

CSC 243 - Java Programming

Java Generics

Java Generics

- Generics let you parameterize types
- This is called parametric polymorphism
- The benefit of generics is to detect errors at compile time
- Generics have a *formal generic type* that can be replaced later with an *actual concrete type*

The Pre-Generic ArrayList

- The add method signature

```
add(Object o);
```

- Usage Example

```
ArrayList list = new ArrayList();  
list.add("one");  
list.add(1);  
Iterator itr = list.iterator();  
while (itr.hasNext()) {  
    // run time error on second iteration  
    System.out.println((String)(itr.next()));  
}
```

The Generic ArrayList<E>

- The add method signature

```
add(E o); // E is a formal generic type
```

- Usage Example

```
ArrayList<String> list = new ArrayList<>();  
list.add("one");  
list.add(1); // compile time error  
for (String s: list) {  
    System.out.println(s);  
}
```

Defining Generic Types

- A generic type can be defined for a class or interface

```
public class Stack<E>
```

- A concrete type must be specified when creating an object or declaring a variable of the type

```
Stack<String> stack = new Stack<>();
```

Generic Methods

- A generic type can be defined for a static method

```
public class C {  
    // The generic type is placed immediately  
    // after the static keyword  
    public static <E> void printArray(E[] a) {  
        for (int i = 0; i < list.length; i++) {  
            System.out.println(list[i]);  
        }  
    }  
}
```

- Calling a generic method

```
Integer[] integers = {1, 2, 3, 4}  
// prefix the method call with the  
// generic type  
C.<Integer>printArray(integers);
```

Bounded Types

- A generic type can be specified as a subtype of another type

```
public static <E extends Number> E max(  
    E o1, E o2)  
{  
    return o1 > o2 ? o1 : o2;  
}
```

Wildcard Generic Types

- Wildcards specify a range for a generic type
 - unbounded wildcard `<?>`
 - bounded wildcard `<? extends T>`
 - lower bounded wildcard `<? super T>`

Type Erasure and Restrictions on Generics

- Type erasure: information on generics is used by the compiler but not available at runtime
- Restrictions due to type erasure
 - 1 Cannot use `new E()`
 - 2 Cannot use `new E[]`
 - 3 A generic type parameter of a class is not allowed in a static context
 - 4 Exceptions cannot be generic