```java
//Savannah Swan
/*Class CS1450
* Due: March 5, 2025
* Assignment 5
*This assignment goes through an array of numbers and puts them onto a stack.
*Then, it replaces the 0's with 10's. Next, the method reads from four files.
*2 int files and 2 string files. It places them into a separate stack. Then,
*it sorts the arrays. Then, it merges the sorted arrays.
*/
import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.Stack;
import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
import java.io.IOException;
import java.util.Scanner;
	public class SwanSavannahAssignment5 {
	public static void main(String[] args) throws IOException {

		System.out.println("Stack Values after replacing 0's with 10");
		System.out.println("---------------------------------------");

		int[] numbers = {0, 0, 4, 3, 0, 0, 2, 1, 0, 0};

		Stack<Integer> intStack = new Stack<>();

		for (int i = 0; i < numbers.length; i++) {
			intStack.push(numbers[i]);
		}

		 replaceZerosWithTen (intStack);
```

```
        GenericStack<Integer> intStack1 = new GenericStack<>();
        GenericStack<Integer> intStack2 = new GenericStack<>();

        final String INTEGERS_ONE = "integers1.txt";

        File integersOne = new File(INTEGERS_ONE);

        Scanner integer1 = new Scanner (integersOne);

        while (integer1.hasNext()) {

        int number = integer1.nextInt();

        intStack1.push(number);

        }

        System.out.println("\nInteger1 Stack: filled with values from
file: integers1.txt");

System.out.println("------------------------------------------------------------");
        printStack (intStack1);

        final String INTEGERS_TWO = "integers2.txt";

        File integersTwo = new File(INTEGERS_TWO);

        Scanner integer2 = new Scanner (integersTwo);

        while (integer2.hasNext()) {

        int numberTwo = integer2.nextInt();

        intStack2.push(numberTwo);
```

```
            }

            System.out.println("\nInteger2 Stack: filled with values from
file: integers2.txt");

System.out.println("-----------------------------------------------------");
            printStack (intStack2);

            GenericStack<String> stringStack1 = new GenericStack<>();
            GenericStack<String> stringStack2 = new GenericStack<>();

            final String STRING_ONE = "strings1.txt";

            File stringOne = new File(STRING_ONE);

            Scanner string1 = new Scanner (stringOne);

            while (string1.hasNext()) {

            String name = string1.next();

            stringStack1.push(name);

            }
            System.out.println("\nString1 Stack: filled with values from file:
string1.txt");

System.out.println("-----------------------------------------------------");
            printStack (stringStack1);

            final String STRING_TWO = "strings2.txt";

            File stringTwo = new File(STRING_TWO);

            Scanner string2 = new Scanner (stringTwo);
```

```
            while (string2.hasNext()) {

            String name = string2.next();

            stringStack2.push(name);

            }
            System.out.println("\nString2 Stack: filled with values from file:
string2.txt");

System.out.println("-------------------------------------------------------");
            printStack (stringStack2);

            System.out.println("Sorted stack integers1: ");

System.out.println("-------------------------------------------------------");
            sortStack (intStack1);
            System.out.println("Sorted stack integers2: ");

System.out.println("-------------------------------------------------------");
            sortStack (intStack2);
            System.out.println("Sorted stack strings1: ");

System.out.println("-------------------------------------------------------");
            sortStack (stringStack1);
            System.out.println("Sorted stack strings2: ");

System.out.println("-------------------------------------------------------");
            sortStack (stringStack1);


            System.out.println("\nMerged String Stacks");

System.out.println("-------------------------------------------------------");
```

```
            System.out.println(mergeStacks (stringStack1, stringStack2));

        }//main
        public static <E extends Comparable<E>> GenericStack<E>
mergeStacks (GenericStack<E> stack1, GenericStack<E> stack2) {

            GenericStack<E> tempStack = new GenericStack<>();

            E value1 = stack1.peek();
            E value2 = stack2.peek();

            while (stack1.isEmpty() != true) {
                tempStack.push(value1);
                stack1.pop();
                }
            while (stack2.isEmpty() != true) {
                tempStack.push(value2);
                stack2.pop();
            }
            return tempStack;

        }
        public static <E extends Comparable<E> > void sortStack
(GenericStack<E> aStack) {

            Stack<E> tempStack = new Stack<>();

            while (aStack.isEmpty() != true) {

                E current = aStack.peek();

                while (!tempStack.isEmpty() &&
(aStack.peek()).compareTo(current) > 0) {
```

```
            aStack.push(tempStack.pop());
        }
        tempStack.push(current);
    }
    while (!tempStack.isEmpty()) {
        aStack.push(tempStack.pop());
    }

    printStack(aStack);
            }

public static <E> void printStack (GenericStack<E> stack) {

    GenericStack<E> tempStack = new GenericStack<>();

    while (stack.isEmpty() != true) {
            System.out.println(stack.peek());
            tempStack.push(stack.peek());
            stack.pop();

    }
    while (tempStack.isEmpty() != true) {
            stack.push(tempStack.peek());
            tempStack.pop();
    }

}
public static void replaceZerosWithTen (Stack<Integer> intStack) {

    Stack<Integer> tempStack = new Stack<>();

            while (intStack.isEmpty() != true) {

                    if (intStack.peek() == 0) {
```

```
                intStack.pop();
                tempStack.push(10);
            }
            else if(intStack.peek() == 1) {
                intStack.pop();
                tempStack.push(1);
            }
            else if (intStack.peek() == 2) {
                intStack.pop();
                tempStack.push(2);
            }
            else if (intStack.peek() == 3) {
                intStack.pop();
                tempStack.push(3);
            }
            else if (intStack.peek() == 4) {
                intStack.pop();
                tempStack.push(4);
            }
            }
        while (tempStack.isEmpty() != true) {

            if (tempStack.peek() == 10) {
                tempStack.pop();
                intStack.push(10);
            }
            else if (tempStack.peek() == 1) {
                tempStack.pop();
                intStack.push(1);
            }
            else if (tempStack.peek() == 2) {
                tempStack.pop();
                intStack.push(2);
            }
            else if (tempStack.peek() == 3) {
```

```
                        tempStack.pop();
                        intStack.push(3);
                    }
                    else if (tempStack.peek() == 4) {
                        tempStack.pop();
                        intStack.push(4);
                    }
                }
                System.out.println("Stack: " + intStack);
            }

}//SwanSavannahAssignment5
    class GenericStack <E> {

        private ArrayList<E> list1 = new ArrayList<>();

        public GenericStack() {

        }

        public boolean isEmpty() {

            return list1.isEmpty();
        }

        public int getSize() {
            return list1.size();
        }
        public E peek() {
            return list1.get(getSize()-1);

        }
        public E pop() {
            E value = list1.remove(getSize() -1);
            return value;
```

```
        }
        public void push(E value) {
                list1.add(value);

        }

}
```