Homework Assignment #9 due Tuesday April 17, 2018
Java Generics

Problem 1:

Modify the class Pair<T, S> below so that it implements Comparable<Pair<T, S>>.

To compare a pair with another, you compare the first element (using compareTo). If they are equal, you should compare the second elements and return the result of their comparison. If the first elements are not equal, the compareTo method returns the result of first.compareTo(other.getFirst()).

```
public class Pair<T, S>
{
   private T first;
   private S second;

   public Pair(T firstElement, S secondElement)
   {
      first = firstElement;
      second = secondElement;
   }

   public T getFirst() { return first; }
   public S getSecond() { return second; }
}
```

Problem 1.1:

Create a PairDemo class that demonstrates the class you just created. Create a new TreeSet and add pairs to it. The pairs should consist of people names (last name, first name). At the end, iterate through the set and print its contents.

If your Pair class was correctly implemented, the output should be displayed in alphabetic order (because a TreeSet implements a set using a binary search tree). Be sure to try a few pairs with the same last name.

Problem 1.2:

What happens if you try to add a Pair<String, Integer> to the set? What happens if you try to create a Pair<String, Rectangle>? Try it out and explain.

Problem 2:

Convert the simple `Stack` class below to a generic class that can work with any class. Currently the code only works with class `Integer`.

```
import java.util.*;

public class Stack
{
   private ArrayList<Integer> myList;

   public Stack()
   {
      myList = new ArrayList<Integer>();
   }

   public void push(Integer item)
   {
      myList.add(item);
   }

   public Integer pop()
   {
      Integer retItem = null;
      if (!myList.isEmpty())
      {
         retItem = myList.remove(myList.size() - 1);
      }
      return retItem;
   }

   public boolean  empty()
   {
      return myList.isEmpty();
   }
}
```

Here is a driver program that exercises the `Stack` class:

```
public class StackRunner
{
   public static void main(String[] args)
   {
      Stack<Integer> stk1 = new Stack<Integer>();
      stk1.push(1);
      stk1.push(2);
      stk1.push(3);
      stk1.push(4);
      while (!stk1.empty())
      {
         System.out.println(stk1.pop());
      }
   }
}
Modify the Stack and StackRunner classes so they can also create and exercise
a stack of type Character
```