Java Generics

Parametric Polymorphism

WHAT IS GENERICS?

DR. ERIC CHOU

IEEE SENIOR MEMBER

Mini-Course Objectives

- To know the benefits of generics.
- To use generic classes and interfaces.
- To define generic classes and interfaces.
- To understand why generic types can improve reliability and readability.
- To define and use generic methods and bounded generic types.
- To develop a generic sort method to sort an array of **Comparable** objects.
- To use raw types for backward compatibility.
- To explain why generics are necessary.
- To describe generic type erasure and list certain restriction and limitations on generic types caused by type erasure.
- To design and implement generic matrix classes.

Polymorphism in Java

- Sub-Type Polymorphism
  - Overloading (Sharing an operator for different data types)
  - Overriding (Polymorphic Method)
  - Dynamic Binding
- **Parametric Polymorphism (Generics)**
  - Generic Containers (Generic Data Structure)
  - Generic Methods
  - Generic Parameters

Parametric Generics

- Generic Parameters on the parameter list of a method
- Generic return value for the output of a method
- Generic container such as **ArrayList**

We may define a generic sort method with input generic array and returned sorted generic array.

Java define a generic **ArrayList** class for storing the elements of a generic type.

Generic in Java

- Use a type variable declaration to notify compiler that the data type can be substituted in the implementation of method, data structure so that the method can take different parameter types and the data structure can contain element of different data types.

```
<T> T method(T arg);
};
```

```
Bar b = method(aBar); //OK
b = method(not_a_Bar);//Compile error
```

- The actual type parameter is not passed
  - The compiler infers the type argument
- Compiler ensures arg and return are same type
  - When the method is compiled
  - On method calls

```
class Foo <T> {
  void method(T arg);
};
```

```
Foo<Bar> fb = new Foo<bar>;
fb.method(aBar); //OK
fb.method(not_a_Bar); // compile error
```

- **Foo is generic:** Same behavior for any possible T (the type parameter)
- **Single Source:** only one foo.java
  - Low maintence
- **Single binary:** only one foo.class
  - No binary explosion (but imposes restrictions)

Advantage of Using Generic

To detect errors at compile time rather than run-time.

A generic class or method permits you to specify allowable types of objects that the class or method can work with. If you attempt to use an incompatible object, the compiler will detect that error.

To improve the readability and reliability of software.