
WEEK SIX

Acknowledgements: Slides created based off material provided by Dr. Michael Raymer and Dr. Travis Doom

ARRAYLIST EXAMPLE

- ArrayLists do not *require* you to specify a data type when you define them
- So why do we?
- Let's take a look

HOW DOES ARRAYLIST WORK WITH DIFFERENT DATA TYPES?

Because of generics!

GENERICCS

- Allow classes to work with various data types
- Signified by a single capital letter (usually E for Element)
 - Others include N for Number, T for Type, V for Value
- Often accompanied by <>
- Help catch typing errors early
- Control the type a container will hold
- Avoid extra typecasting

GENERIC CLASSES

- If we want our class to work with any data type, we add the *type parameter* to the class header inside <>
 - `public class BingoMachine<E>`
- Represents a class with a collection of elements that are stored and procured randomly
- Can store different kinds of objects because of the generic type E

BINGOMACHINE DESIGN

BingoMachine

-contents : ArrayList <E>
-rng : Random

+BingoMachine()
+add(item : E)
+pickItem() : E
+isEmpty() : boolean
+clear()

- ASIDE: Why do we contain an ArrayList instead of extending it?
 - Extending means we allow access to *all* of ArrayLists' methods
 - Do we want the user to be able to get any item on their own?
 - By containing ArrayList, we force the user to only be able to access an item via the pickItem() method (a random pick)
- Let's implement it!!!!!!!!!!!!!! :D :D :D

WHAT IF I WANT TO RESTRICT WHAT CAN BE PUT IN MY CLASS?

Bounded type parameters!

BOUNDED TYPE PARAMETERS

- We use the `extends` keyword to denote the class that restricts our generic type
 - `public class BingoMachine <E extends Number>`
- Now we can only store subclasses of `Number` in our `BingoMachine`
- The `extends` keyword is not only used for regular classes but abstract classes and interfaces too
 - `public class BingoMachine <E extends Comparable<E>>`
- Says that the type `E` must be comparable to itself
- In other words, forces the class `E` to implement the `compareTo` method
 - `public boolean compareTo(E otherItem)`

PUTTING EVERYTHING TOGETHER

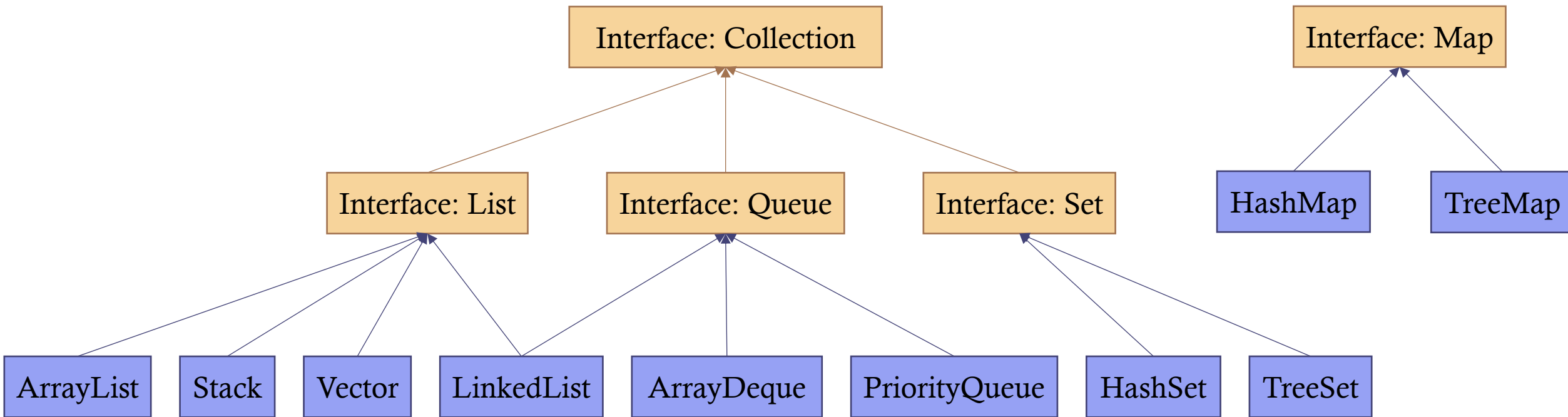
- What does this mean?
- `public class SortedList<E extends Comparable<E>> extends ArrayList<E>`
- SortedList IS-A ArrayList of generic types where the types are comparable to themselves

JAVA COLLECTION INTERFACE

- Stores a collection of elements
- No get() because not all collections have an order

Modifier and Type	Method and Description
boolean	<u>add(E e)</u> Appends the specified element to the end of this list.
boolean	<u>addAll(Collection<? extends E> c)</u> Adds all of the elements in the specified collection to this collection (optional operation).
void	<u>clear()</u> Removes all of the elements from this list.
boolean	<u>contains(Object o)</u> Returns true if this list contains the specified element.
boolean	<u>isEmpty()</u> Returns true if this list contains no elements.
<u>E</u>	<u>remove(Object o)</u> Removes the element at the specified position in this list.
int	<u>size()</u> Returns the number of elements in this list.
Object[]	<u>toArray()</u> Returns an array containing all of the elements in this collection.

JAVA COLLECTION INTERFACE



JAVA COLLECTIONS

- Java class with a *very* similar name to the interface
- Contains a collection of useful static methods for collections
 - `Collections.sort(List<T> list)`
 - `Collections.max(Collection<? extends T> coll)`
 - `Collections.min(Collection<? extends T> coll)`
 - `Collections.reverse(List<?> list)`
 - `Collections.shuffle(List<?> list)`
 - `Collections.swap(List<?> list, int i, int j)`