Maps (aka Key-Value dictionary)
Representing sound.
Reasoning about Shallow and Deep Copy
Inheritance (Extending Existing Classes)

// Complete *.equals* and write the two Ghost constructors so we can make ghosts such as -
new Ghost(); // creates ghost at (1, random Y position)
new Ghost(new int[] {15,20}); // ghost at (15,20)

```
public class Ghost {
 private static int count=0;
 private static int nextId() {
   count ++; // first ghost will have an id of 1
   return count;
 }

 // each ghost has an x,y and unique id
 private int x = 1, y = 2 + (int)(Math.random() * 10);
 private int id;// Your c'tor sets id to a unique value

 public String toString() {
     return "Ghost #" + id + ": " + x + "," + y;
 }

 public boolean equals(Object other) {
    if (other instanceOf Ghost) {
       Ghost g = (Ghost) other; // zombie
       return _______
   } else return false;
 }
```

MAPS. (Store a value for a particular key)

```
public class CallerIdPair {
   public int _______; // the extension (a unique key)
   public String  _______; // the value (can be anything)
}

public class CallerIdMap {

// use an array of pairs
  private



  public ____ add(int extn, String name) {
// for now, assume that the extension (the key) has not already been added to this map.
// better implementations would prevent or remove/replace an existing match.










 }
   public String get(int extn) {
// return "?" if we do not know this extension's name
```

I want to create and edit sounds. Complete the Sound class below so that I can write the following ode:

```
 1 Sound s1 = new Sound("Crash", new byte[] { ...} );
 2 Sound deepCopy = new Sound(s1, true); // also copies array
 3 Sound shallowCopy = new Sound(s1,false); // shares the dataarray
 4 shallowcopy.silence(180, 190); // zero out samples 180 to 190 inclusive
 5 shallowcopy.cut(163, 200); // Shorten to samples 163 to 200 inclusive.
 6 s1.playSound(); // shortened and/or with 11 zero samples?
```

```
class Sound {
   private String name;
   private byte[] data;
   public int getSize() { return data.length; }
   public byte getByte(int i) { return data[i]; }

   public void silence(int start, int end) {
      for (int i = start, i <= end ; i ++) data[i] = 0;
   }
   public void cut(int start, int end) { // This instance method needs fixing
     int newLength = end - start;
     byte[] cutData = new byte[newLength];
     for (int i = 0; i < newLength ; i ++)
        cutData[i] = data[i + start];
}
```

First Constructor:

Copy Constructor:

Show how s1,deepCopy and shallowCopy share arrays (or not) after line 3 completes

O	R	I	G	I	N	A	L		A	R	R	A	Y

s1.data

shallowCopy.data

deepCopy.data

How does this change after line 4 executes? shallowcopy.silence(180,190);
How does this change after line 5 executes? shallowcopy.cut(163,200)

THIS SPACE FOR INHERITANCE NOTES

```
class FastGhost extends Ghost {...}

Ghost g = new FastGhost();
g.move();
```