Java SE Generics Introduction

abelski

Introduction

What Are Generics?

- A generic class is a class that each time we instantiate it we specify a specific type (or types) that will be used by the new object.

Using JDK 1.4 (or Lower)

- Given the following code:

```
import java.util.*;

public class GenericsSample1
{
        public static void main(String[] args)
        {
                ArrayList array = new ArrayList();
                array.add(new Date());
                System.out.println( ((Date) array.get(0)).getTime());
        }
}
```

- Try to compile & run first with a JDK version lower than 1.5 and then with a JDK version 1.5 (or higher).

Using JDK 1.4 (or Lower)

- Trying to compile the given code using JDK version lower than 1.5 won't generate any compile error/warning.
- Trying to run the following code using JDK version lower than 1.5 won't generate any exception. Yet, if we do a mistake and instead of casting to Date we cast to a wrong type then during run time the code will throw an exception and when compiling it we won't get any compile error/warning.

Using JDK 1.5 (or Higher)

- Trying to compile the given code using JDK version 1.5 (or higher) will generate a warning message.
- Trying to run the given code using JDK version 1.5 (or higher) won't generate any exception. If we introduce a casting mistake in our code then during run time the code will throw an exception.
- Yet, if we do pay attention to compile warnings and rewrite the code according to generics' rules then a successful compilation guarantees that executing the code won't throw any type cast exception.

Using JDK 1.5 (or Higher)

- Thanks to generics we can be on the safe side ensuring the compilation succeeds without any warning and based on that be 100% confident that during runtime we won't get any runtime exception.
- Thanks to generics we can avoid the need to cast type during runtime.

Basic Sample

- Let's look into the API documentation of the class ArrayList at http://java.sun.com/j2se/1.5.0/docs/api/java/util/ArrayList.html.
- From browsing ArrayList documentation we can see that the ArrayList was declared with generics support. For that reason, using this class with JDK 1.5 should be different comparing with using it with JDK 1.4 (or lower).

Basic Sample

- When using JDK 1.4 (or lower):

```
ArrayList array = new ArrayList();
array.add(new Date());
Date date = (Date)array.iterator().next();
```

- When using JDK 1.5 (or higher):

```
ArrayList<Date> array = new ArrayList<Date>();
array.add(new Date());
Date date = array.iterator().next();
```

- Now, when using JDK 1.5 (or higher) the compiler can check the type correctness of the program at compile-time.

Generics Definition Overview

- A generic class is a class that each time we instantiate it we specify a specific type (or types) that will be used by the new object.
- “Generic” [1, adjective]

  “1 a: relating to or characteristic of a whole group or class"

  Merrian-Webster Online Dictionary

  http://www.m-w.com/dictionary/generic
- The collection classes are a good sample for generic classes:

  ```
  LinkedList<Date> list = new LinkedList<Date>();
  ```

  This new LinkedList object holds elements, that each one of thes is of type Date.

Generics Definition Overview

- "This long-awaited enhancement to the type system allows a type or method to operate on objects of various types while providing compile time type safety. It adds compile-time type safety to the Collections Framework and eliminates the drudgery of casting." (Sum Microsystems)
- Using generics we can validate the types when compiling the code. Using generics we can avoid the need of doing type casting during runtime.

Generics Basic Use

- When using a class that uses generics there is a need to mention the type we choose to replace the generic type in the following two cases:

  When referring to a type of a class that uses generics...

  Examples: declaring a variable of this type, declaring a method that one of its parameters is of this type...

  When instantiating a class that uses generic...

  Examples: instantiating a class calling its constructor we will do it in accordance with the generics format guidelines...

Declaring Class

- Declaring a new class that uses generics can be easily done. The syntax would be the same as the one used for declaring Java core classes that use generics (Collection API classes).

Declaring Class

```
import java.util.*;

public class GenericsDeclare1
{
	public static void main(String[] args)
	{
		MyStack<Integer> stack = new MyStack<Integer>();
		stack.push(new Integer(2));
		stack.push(new Integer(4));
		stack.push(new Integer(24));
		stack.push(new Integer(14));
		Integer temp = stack.pop();
		System.out.println("temp="+temp);
	}
}
```

Declaring Class

```
class MyStack<E>
{
        ArrayList<E> array = new ArrayList<E>();

        void push(E element)
        {
              array.add(element);
        }

        E pop()
        {
              E reply = array.get(array.size()-1);
              array.remove(array.size()-1);
              return reply;
        }
}
```

Type Inference

- As of Java 7 we can take out the type arguments required to instantiate a generic class as long as the compiler can infer the type arguments from the context.
- Many developers name the remaining pair of angle brackets as 'diamond'.

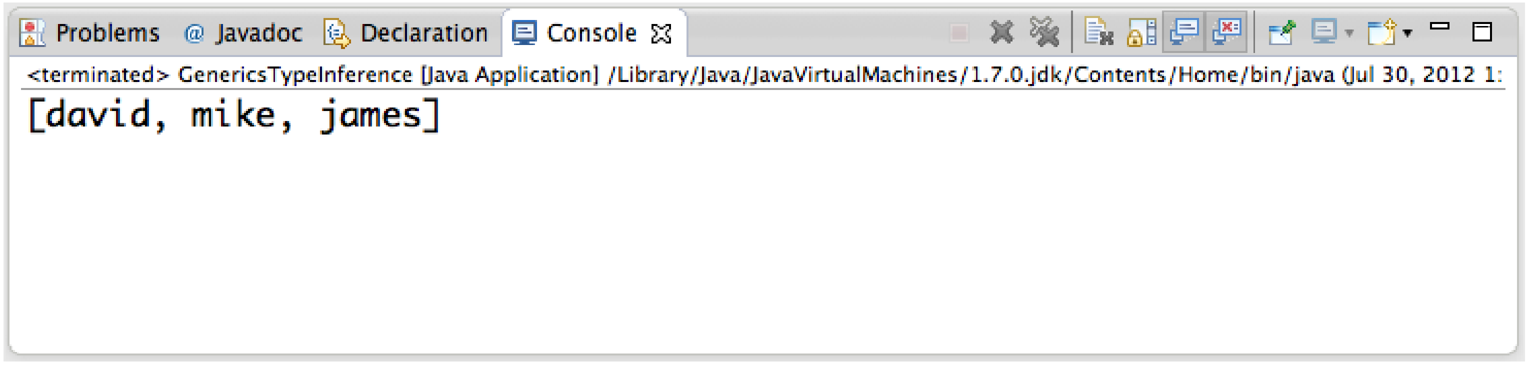
Type Inference

```java
package com.abelski.samples;

import java.util.LinkedList;
import java.util.List;

public class GenericsTypeInference
{
    public static void main(String[] args)
    {
        List<String> list = new LinkedList<>();
        list.add("david");
        list.add("mike");
        list.add("james");
        System.out.println(list);
    }
}
```

Type Inference



```
<terminated> GenericsTypeInference [Java Application] /Library/Java/JavaVirtualMachines/1.7.0.jdk/Contents/Home/bin/java (Jul 30, 2012 1:
[david, mike, james]
```