Java Generics

Parametric Polymorphism

RAW TYPE

DR. ERIC CHOU

IEEE SENIOR MEMBER

Raw Type and Backward Compatibility

- The **ArrayList<E>** Class is a generic class with type variable (type parameter) E.
- We may replace **E** with any reference data type (object).
- If the type variable **E** is not specified, we call it **raw type**.

Raw Type:

```
ArrayList list = new ArrayList();
```

This is roughly equivalent to

```
ArrayList<Object> list = new ArrayList<Object>();
```

- Even though it is raw type, it can take element of any type. However, if programmers want to use it for features other than supported by Object class, then it will create compilation error. It will be necessary to cast the Object type back to the original data type to work normally.

Raw Type and Backward Compatibility

A generic class or interface used without specifying a concrete type, called a raw type, enables backward compatibility with earlier versions of Java.

- You can use a generic class without specifying a concrete type like this:

  ```
  GenericStack stack = new GenericStack();
  ```

- This is roughly equivalent to

  ```
  GenericStack<Object> stack = new GenericStack<Object>();
  ```

- A generic class such as **GenericStack** and **ArrayList** used without a type parameter is called a raw type. Using raw types allows for backward compatibility with earlier versions of Java.

- For example, a generic type has been used in **java.lang.Comparable** since JDK 1.5, but a lot of code still uses the raw type Comparable, as shown in the program we will demo.

Generic Maximum Finding

Demo Program:Max.java+MaxUsingGenericType.java+MaxDemo.java:

Go BlueJ!