Generic Types in Collections

Mason Vail
Boise State University Computer Science

What’s With the Angle Brackets?

```
ArrayList<Integer> intList =
  new ArrayList<Integer>();
```

Type Safety

```
ArrayList<Integer> intList =
   new ArrayList<Integer>();

//valid
intList.add ( new Integer ( 5 ) );

//invalid, won’t compile
intList.add ( “Hello” );
```

Why Not Custom Collections for Every Type?

```
StringsArrayList
IntegersArrayList
BooksArrayList
WidgetsArrayList
RecordsArrayList
```

…

Enormous code duplication where the only difference between classes is the stored type.

Does it Matter What Type is Stored?

What does a Collection do with the items it stores?

- Holds them
- Returns them

Does it actually need to know anything about them?

- Not really

Does it call any of their methods?

- Maybe toString()

Generics Let Us Decide a Type On the Fly

Generics, or Generic Types, are placeholders in a class or interface, for an unknown data type.

We specify the actual type when we actually create a reference or call a constructor.

Generics in a Class

```
public class TinyBag<E> {
 private E item;

 public boolean add(E element) {
  if (item != null) return false;
  item = element;
  return true;
 }
}
```

Generics in a Class

```
TinyBag<String> bag = new TinyBag<String>();
```

```
public class TinyBag<E> {
 private E item;

 public boolean add(E element)
 {
  if (item != null) return false;
  item = element;
  return true;
 }
}
```

```
public class TinyBag {
 private String item;

 public boolean add(String element)
 {
  if (item != null) return false;
  item = element;
  return true;
 }
}
```

Specified Type Could be Different for Every Object

```
TinyBag<String> favoriteWord =
   new TinyBag<String>();
favoriteWord.add( “Tergiversation” );

TinyBag<Integer> favoriteNumber =
   new TinyBag<Integer>();
favoriteNumber.add( new Integer ( 3 ) );
```

Generics in Interfaces

```java
/** Simple container */
public interface Bag<E> {
  /** Add element to the Bag.
   * @return true if element is
   *   added, else false */
  public boolean add ( E element );

  /** Remove and return one element
   * @return element or null if no
   *   element */
  public E remove (  );
}
```

```java
/** Contains only one item */
public class SmallBag<E>
             implements Bag<E> {
  private E item;

  public boolean add(E element) {
    //code not shown
  }

  public E remove() {
   //code not shown
  }
}
```

Generic Types in Collections

Mason Vail
Boise State University Computer Science