COMS W3134 Homework 2

Jackson Chen

3 October 2016

1.
```
public static <AnyType> void printLots(Collection<AnyType> L,
Collection<Integer> P) {
  Iterator<AnyType> itrL = L.iterator();
  Iterator<Integer> itrP = P.iterator();
  Integer lcounter = -1;

  while (itrP.hasNext()) {
    Integer nextIndex = itrP.next();
    AnyType nextL = null;
    while (lcounter < nextIndex) {
      if (itrL.hasNext()) {
        nextL = itrL.next();
        lcounter++;
      }
      else {
        System.out.println("Out of bounds!");
        break;
      }
    }
    if (lcounter.equals(nextIndex)) {
      System.out.println(nextL);
    }
  }
}
```

2. *intersect* contains $L_1 \cap L_2$.

```
List<AnyType> intersect = new ArrayList<AnyType>();
int i = 0;
int j = 0;
AnyType history = null;

while (i < l1.size() && j < l2.size()) {
  switch (l1.get(i).compareTo(l2.get(j))) {
    case -1:
      i++;
      break;
    case 0:
      // Avoid duplicates
      if (!l1.get(i).equals(history)) {
        intersect.add(l1.get(i));
        history = l1.get(i);
      }
      i++;
      j++;
      break;
    case 1:
      j++;
      break;
  }
}
return intersect;
```

3. The first stack populates the array from the 0th index, while the second stack populates the array from the $n-1$th index.

```
public class TwoStacks <AnyType> {
  private AnyType[] arr;
  private int stack1;
  private int stack2;

  public TwoStacks(AnyType[] input) {
    this.arr = input;
    this.stack1 = 0;
    this.stack2 = input.length - 1;
  }

  public void push1(AnyType val) {
    if (stack1 > stack2) {
      throw new StackOverflowError();
    }
    else {
      arr[stack1] = val;
      stack1++;
    }
  }

  public void push2(AnyType val) {
    if (stack2 < stack1) {
      throw new StackOverflowError();
    }
    else {
      arr[stack2] = val;
      stack2--;
    }
  }
```

```
    public AnyType pop1() {
      if (stack1 == 0) {
        System.out.println("Cannot pop stack!");
        return null;
      }
      else {
        stack1--;
        AnyType tmp = arr[stack1];
        arr[stack1] = null;
        return tmp;
      }
    }

    public AnyType pop2() {
      if (stack2 == arr.length - 1) {
        System.out.println("Cannot pop stack!");
        return null;
      }
      else {
        stack2++;
        AnyType tmp = arr[stack2];
        arr[stack2] = null;
        return tmp;
      }
    }
  }
```

4. (a)
   - Move input (4) to S_1
   - Move input (3) to S_1
   - Move input (1) to output
   - Move input (8) to S_2
   - Move input (2) to output
   - Move S_1 (3) to output
   - Move S_1 (4) to output
   - Move input (7) to S_2
   - Move input (6) to S_2
   - Move input (9) to S_1
   - Move input (5) to output
   - Move S_2 (6) to output
   - Move S_2 (7) to output
   - Move S_2 (8) to output
   - Move S_1 (9) to output

   (b) The train `[1, 9, 8, 7, 6, 5, 4, 3, 2]` cannot be rearranged to be in increasing order.
   This is because the stacks cannot be out of order otherwise they will be unusable.
   Thus 2 will go to S_1, 3 to S_2, 4 to S_3 and 5 has no where to go.