Topics

1. References
2. Polymorphism
3. Memory Management
4. Imports and Libraries
5. Serialization

EMP session links (if you want to have some more practices)

1. EMP 10-06: Topics including Polymorphism/Casting/super/this
2. EMP 10-08: Topics including Polymorphism/Casting/shallow and deep copy/dot notation
3. EMP 10-13: Topics including libraries and imports/type inference/serialization

Practice Problem

- Define a class named *Thing* with two private fields, a String *name* and a String *type*.
- Write a public constructor with two String inputs that update the values of these fields.
- Create a public class method *isReference* that takes in two Thing objects, and returns *True* only if both references are equal. You should *assert* that your objects are not null.
- Write a public class method named *copy* that takes in an array of Things and returns an array containing two Things arrays, one with a shallow copy of all the elements in the input array and another that returns a deep copy of all the elements in the input array. (So, your function should return a 2D array). *(Assume that Thing has an existing copy constructor)*
- Write a public class method named *randomThings* that takes in an array of Thing and prints out (does not return!) 4 random "things" from that array. (You might wanna import a library for this)

Solution

▶ Spoiler!

```
import java.util.Random
public class Thing implements Cloneable {
   private String name;
   private String type;
   public class Thing(String n, String t) {
    name = n;
    type = t;
   }

   public class Thing(Thing other) {
    // assume the implementation is given
   }

   public Object clone() throws CloneNotSupportedException {
    return super.clone();
   }

   public static boolean isReference(Thing a, Thing b) {
    assert a != null;
    assert b != null;

    return a == b;
```

```java
    }

    public static Thing[][] copy(Thing[] arr) {
     assert arr != null;
     Thing[] shallow = new Thing[arr.length];
     Thing[] deep = new Thing[arr.length];
     for (int i = 0; i < arr.length; i++) {
       shallow[i] = (Thing) arr[i].clone();
       deep[i] = new Thing(arr[i]);
     }
     Thing[][] output = {shallow, deep};
     return output;
     }

     public static void randomThings(Thing[] arr) {
       //You will tell me this step by step
     }
}
```