

How to swap objects in Java?

Description: Swapping refers to exchanging properties of objects in Java. Collections class of Java util package has the built in swap() method to swap elements of specified positions in a given list.

Object swapping in Java collections refers to swapping two specified elements of a collection using the swap function(). While the swap() method of Collections class is for collections from Collection interface, swapping custom/user defined objects need the logic specifically implemented by the programmer.

Swapping objects is different from swapping primitive data types.

Since Java uses pass by value when passing parameters to the method, the mere swapping of the objects of a Class in a separate method does not work. This is explained as below.

```
public class ObjectSwapper {
    public static void main(String[] args) {
        InnerClass obj1 = new InnerClass(12345);
        InnerClass obj2 = new InnerClass(11111);
        swap(obj1, obj2);
        System.out.println("Obj1 ID value : " + obj1.id + "\n" + "Obj2 ID
value : " + obj2.id);
    }
    static void swap(InnerClass x, InnerClass y) {
        InnerClass temp;
        temp = x;
        x = y;
        y = temp;
    }
    static class InnerClass {
        public int id;
        public InnerClass(int Id) {
            this.id = Id;
        }
    }
}
```

The output:

```
Console x
<terminated> ObjectSwapper [Java Application] C:\Users\ASUS\AppData\Local\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0
Obj1 ID value : 12345
Obj2 ID value : 11111
```

When we are calling the swap method and passing the objects, we pass the copy of the object with all the property values. That is why it is called pass by value. The swap function actually swaps the objects but it impacts only to the copy of the original objects. So, the original objects are kept unchanged. The solution to this would be to use a wrapper class to wrap the objects as wrapper class properties and swap.

Swapping objects with a wrapper class

```
public class ObjectSwapper {
    public static void main(String[] args) {

        InnerClass obj1 = new InnerClass(12345);
        InnerClass obj2 = new InnerClass(11111);
        objectWrapper wrapperObj1 = new objectWrapper(obj1);
        objectWrapper wrapperObj2 = new objectWrapper(obj2);
        // Values before swapping
        System.out.println("WrapperObj1 InnerClass ID value : " +
wrapperObj1.innerObject.id + "\n" + "WrapperObj2 InnerClass ID value : "
        + wrapperObj2.innerObject.id + "\n");

        swap(wrapperObj1, wrapperObj2);

        // Values after swapping
        System.out.println("WrapperObj1 InnerClass ID value : " +
wrapperObj1.innerObject.id + "\n" + "WrapperObj2 InnerClass ID value : "
        + wrapperObj2.innerObject.id);
    }
}
```

```

        static void swap(objectWrapper wrapperObj1, objectWrapper
wrapperObj2) {
            InnerClass temp;
            temp = wrapperObj1.innerObject;
            wrapperObj1.innerObject = wrapperObj2.innerObject;
            wrapperObj2.innerObject = temp;
        }

        static class InnerClass {
            public int id;
            public InnerClass(int Id) {
                id = Id;
            }
        }

        static class objectWrapper {
            InnerClass innerObject;
            public objectWrapper(InnerClass objInnner) {
                this.innerObject = objInnner;
            }
        }
    }
}

```

Here, we have used a wrapper class which has a property of the type of the objects we need to swap. By using a simple swap method, it swaps the content of the XYZ objects of the wrapper class objects. For any further implementation of the swapped XYZ objects, we can use wrapperObj1.xyz and wrapperObj2.xyz accordingly.

The output:

```

Console x
<terminated> ObjectSwapper [Java Application] C:\Users\ASUS\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0
WrapperObj1 InnkerClass ID value : 12345
WrapperObj2 InnkerClass ID value : 11111

WrapperObj1 InnkerClass ID value : 11111
WrapperObj2 InnkerClass ID value : 12345

```

The built-in swap() method in Java

Java Collections class in the Java util package has a built-in static method for element swapping called swap(). The java.util is a utility class which contains static methods that can operate on collections from Collection interface. Collections.swap() is used to swap elements of specified positions in a given list.

Since this is a static method, trying to invoke it as an instance method is unnecessary but will not show as an error. The return type of the swap() method is void so it will not return anything. Rather than writing custom implementations as above examples we can use this built-in swap() method on the collections from the Collection interface.

Syntax of the method:

```
swap(List<?> list, int a, int b)
```

Parameters:

list - The list we need to swap elements.

a - index of an element to be swapped.

b - index of another element to be swapped.

Note that this method will throw an IndexOutOfBoundsException if either a or b is out of the range of the list size. The maximum index of the list is one less than the size of the list. So an index value more than that will cause this exception to be thrown.

An example of how this is implemented on an ArrayList is as below.

swap() method on an ArrayList

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class ArrayListSwapper {
    public static void main(String[] args) {
        try {
            List<String> fruits = new ArrayList<String>();
            fruits.add("Mango");
            fruits.add("Papaya");
            fruits.add("Apple");
```

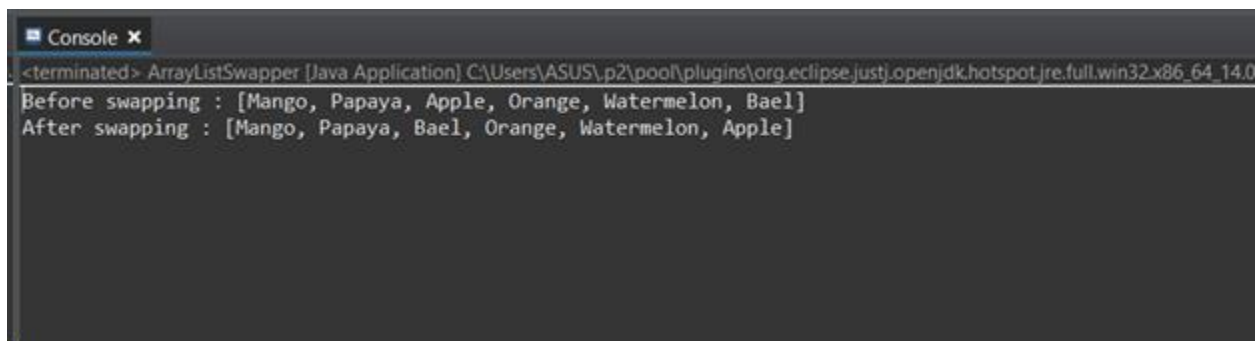
```

        fruits.add("Orange");
        fruits.add("Watermelon");
        fruits.add("Bael");
        System.out.println("Before swapping : " + fruits);
        //using Collection.swap() method
        Collections.swap(fruits, 2, 5);
        System.out.println("After swapping : " + fruits);
    } catch (IndexOutOfBoundsException e) {
        System.out.println("Index Out of Bound Exception thrown : " +
e);
    }
}
}
}

```

The ArrayList of fruits is populated with 6 elements. We swap the 2nd and 5th element with each other using function swap() from the Collections class. The best practice is to write within a try-catch block as can throw an IndexOutOfBoundsException at run time.

The output:



The screenshot shows a console window titled "Console x" with the following output:

```

<terminated> ArrayListSwapper [Java Application] C:\Users\ASUS\AppData\Local\Temp\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_14.0
Before swapping : [Mango, Papaya, Apple, Orange, Watermelon, Bael]
After swapping : [Mango, Papaya, Bael, Orange, Watermelon, Apple]

```

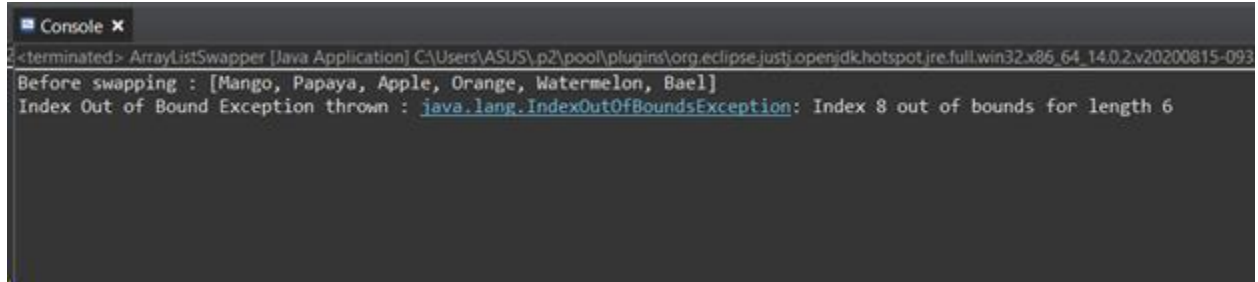
Following is an implementation where the exception is thrown. We are trying to swap 2nd and 8th elements where the list size is 6.

```

Collections.swap(fruits, 2, 8);

```

For this the output is:

A screenshot of a Java IDE console window. The title bar says "Console x". The text in the console is as follows:

```
<terminated> ArrayListSwapper [Java Application] C:\Users\ASUS\AppData\Local\Temp\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-093
Before swapping : [Mango, Papaya, Apple, Orange, Watermelon, Bael]
Index Out of Bound Exception thrown : java.lang.IndexOutOfBoundsException: Index 8 out of bounds for length 6
```

Conclusion

In this tutorial we discussed a couple of swapping techniques used in Java. One of them is using a wrapper class and the other is using `Collection.swap()` method. It is important to remember that exception handling should be done when using the function `swap()` as it throws a runtime exception.