Learning Consolidation

Store and Manipulate Objects Using Ordered Collections

Learning Objectives

- Understand Collection Framework
- Introduction to Generics
- Understand the difference between ArrayList and LinkedList

Slide Note

Collection Framework

- The collection framework provides built-in interfaces and classes to store and manipulate a group of objects.
- A collection in Java:
  - Consists of a group of objects that are built-in or user-defined.
  - Only holds objects and not primitive datatypes.
- Java Collection Framework helps us to do searching, sorting, insertion, manipulation, and deletion on different types of data.
- Java collections can grow dynamically.
- Java collections are in the package `java.util`.

Menu

Slide Note

Here, the collection is a Super interface that extends the Iterable interface.

List, set, dequeue, queue, etc., are the interfaces that extend the Collection interface.

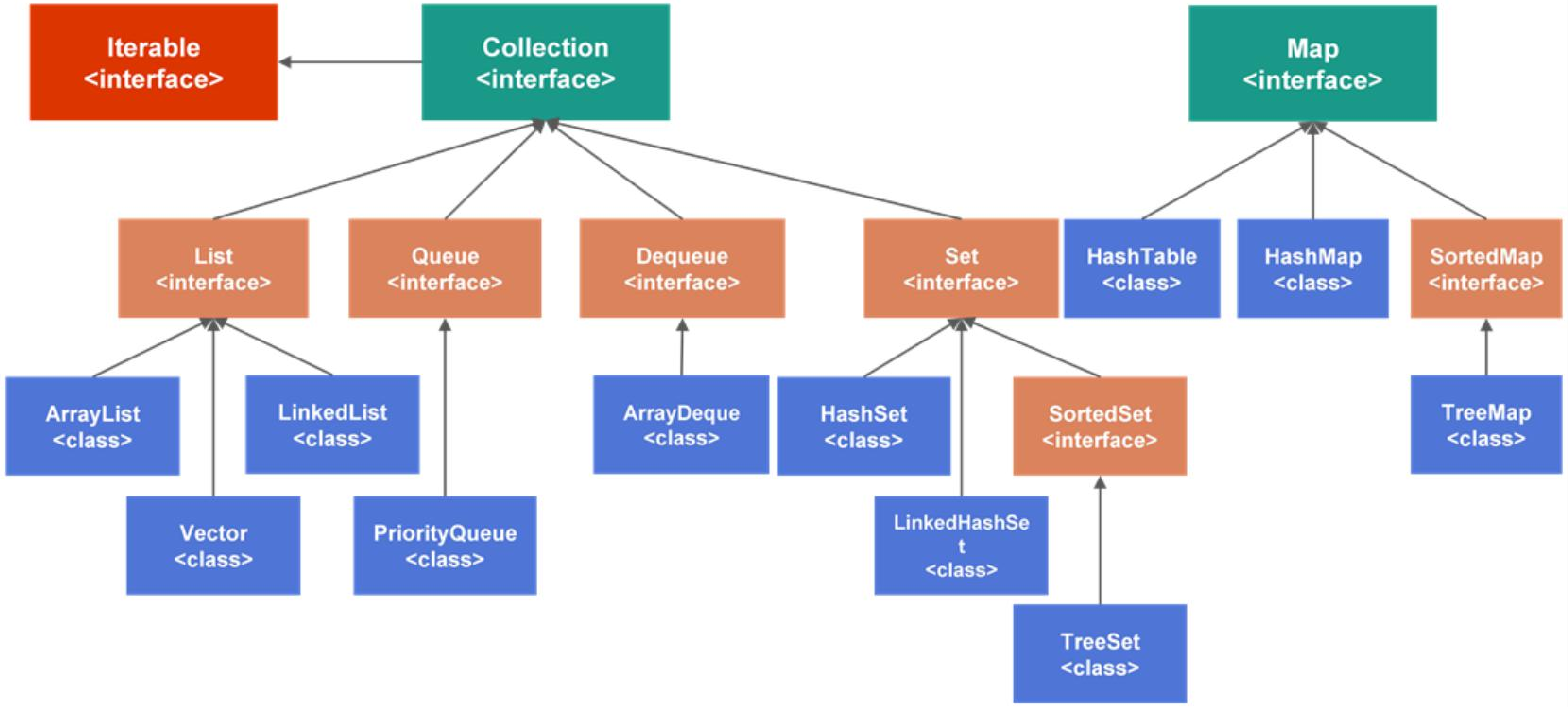
All the interfaces have their own implementation classes.

So for the list, the implementation classes are ArrayList, LinkedList, and Vector.

Same there are implementation class for Set.

Map is a different interface that does not extend Collection interface, but it is in the Java utility package.

Collection Framework Hierarchy



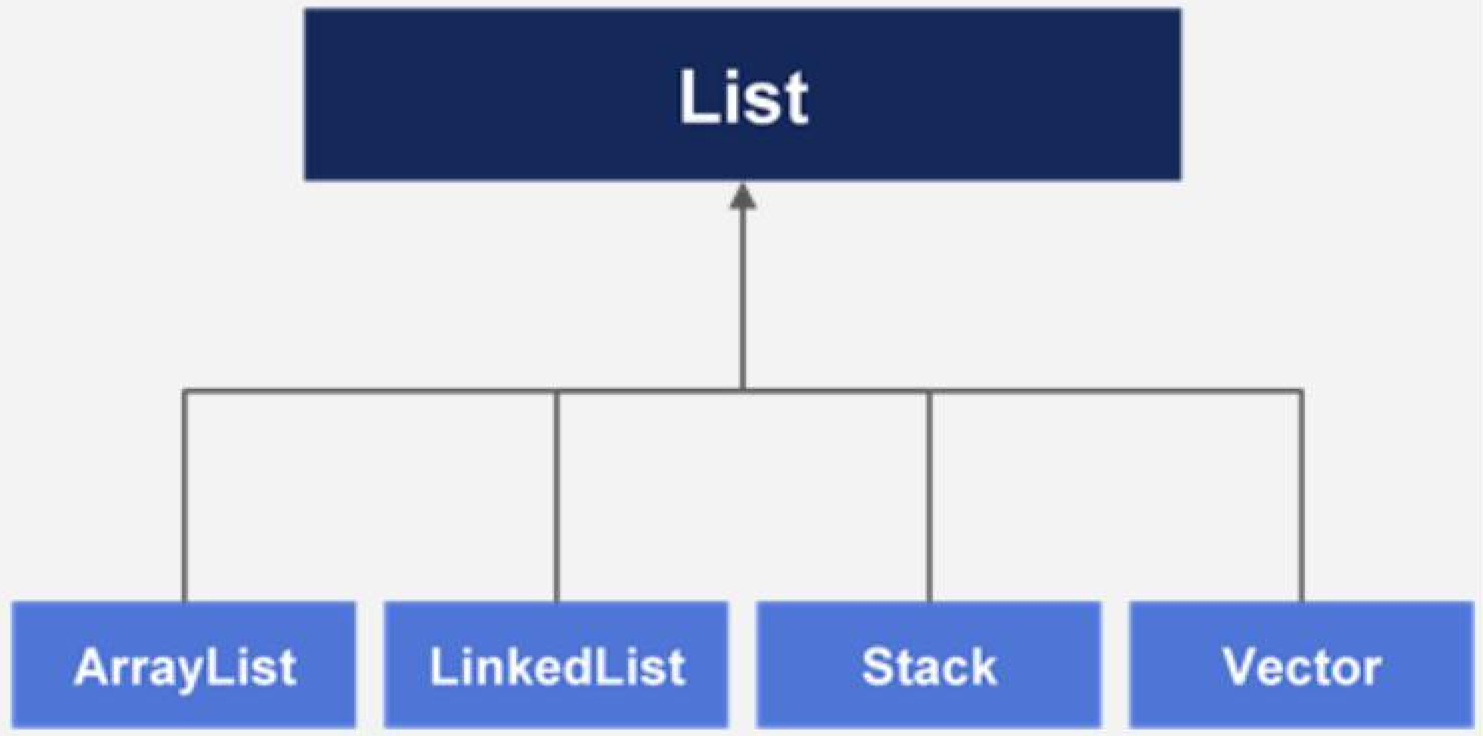
Menu

00:04 00:00:12 04/ 12

Slide Note

List Interface Implementations

- Implementation classes for the List Interface:
  - **ArrayList** - The ArrayList class provides the implementation of the list interface. It enables you to create a resizable array. It is efficient at searching for an object in the list and inserting the object at the end of the list.
  - **LinkedList** - The LinkedList class provides the implementation of the list interface. It enables you to create a doubly-linked list. It allows you to move forward or backward direction.
  - **Vector** - The vector class is similar to the ArrayList and LinkedList classes. The methods of the vector class are synchronized.



Menu

Slide Note

List Interface Methods

A few methods are shown below:

- `boolean add(Object o)` - Adds the specified element to the end of the list
- `add(int index, E element)` - Inserts the specified element at the specified position in this list
- `set(int index,E element)` - Replaces the element at the specified position in this list with the specified element
- `remove(int index)` - Removes the element at the specified position in this list
- `indexOf(Object o)` - Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element
- `get(int index)` - Returns the element at the specified position in this list

Menu

00:06:00:00:12

Slide Note

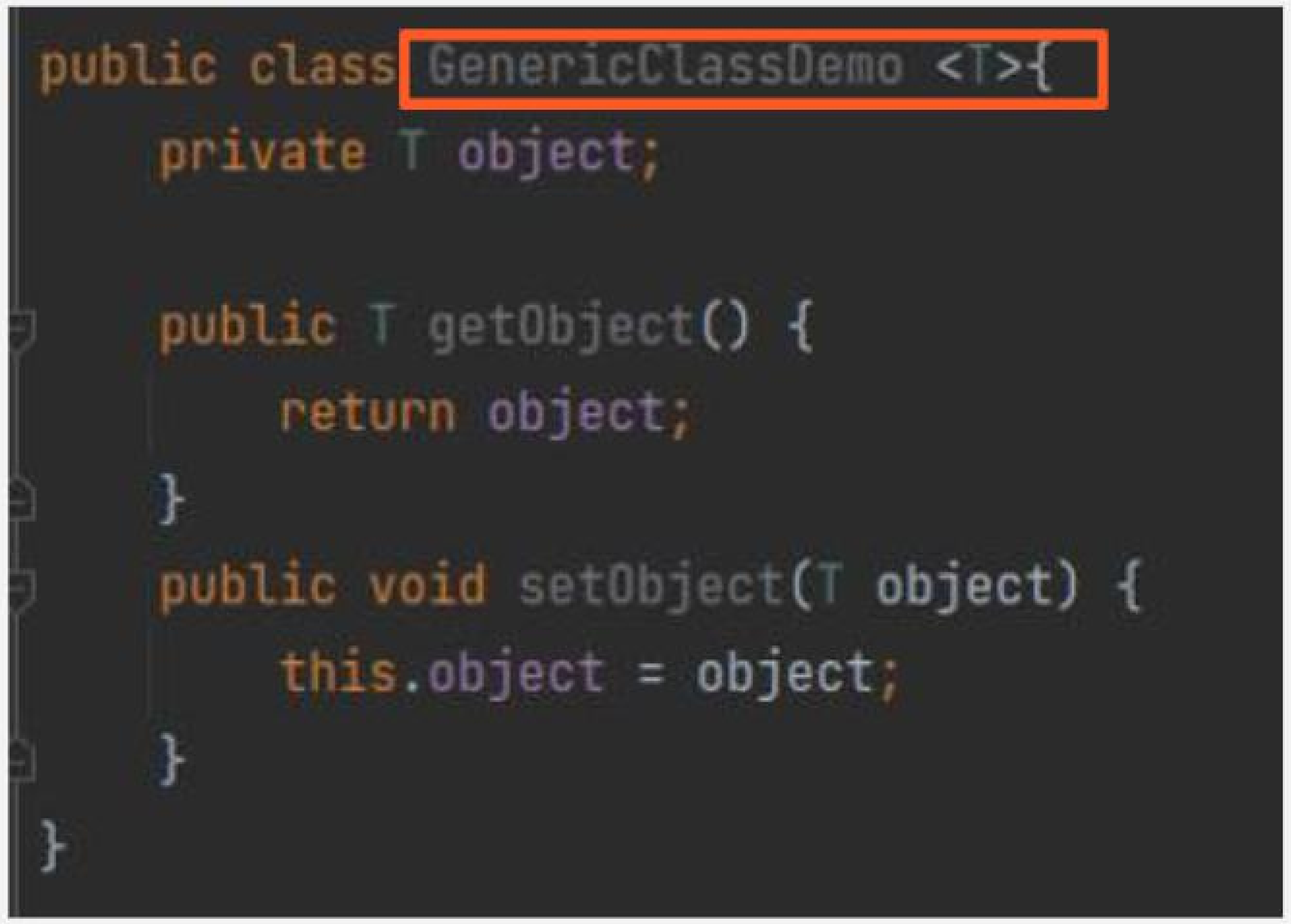
Generics

- Generics are referred to as parameterized types. Parameterized types are important because they enable you to create classes, interfaces, and methods in which the type of data upon which they operate is specified as a parameter.
- The generic collections were introduced in **Java 5 Version.** Generic collections **avoid type-casting,** are **type-safe,** and are checked at **compile time**.
- Generic collections allow the data types to be passed as parameters to classes where compatibility is checked at compile-time.
- A generic class:
  - Provides type safety to code
  - Eliminates runtime exceptions
  - Provides a cleaner and easier way to write code
  - Reduces the need for casting

Menu

00:07:00:00:12 07/ 12

Slide Note



```
public class GenericClassDemo <T>{
    private T object;

    public T getObject() {
        return object;
    }
    public void setObject(T object) {
        this.object = object;
    }
}
```

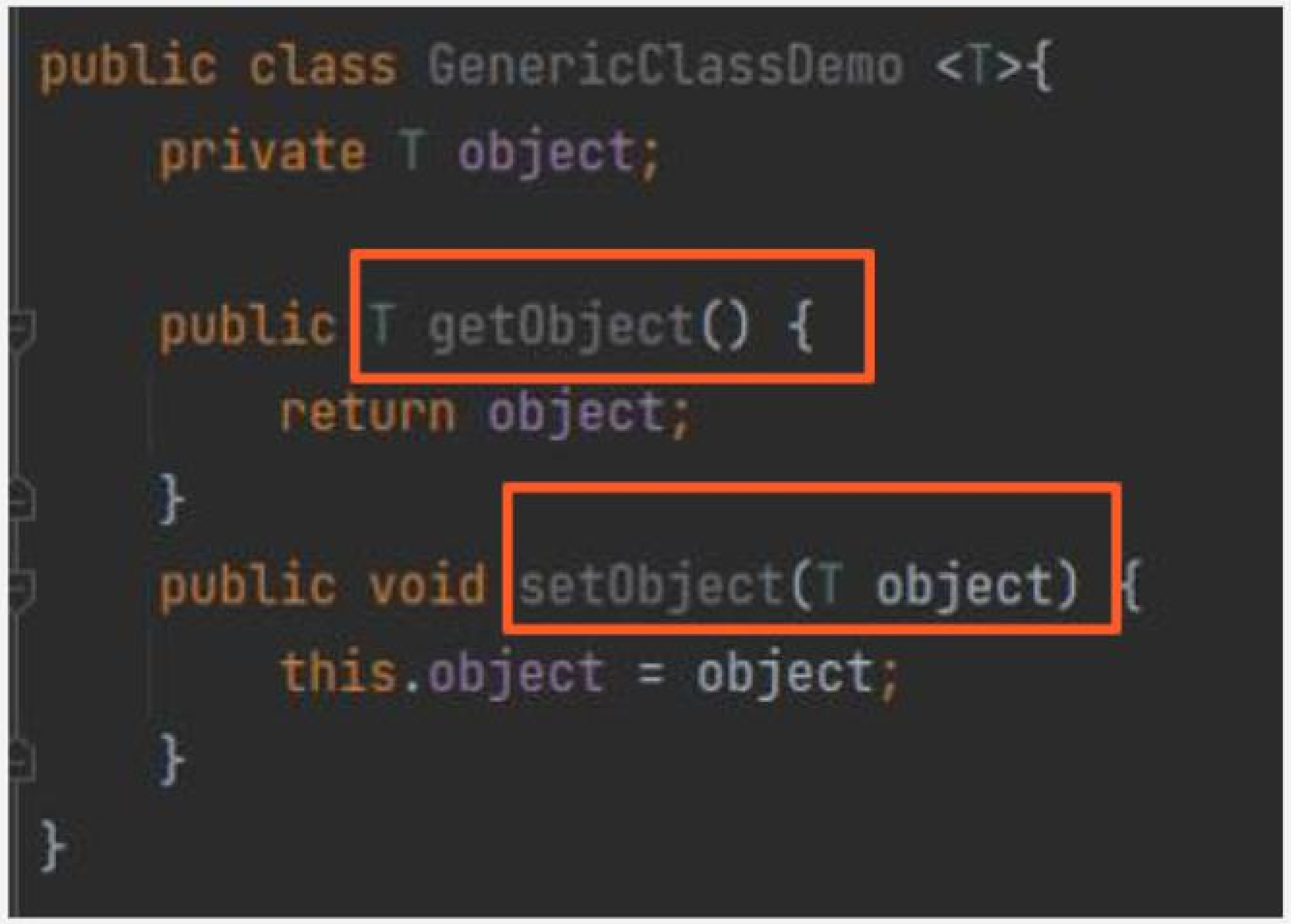
Generic Class

- Generics enable you to generalize classes.
- In the declaration of a generic class, the name of the class is followed by a type parameter section.
- GenericClassDemo<T> is a generic class.
- <> The angular brackets here represent a type parameter, which can have one or more types of parameters separated by commas.
- Here, a variable named T has been added to the class definition, surrounded by the angular <> brackets. This T variable stands for “type” and can represent any type.
- This GenericClassDemo class can be used to set and get any type of object (e.g., integer, string, double, float, etc.)

Menu

00:08 00:00:12 08/ 12

Slide Note



```
public class GenericClassDemo <T>{
    private T object;

    public T getObject() {
        return object;
    }
    public void setObject(T object) {
        this.object = object;
    }
}
```

Generic Method

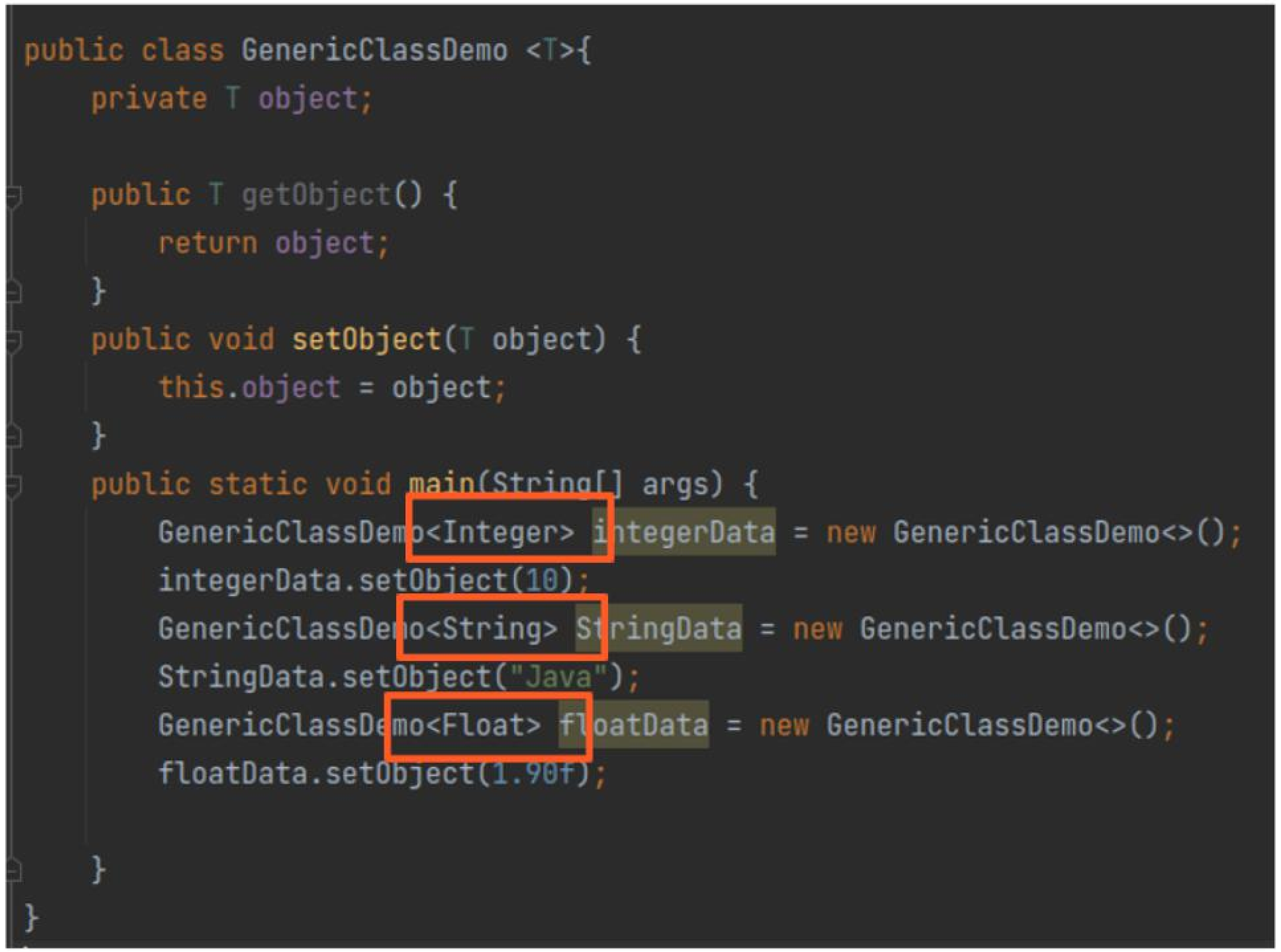
- In a generic class, a method can use the type parameter of the class, which automatically makes the method generic.
- Here T represents any type, which can be a method parameter or even a method return type.
- The other advantage is the avoidance of code duplication. Without generics, they would have to rewrite the same code for different types. With generics, you do not have to do this.

Menu

00:09 00:00:12 09/ 12

Slide Note

Different Objects Passed in the Same Class



```
public class GenericClassDemo <T>{
    private T object;

    public T getObject() {
        return object;
    }
    public void setObject(T object) {
        this.object = object;
    }
    public static void main(String[] args) {
        GenericClassDemo<Integer> integerData = new GenericClassDemo<>();
        integerData.setObject(10);
        GenericClassDemo<String> StringData = new GenericClassDemo<>();
        StringData.setObject("Java");
        GenericClassDemo<Float> floatData = new GenericClassDemo<>();
        floatData.setObject(1.90f);

    }
}
```

- Here, setter and getter can be called for any type of object.
- The generic type should only be of wrapper class, not the primitive type.
- As the code indicates, the first generic type is integer, which is thus inside the set; you can set the integer value.

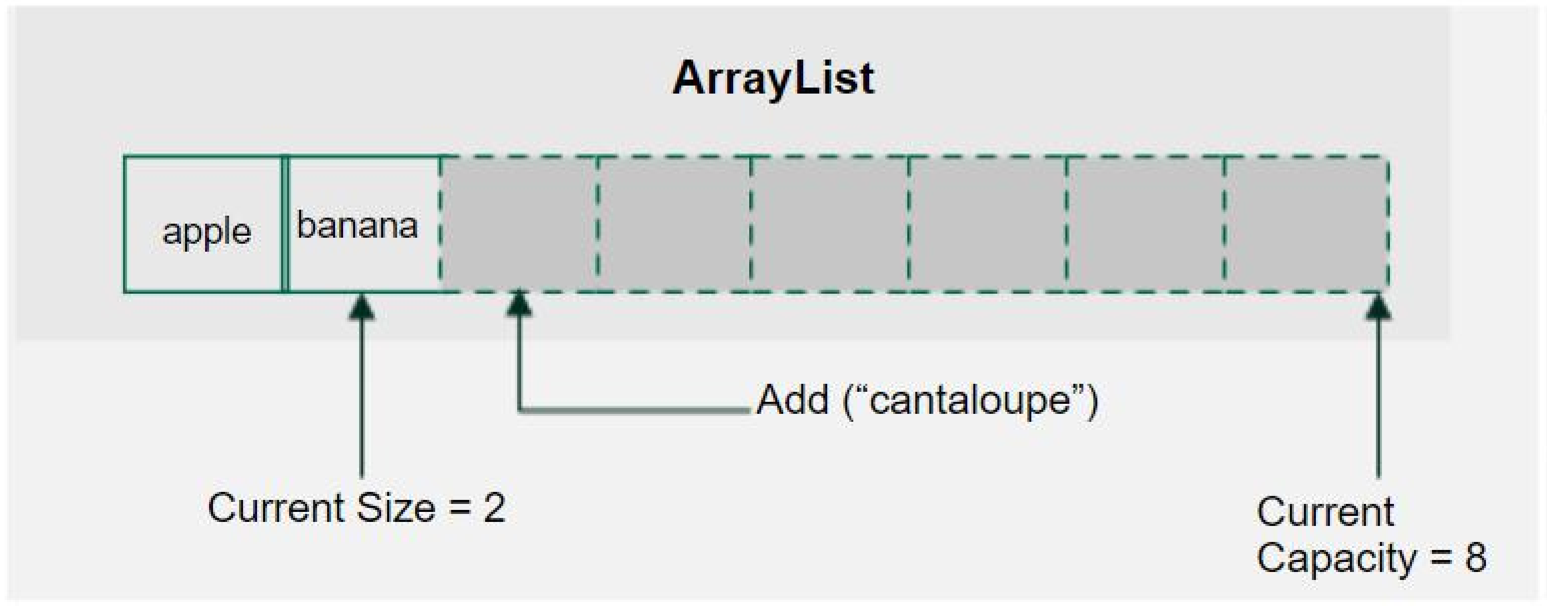
Menu

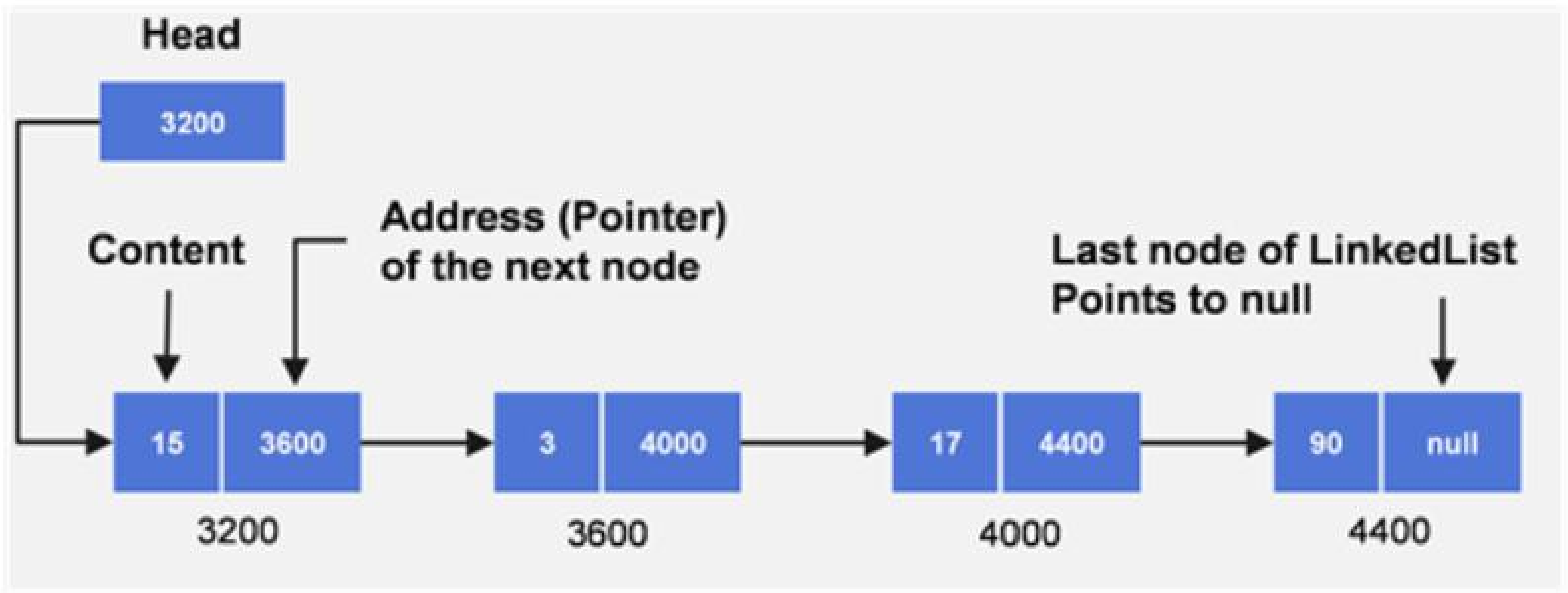
00:10 00:00:12 10/ 12

Slide Note

ArrayList vs. LinkedList

ArrayList	LinkedList
This class enables you to create a resizable array that is dynamically growable to store elements in it.	This class enables you to create a doubly-linked list and store elements in it.
Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array can be shrunk. The array list stores data in continuous memory locations.	LinkedList is a linear data structure.elements are not stored in contiguous locations like an ArrayList. They are linked with each other using pointers. Each element of the LinkedList has a reference (address/pointer) to the next element of the LinkedList.
When the demand is for quick insert and retrieval of data, an ArrayList should be used.	When demand for manipulation of stored data LinkedList works better.




Head
3200
Content
Address (Pointer) of the next node
Last node of LinkedList Points to null
15
3600
3
4000
17
4400
90
null
3200
3600
4000
4400

Menu

00:11:00:00:12

11/ 12