Course: Data Structures

Assignment number: 4

Name: Aviv Laban

Student ID: 200358976

E-mail: aviv.laban@post.idc.il

Attached HardCopy files:

1. UpTreeForest.java
2. IslandsConnectivityChecker.java

Additional classes used for Union Project:

1. Archipelago.java
2. LandType.java
3. ImageReader.java

```
public class UpTreeForest {

	private static final int ROOT = -1;
	private int up[];
	private int weight[];
	private int numberOfSets;

	public UpTreeForest(int size){
		//Creating new array to represent every element's location and
		//initalizing it to ROOT
		up = new int[size];
		for(int i = 0; i < up.length; i++){
			up[i] = -1;
		}
		//Creating new array to represent number of elements in every set
		weight = new int[size];
		for(int i = 0; i < weight.length; i++){
			weight[i] = 1;
		}

		numberOfSets = size;
	}

	public void union(int i, int j){
		//If one of the givven int is not a ROOT returnes error
		if((up[i] != ROOT) || (up[j] != ROOT)){
			throw new IllegalArgumentException("One or two of the givven
			integers aren't representatives");
		}

		if(weight[i] > weight[j]){
			weight[i] = weight[i] + weight[j];
			up[j] = i;
		}

		else{
			weight[j] = weight[j] + weight[i];
			up[i] = j;
		}

		numberOfSets--;
	}

	public int find(int i){
```

```
            int currentIndex = i;
            while(up[currentIndex] != ROOT){
                    currentIndex = up[currentIndex];
            }
            if (i != currentIndex) {
           int k = up[i];
           while (k != currentIndex) {
             up[i] = currentIndex;
             i = k;
             k = up[k];
           }
            }

            return currentIndex;
    }

    public int getNumDisjointSets(){

            return numberOfSets;
    }

}
```

```
import java.io.IOException;



public class IslandsConnectivityChecker {

	private UpTreeForest UTF;
	private Archipelago image;
	private int numComponents;

	public IslandsConnectivityChecker(String bmpPath){
		try {
			image = ImageReader.readImage(bmpPath);
		} catch (IOException e) {
			e.printStackTrace();
		}
		UTF = new UpTreeForest(image.getHeight() * image.getWidth());
		numComponents = (image.getHeight() * image.getWidth());
		//Going over the image and connect islands and seas
		for(int i = 0; i < image.getWidth(); i++){
			for(int j = 0; j < image.getHeight(); j++){

				if(i < (image.getWidth() - 1)){
					connectCoords(i, j, i+1, j);
				}

				if(j < (image.getHeight() - 1)){
					connectCoords(i, j, i, j+1);
				}

			}
		}


	}

	public void connectCoords(int x1, int y1, int x2, int y2){
		int firstLocation = UTF.find(position(x1, y1));
		int secondLocation = UTF.find(position(x2, y2));
		//Connects two elements only if the two elements have the same type
		//and have a different ROOT
		if((image.getLandType(x1, y1) == image.getLandType(x2, y2)) &&
				(firstLocation != secondLocation)){
			UTF.union(firstLocation, secondLocation);
```

```
                numComponents--;
            }

        }

        public boolean areConnected(int x1, int y1, int x2, int y2){
            if(UTF.find(position(x1, y1)) == UTF.find(position(x2, y2))){
                return true;
            }

            return false;
        }

        public int getNumComponents(){

            return numComponents;
        }

        private int position(int x1, int y1){
            //Finds the location of (x,y) in a single array - the tree.
            int position = ((image.getWidth()*y1) + x1);
            return position;
        }


}
```