16. OOP. JAVA

POJO, JavaBeans.
Collections, Deep Copy Example

POJO, JavaBeans

- Plain Old Java Object

https://en.wikipedia.org/wiki/Plain_Old_Java_Object

- JavaBeans

https://en.wikipedia.org/wiki/JavaBeans

https://docs.oracle.com/javase/tutorial/javabeans/

Class Data with Copy Constructor

```
package data;
public class Data {
  private String name;
  private int i;
  /** Copy constructor is simple for immutable or primitive objects. */
  public Data(final Data data) {
     this(data.name, data.i);
  }
  public Data(String name, int i) {
     this.name = name; this.i = i;
  }
  public String getName() {
     return name;
  }
  public void setName(String name) {
     this.name = name;
  }
  public int getI() {
     return i;
  }
  public void setI(int i) {
     this.i = i;
  }
  public String toString() {
     return "[name = " + name + ", i = " + i + "]";
  }
}
```

Shallow Copy Example

```
package copy.shallow;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import data.Data;
public class Main {
  public static void main(String[] args) {
     // Create data[].
     Data[] data = { new Data("one", 1), new Data("two", 2) };
     // Create first collection using addAll().
     Collection<Data> source = new ArrayList<>();
     source.addAll(Arrays.asList(data));
     // Create second collection using ArrayList copy constructor.
     Collection<Data> copy = new ArrayList<>(source);
     // Print result.
     System.out.println(Arrays.toString(data));
     System.out.println(source);
     System.out.println(copy);
     // Change data...
     for (Data x : data) {
        x.setName(x.getName() + " is changed");
        x.setI(x.getI() + 100);
     }
     // ...or change copy.
     // copy.forEach((x) -> {
     // x.setName(x.getName() + " copy changed");
     // x.setI(x.getI() + 200);
     // });
     // Print result.
     System.out.println(Arrays.toString(data));
     System.out.println(source);
     System.out.println(copy);
  }
}
```

Shallow Copy Example

Results

```
[[name = one, i = 1], [name = two, i = 2]]
[[name = one, i = 1], [name = two, i = 2]]
[[name = one, i = 1], [name = two, i = 2]]
[[name = one is changed, i = 101], [name = two is changed, i = 102]]
[[name = one is changed, i = 101], [name = two is changed, i = 102]]
[[name = one is changed, i = 101], [name = two is changed, i = 102]]
```

Deep Copy Example

```
package copy.deep;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import data.Data;
public class Main {
  public static void main(String[] args) {
     // Create data[].
     Data[] data = { new Data("one", 1), new Data("two", 2) };
     // Create first collection using Data copy constructor.
     Collection<Data> source = new ArrayList<>();
     for (Data x : data) {
        source.add(new Data(x));
     }
     // Create second collection using Data copy constructor.
     Collection<Data> copy = new ArrayList<>();
     Arrays.stream(data).forEach(x -> copy.add(new Data(x)));
     // Print result.
     System.out.println(Arrays.toString(data));
     System.out.println(source);
     System.out.println(copy);
     // Change data...
     for (Data x : data) {
        x.setName(x.getName() + " is changed");
        x.setI(x.getI() + 100);
     }
     // ...and change copy.
     copy.forEach((x) -> {
        x.setName(x.getName() + " copy changed");
        x.setI(x.getI() + 200);
     });
     // Print result.
     System.out.println(Arrays.toString(data));
     System.out.println(source);
     System.out.println(copy);
  }
}
```

Deep Copy Example

Results

```
[[name = one, i = 1], [name = two, i = 2]]
[[name = one, i = 1], [name = two, i = 2]]
[[name = one, i = 1], [name = two, i = 2]]
[[name = one is changed, i = 101], [name = two is changed, i = 102]]
[[name = one, i = 1], [name = two, i = 2]]
[[name = one copy changed, i = 201], [name = two copy changed, i = 202]]
```