterizing

Recursion Software, Inc.

- **Marginal cases:**
 - **Code appears to demand parameterization, even though class hard-wired for parameter type**
 - ◆ **Case with adapters which adapt arrays of primitive types**
 - ◆ **Adapters wrap primitive arrays so they behave like a collection**
 - **Eliminate unchecked conversion warnings by parameterizing**





Initial Approach to Parameterizing

Recursion Software, Inc.

- **Types in generic programming**
 - **To do parameterization correctly for any STL-like generic programming system, input parameters and returned values for functions could all be different types, especially in interfaces for functions and predicates**
 - **Some function classes that implement an interface might have one or more identical parameter types and returned value type**
 - **Operations on parameterized variables often drive type decisions**
 - ◆ **E.g. Strings and numbers (the boxed classes)**



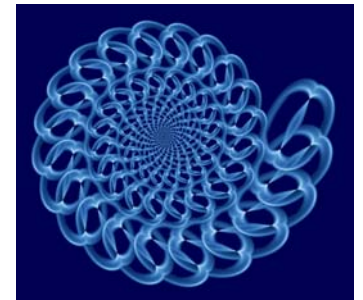


Lessons Learned

Recursion Software, Inc.

■ Static methods

- **Syntax for parameterizing static methods seems peculiar with leading parameters prior to return type, but necessary as declarations prior to use in method signature and body**
- **Classes with only static methods don't have class level generic parameters**
- **An advantage of parameterized static methods is that nothing special is needed when invoked**
- **Compiler infers correct generic types**





Lessons Learned

Recursion Software, Inc.

■ Simple illustration shows method signature for static method:

```
Public static <T extends Number> T mean(Collections<T> c)
```

- One revelation is the case where there are conflicts due to ambiguities caused by the inability of the compiler to choose the correct erasure
- Another issue is some warnings about unchecked conversion cannot be removed





Constraints with Generics

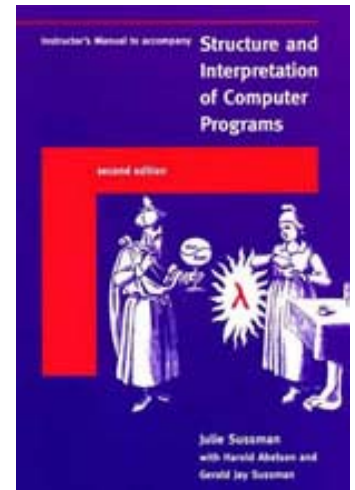
Recursion Software, Inc.

■ Generics

- Easily do generic arithmetic on boxed numbers
- One toy example used constraint network and generic arithmetic in a Junit test case

■ Research on constraints

- MIT AI Lab dissertation circa 1980
- **Book: Structure and Interpretation of Computer Programs** by Harold Abelson, Julie Sussman, Gerald Jay Sussman
 - ◆ Has simplified framework



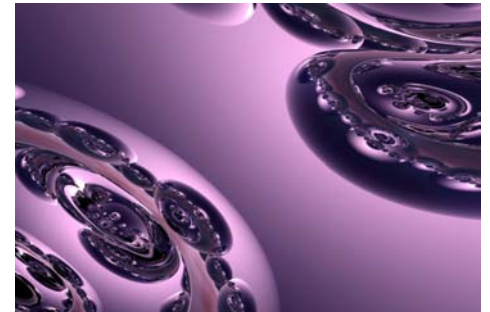


Constraints with Generics

Recursion Software, Inc.

- **Illustration tool: toy constraint network for converting between Fahrenheit and Celsius temperatures**

- **Framework uses following classes:**
 - **Constraint**
 - ◆ **Adder**
 - ◆ **Multiplier**
 - ◆ **Constant**
 - **Connector**

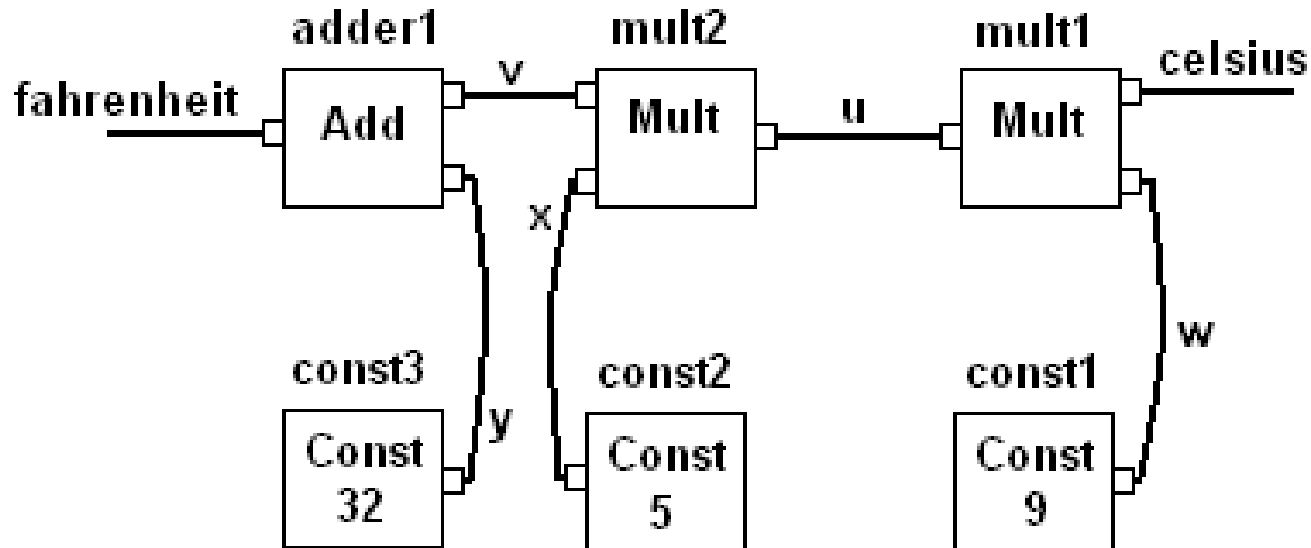




Constraints with Generics

Recursion Software, Inc.

- First need Celsius to Fahrenheit converter class:





Constraints with Generics

Recursion Software, Inc.

```
public class CelsiusFahrenheitConverter<T extends Number> {  
  
    Connector<T,T> c = null;  
    Connector<T,T> f = null;  
    Connector<T,T> u = null;  
    Connector<T,T> v = null;  
    Connector<T,T> w = null;  
    Connector<T,T> x = null;  
    Connector<T,T> y = null;  
    Constraint<T,T> mult1 = null;  
    Constraint<T,T> mult2 = null;  
    Constraint<T,T> adder1 = null;  
    Constraint<T,T> const1 = null;  
    Constraint<T,T> const2 = null;  
    Constraint<T,T> const3 = null;  
}
```





Constraints with Generics

Recursion Software, Inc.

```
/**
 * @throws ConstraintException
 * @throws ClassNotFoundException
 * @throws IllegalAccessException
 * @throws InstantiationException
 *
 */
public CelsiusFahrenheitConverter(
    Class<T> number,
    Connector<T,T> celsius,
    Connector<T,T> fahrenheit
) throws ConstraintException, InstantiationException,
    IllegalAccessException, ClassNotFoundException {
```





Constraints with Generics

Recursion Software, Inc.

```
super();
c = celsius;
f = fahrenheit;
u = new Connector<T,T>("u");
v = new Connector<T,T>("v");
w = new Connector<T,T>("w");
x = new Connector<T,T>("x");
y = new Connector<T,T>("y");
mult1 = new Multiplier<T>("mult1",number,c,w,u);
mult2 = new Multiplier<T>("mult2",number,v,x,u);
add1 = new Adder<T>("add1",number,v,y,f);
GenArithI<T> arithBox = ArithmeticUtil.arithType(number);
const1 = new ConstConstraint<T>(number,(T)arithBox.toValue(9.0),w);
const2 = new ConstConstraint<T>(number,(T)arithBox.toValue(5.0),x);
const3 = new ConstConstraint<T>(number,(T)arithBox.toValue(32.0),y);
}
}
```



Constraints with Generics

Recursion Software, Inc.

- **Junit test method - performs the generic instantiation and drives the constraint network**

```
public void testConstraintsFahrenheit212BigDecimal() throws Exception {  
    Constraint<BigDecimal,BigDecimal> user = new User<BigDecimal>();  
    Connector<BigDecimal,BigDecimal> c =  
        new Connector<BigDecimal,BigDecimal>("c");  
    Connector<BigDecimal,BigDecimal> f = new  
        Connector<BigDecimal,BigDecimal>("f");  
    CelsiusFahrenheitConverter<BigDecimal> converter =  
        new CelsiusFahrenheitConverter<BigDecimal>(BigDecimal.class,c,f);  
    GenArithI<BigDecimal> arithBox = ArithmeticUtil.arithType(BigDecimal.class);  
    c.setValue(arithBox.toValue(100),user);  
    assertEquals("Correct value calculated for a Celsius of 100",  
        arithBox.toValue(212),f.getValue());  
}
```





Autoboxing with Parameterized Numbers

Recursion Software, Inc.

Autoboxing example with Double:

```
public void testVarianceDouble(){
    Vector<Double> data = new Vector<Double>();
    data.add(1.0);
    data.add(3.0);
    data.add(3.0);
    data.add(5.0);
    data.add(5.0);
    data.add(5.0);
    data.add(5.0);
    data.add(7.0);
    data.add(7.0);
    data.add(7.0);
    data.add(9.0);
    data.add(9.0);
    data.add(9.0);
    data.add(9.0);
    data.add(1.0);
    data.add(1.0);
    Double variance = Statistics.variance(data);
    assertEquals(variance.doubleValue(), 8.65, 0.0001);
}
```





Autoboxing with Parameterized Numbers

Recursion Software, Inc.

Autoboxing example with Float (what you would expect):

```
public void testVarianceFloat() {  
    Vector<Float> data = new Vector<Float>();  
    data.add(1.0);  
    data.add(3.0);  
    ...  
    Double variance = Statistics.variance(data);  
    assertEquals(variance.doubleValue(), 8.65, 0.0001);  
}
```



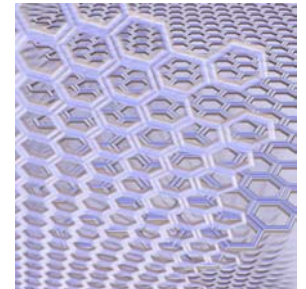


Autoboxing with Parameterized Numbers

Recursion Software, Inc.

**Autoboxing example with Float:
(what is actually required: it won't autobox)**

```
public void testVarianceFloat(){  
    Vector<Float> data = new Vector<Float>();  
    data.add(new Float(1.0));  
    data.add(new Float(3.0));  
    ...  
    Double variance = Statistics.variance(data);  
    assertEquals(variance.doubleValue(), 8.65, 0.0001);  
}
```





Benefits from Using Generics

Recursion Software, Inc.

■ Parameterization:

- Offers degree of protection not found with raw types
- If unsure of proper generic type to use, use wildcard
- Otherwise use bounded generic parameter types
 - ◆ If accessing methods, unbounded wildcard works as compiler enforces restrictions to prevent type problems
 - ◆ If need access methods, use either bounded type or wildcard
 - ◆ Bound allows compiler to determine method visibility

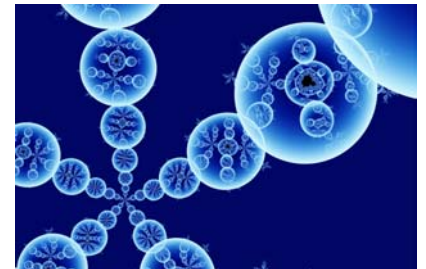




Benefits from Using Generics

Recursion Software, Inc.

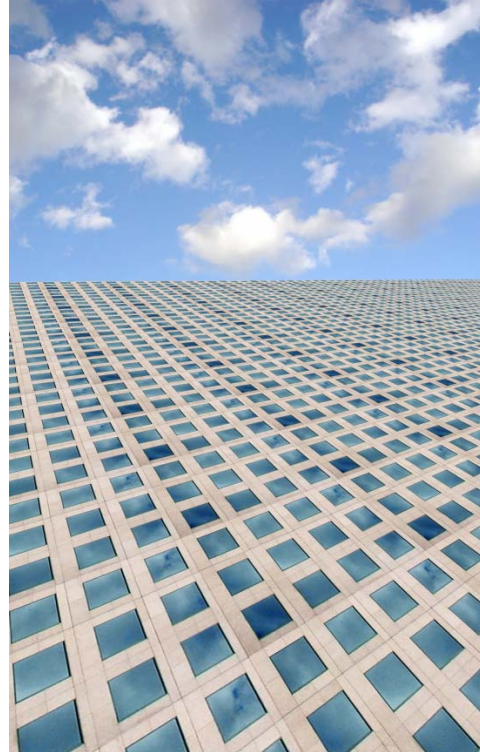
- **Generic parameterization allows much tighter and more readable code**
- **Adds new layer of protection from runtime errors that are prevented at compile time**
- **Generic type parameters also provide information previously not present**
- **Can immediately see type of objects inserted into collections**





Implementing Java Generics from JDK 5.0

Recursion Software, Inc.



Jim Bender

jbender@recursionsw.com

972.731.8800 x108



Contact Information

Software
for Software Developers™

Recursion Software, Inc.

Headquarters: Recursion Software, Inc.
2591 North Dallas Parkway, Suite 200
Frisco, Texas 75034
Tel: 972.731.8800
800.727.8674
Fax: 972.731.8881

Email: info@recursionsw.com

Sales: sales@recursionsw.com

Training: training@recursionsw.com

Support: support@recursionsw.com

Technical: engineer@recursionsw.com

Website: www.recursionsw.com

