Java Generics

Parametric Polymorphism

UNIVERSAL REFERENCE TYPE - OBJECT

DR. ERIC CHOU

IEEE SENIOR MEMBER

Generic Programming Using Universal Reference Type

Object

- Java's Object class type provides a possibility for writing one code and using it for all types.
- Java's Object class is on top of the class hierarchy so all classes are its sub-classes.
- Java's primitive data types can all be converted to its related wrapper class for object-oriented processing.
- Object Reference type can be used to refer to any object types. To convert back, just need to (cast) back to the original class.

```
String s = new String("Java Programming");
Object obj = s;
```

To convert back

```
S = (String) obj;
```

Issues:

1. Depends on run-time dynamic binding
2. Not all types are compatible with the operator (function)
3. Hard to identify the reference type if program is huge.

Generic Object Container using Object Reference Type

- Really collection of references to objects.
- Class **ArrayBag**
  - Object is a bag of objects of any type
  - Implemented on an array of Object type
  - Object[] data;
- A collection of objects is actually a collection of references to objects
- Each element of the data array is a reference to an object
- Some of them may refer to the same object.

Object Operators

Equality check: == and != tests for objects

- Only test if two references refer to the same object
- Not if they have the same content
- Two distinct objects can have the same content.
- Provide test of equality of contents, write an equals method in which object contents are compared.
- Hard to implement interfaces such as Comparable, Iterable because Object is the top level class. Need to convert back to some concrete class type or interface type reference to perform sorting.

Object Method Versus Generic Method

Object Method:

```
static Object middle(Object[] data){
   if (data.length== 0){
      return null;
   }
   else{
      return data[data.length/2];
   }
}
```

Generic Method:

```
static <T> T middle(T[] data){
   if (data.length== 0){
      return null;
   }
   else{
      return data[data.length/2];
   }
} // This is in Generic Method Format
```

Object Container Versus Generic Container


```
package java.lang;

public interface Comparable {
  public int compareTo(Object o)
}
```

(a) Prior to JDK 1.5

```
package java.lang;

public interface Comparable<T> {
  public int compareTo(T o)
}
```

(b) JDK 1.5

Figure A.

```
Comparable c = new Date();
System.out.println(c.compareTo("red"));
```

(a) Prior to JDK 1.5

```
Comparable<Date> c = new Date();
System.out.println(c.compareTo("red"));
```

(b) JDK 1.5

Figure B.