Objectives

- In this session, you will learn to:
  - Explore generics
  - Create a custom generic class
  - Use the type inference diamond to create an object
  - Create a collection without using generics
  - Use collections and generics
  - Implement an ArrayList
  - Use autoboxing and unboxing
  - Implement a Set
  - Implement a HashMap

Generics

- Generics:
  - Provides type safety to code
  - Moves many errors from runtime to compile time
  - Provides cleaner and easier-to-write code
  - Reduces the need for casting with collections
  - Used extensively in the Java Collections API

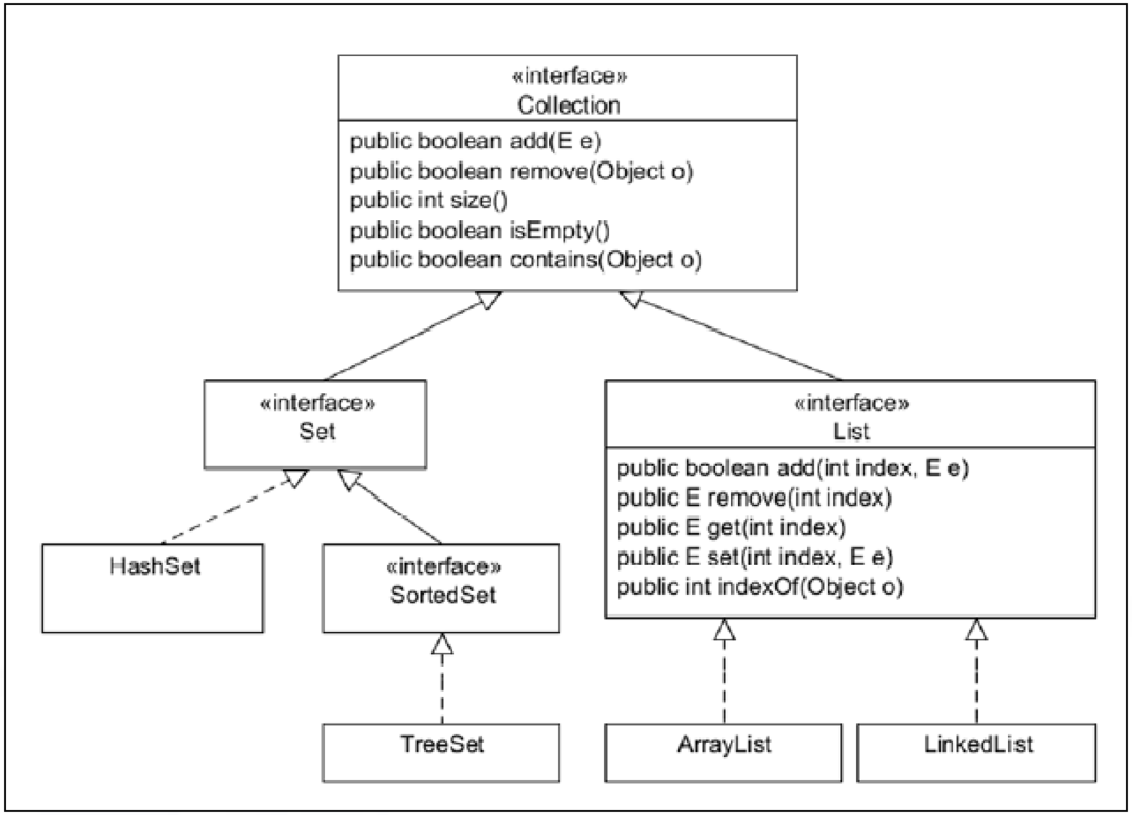
Collections

- Collection:
  - Single object designed to store a group of objects
  - Does not hold primitive types
  - Implements many data structures including `stack`, `queue`, `dynamic array`, and `hash`
- Collections API:
  - Relies on generics for implementation
  - Classes are stored in the `java.util` package

Programming in Java

Collection Types

- The following figure shows all the collection types that inherit from the `Collection` class.



Collection Types (Contd.)

- The following collection types are inherited from the `Collection` class:
  - `HashSet`
  - `TreeSet`
  - `ArrayList`
  - `Deque`

List Interface

- `List` interface:
  - Defines the generic list behavior
  - Helps to store ordered collection of elements
  - Defines the behavior of all `Collections` classes that exhibit list behavior
  - Reference type is used to hide the implementation details
- `List` behaviors:
  - Adding elements at a specific index
  - Adding elements to the end of the list
  - Getting an element based on an index
  - Removing an element based on an index
  - Overwriting an element based on an index
  - Getting the size of the list

ArrayList Implementation Class

- `ArrayList`:
  - Implements a `List` collection
  - Dynamically growable array
  - Has a numeric index
  - Allows duplicate items
- The following embedded Word document shows how to use `ArrayList`.

ArrayList

Set Interface

- Set:
  - List that contains only unique elements
  - Has no index
  - Does not allow duplicate elements
  - Allows iteration through its elements to access them
  - Provides `TreeSet` for sorted implementation

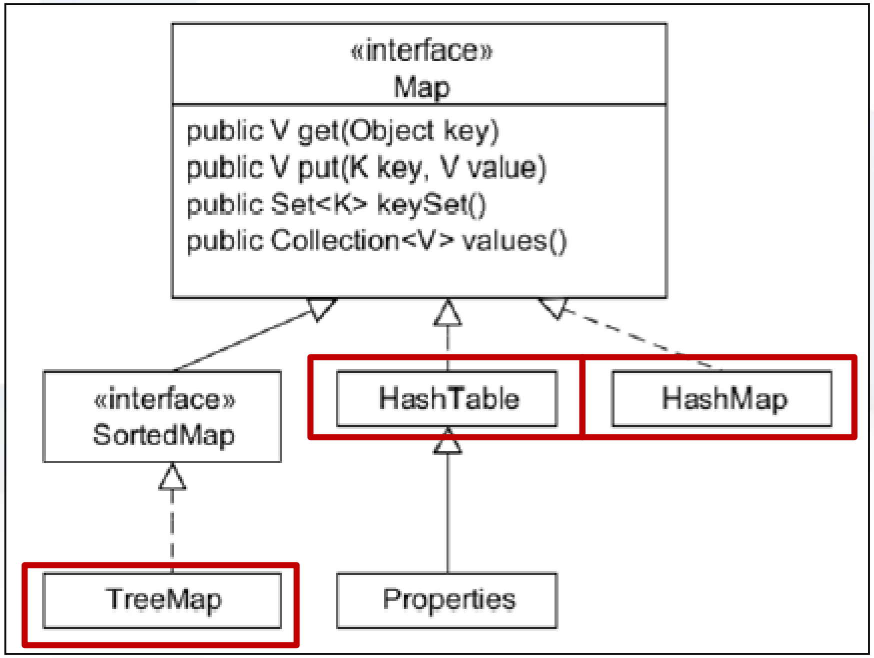
Map Interface

- Map:
  - Collection that stores multiple key-value pairs
  - Called associative arrays in other languages
- The following table is an example of data stored in key-value pairs.

Key	Value
101	Blue Shirt
102	Black Shirt
103	Gray Shirt

Map Types

- `Map` interface:
  - Does not extend the `Collection` interface
  - Represents mappings and not a collection of objects
- The following figure shows the key implementation classes of the `Map` interface.



The `HashMap` class is just like `HashTable`, except that it accepts null keys and values and it is not synchronized.

The `TreeMap` class is a map where the keys are automatically sorted.

The `HashTable` class is a classic associative array implementation with keys and values. The `HashTable` class is synchronized.

Programming in Java

Quiz

Get Ready for the Challenge

Quiz (Contd.)

- Which of the following statements are correct regarding `Collection` classes?
  - `ArrayList` implements a `Set` collection.
  - `List` is a collection that can be used to implement a stack or a queue.
  - The Collections classes are all stored in the `java.lang` package.
  - `ArrayList` is a dynamically growable array.

- Solution:
  - `ArrayList` is a dynamically growable array.

Programming in Java

Quiz (Contd.)

- Fill in the blank:
  - A ____ interface is a collection that stores multiple key-value pairs.

- Solution:
  - `Map`

Summary

- In this session, you learned that:
  - Generics provides type safety to code.
  - The type inference diamond indicates that the right type definition is equivalent to the left.
  - A collection is a single object designed to store a group of objects.
  - The `List` interface helps to store ordered collection of elements.
  - An `ArrayList` is a dynamically growable array.
  - A wrapper class wraps a primitive type into an object of the class.
  - A `Set` is a list that contains only unique elements.
  - A `Map` is a collection that stores multiple key-value pairs.