• Chapter 3: Core Java APIs

Understanding an ArrayList

Creating an ArrayList

```
import java.util.*          // import whole package including ArrayList
import java.util.ArrayList; // import just ArrayList

ArrayList list1 = new ArrayList();
ArrayList list2 = new ArrayList(10);
ArrayList list3 = new ArrayList(list2);
```

Java 5 introduced generics, which allow you to specify the type of class that the ArrayList will contain.

```
ArrayList<String> list4 = new ArrayList<String>();
ArrayList<String> list5 = new ArrayList<>();

List<String> list6 = new ArrayList<>();
ArrayList<String> list7 = new List<>(); // DOES NOT COMPILE
```

ArrayList implements an interface called List. In other words, an ArrayList is a List. You will learn about interfaces in Chapter 5. In the meantime, just know that you can store an ArrayList in a List reference variable but not vice versa.

• Chapter 3: Core Java APIs

Understanding an ArrayList

add()

The add() methods insert a new value in the ArrayList. The method signatures are as follows:

boolean add(E element)
void add(int index, E element)

Don't worry about the boolean return value. It always returns true. It is there because other classes in the collections family need a return value in the signature when adding an element.

```
ArrayList list = new ArrayList();
list.add("hawk");           // [hawk]

list.add(Boolean.TRUE);     // [hawk, true]
System.out.println(list);   // [hawk, true]

ArrayList<String> safer = new ArrayList<>();
safer.add("sparrow");
safer.add(Boolean.TRUE);    // DOES NOT COMPILE
```

```
4: List<String> birds = new ArrayList<>();
5: birds.add("hawk");              // [hawk]
6: birds.add(1, "robin");          // [hawk, robin]
7: birds.add(0, "blue jay");       // [blue jay, hawk, robin]
8: birds.add(1, "cardinal");       // [blue jay, cardinal, hawk, robin]
9: System.out.println(birds);      // [blue jay, cardinal, hawk, robin]
```

Chapter 3: Core Java APIs

Understanding an ArrayList

remove()

The remove() methods remove the first matching value in the ArrayList or remove the element at a specified index. The method signatures are as follows:

boolean remove(Object object)
E remove(int index)

```
3: List<String> birds = new ArrayList<>();
4: birds.add("hawk");     // [hawk]
5: birds.add("hawk");     // [hawk, hawk]
6: System.out.println(birds.remove("cardinal")); // prints false
7: System.out.println(birds.remove("hawk")); // prints true
8: System.out.println(birds.remove(0)); // prints hawk
9: System.out.println(birds);     // []
```

• Chapter 3: Core Java APIs

Understanding an ArrayList

set()

The set() method changes one of the elements of the ArrayList without changing the size. The method signature is as follows:

E set(int index, E newElement)

The E return type is the element that got replaced. The following shows how to use this method:

```
15: List<String> birds = new ArrayList<>();
16: birds.add("hawk");                    // [hawk]
17: System.out.println(birds.size());     // 1
18: birds.set(0, "robin");                // [robin]
19: System.out.println(birds.size());     // 1
20: birds.set(1, "robin");                // IndexOutOfBoundsException
```

Chapter 3: Core Java APIs

Understanding an ArrayList

isEmpty() and size()

The isEmpty() and size() methods look at how many of the slots are in use. The method signatures are as follows:

boolean isEmpty()

int size()

The following shows how to use these methods:

```
System.out.println(birds.isEmpty());     // true
System.out.println(birds.size());     // 0
birds.add("hawk");                    // [hawk]
birds.add("hawk");                    // [hawk, hawk]

System.out.println(birds.isEmpty());     // false
System.out.println(birds.size());     // 2
```

Chapter 3: Core Java APIs

Understanding an ArrayList

clear()

The clear() method provides an easy way to discard all elements of the ArrayList. The method signature is as follows:

void clear()

The following shows how to use this method:

```
List<String> birds = new ArrayList<>();
birds.add("hawk");                      // [hawk]
birds.add("hawk");                      // [hawk, hawk]
System.out.println(birds.isEmpty());    // false
System.out.println(birds.size());       // 2
birds.clear();                          // []
System.out.println(birds.isEmpty());    // true
System.out.println(birds.size());       // 0
```

After we call `clear()`, `birds` is back to being an empty `ArrayList` of size 0.

Chapter 3: Core Java APIs

Understanding an ArrayList

contains()

The contains() method checks whether a certain value is in the ArrayList. The method signature is as follows:

boolean contains(Object object)

The following shows how to use this method:

```
List<String> birds = new ArrayList<>();
birds.add("hawk");                          // [hawk]
System.out.println(birds.contains("hawk")); // true
System.out.println(birds.contains("robin")); // false
```

This method calls equals() on each element of the ArrayList to see whether there are any matches. Since String implements equals(), this works out well.

• Chapter 3: Core Java APIs

Understanding an ArrayList

equals()

Finally, ArrayList has a custom implementation of equals() so you can compare two lists to see if they contain the same elements in the same order.

boolean equals(Object object)

The following shows how to use this method:

```
31: List<String> one = new ArrayList<>();
32: List<String> two = new ArrayList<>();
33: System.out.println(one.equals(two));      // true
34: one.add("a");                            // [a]
35: System.out.println(one.equals(two));     // false
36: two.add("a");                            // [a]
37: System.out.println(one.equals(two));     // true
38: one.add("b");                            // [a,b]
39: two.add(0, "b");                        // [b,a]
40: System.out.println(one.equals(two));     // false
```

• Chapter 3: Core Java APIs

Understanding an ArrayList

Wrapper Classes

Primitive type	Wrapper class	Example of constructing
boolean	Boolean	`new Boolean(true)`
byte	Byte	`new Byte((byte) 1)`
short	Short	`new Short((short) 1)`
int	Integer	`new Integer(1)`
long	Long	`new Long(1)`
float	Float	`new Float(1.0)`
double	Double	`new Double(1.0)`
char	Character	`new Character('c')`

• Chapter 3: Core Java APIs

Understanding an ArrayList

Wrapper Classes

Wrapper class	Converting String to primitive	Converting String to wrapper class
Boolean	`Boolean.parseBoolean("true");`	`Boolean.valueOf("TRUE");`
Byte	`Byte.parseByte("1");`	`Byte.valueOf("2");`
Short	`Short.parseShort("1");`	`Short.valueOf("2");`
Integer	`Integer.parseInt("1");`	`Integer.valueOf("2");`
Long	`Long.parseLong("1");`	`Long.valueOf("2");`
Float	`Float.parseFloat("1");`	`Float.valueOf("2.2");`
Double	`Double.parseDouble("1");`	`Double.valueOf("2.2");`
Character	None	None