```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab5.given;

/**
 *
 * @author vanting
 */
public class Card {

    // spades, hearts, clubs, diamonds
    public static final String[] SUITS = new String[] { "\u2660", "\u2665", "\u2663", "\u2666" };
    public static final String[] RANKS = new String[] { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K",
            "A" };
    public final String suit;
    public final String rank;

    public Card(String suit, String rank) {
        this.suit = suit;
        this.rank = rank;
    }

    @Override
    public String toString() {
        return suit + rank;
    }
}

```



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package lab5.given;

/**
 *
 * @author Van
 * @param <E>
 */
public interface Stack<E> {

    /**
     * Push an element into the stack
     *
     * @param e the new element to be pushed
     */
    void push(E e);

    /**
     * Remove the top element and return it.
     *
     * @return the top element
     */
    E pop();

    /**
     * Return the top element but leave it on the stack.
     *
     * @return the top element
     */
    E peek();

    /**
     * @return the number of elements in the stack
     */
    int size();

    /**
     *
     * @return true if the stack is empty; otherwise return false
     */
    boolean isEmpty();

}

```

```java
package lab5.solution.given;

import lab5.solution.MyStack;

/**
 *
 * @author Van Ting
 */

public class TestMyStack {

    public static void main(String[] args) {

        // create two stacks
        Stack<Integer> stack1 = new MyStack<>();
        Stack<Double> stack2 = new MyStack<>();

        // Generate random elements and store in these two stacks
        int randomInt;
        for (int i = 0; i < 20; i++) {
            randomInt = (int) (Math.random() * 100);
            stack1.push(randomInt);
            stack2.push(randomInt / 10.0);
        }

        System.out.print("The top element on Stack 1: ");
        System.out.println(stack1.peek());

        System.out.println("The size of Stack 1: " + stack1.size());

        System.out.print("Stack 1: ");
        while (!stack1.isEmpty())
            System.out.print(stack1.pop() + " ");
        System.out.println("");

        System.out.print("Stack 2: ");
        while (!stack2.isEmpty()) {
            System.out.print(stack2.pop() + " ");
        }
        System.out.println("");

        System.out.println("The size of Stack 1: " + stack1.size());
        System.out.println("The size of Stack 2: " + stack2.size());

    }
}

```



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab5.solution;

import lab5.solution.given.Card;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
// import static lab5.solution.Deal.rankCode;

/**
 *
 * @author vanting
 */
public class Deal {

    // Make a normal 52-card deck.
    static final Map<String, Integer> suitCode = new HashMap<>();
    static final Map<String, Integer> rankCode = new HashMap<>();

    static {
        suitCode.put("\u2660", 4);
        suitCode.put("\u2665", 3);
        suitCode.put("\u2663", 2);
        suitCode.put("\u2666", 1);
        rankCode.put("2", 1);
        rankCode.put("3", 2);
        rankCode.put("4", 3);
        rankCode.put("5", 4);
        rankCode.put("6", 5);
        rankCode.put("7", 6);
        rankCode.put("8", 7);
        rankCode.put("9", 8);
        rankCode.put("10", 9);
        rankCode.put("J", 10);
        rankCode.put("Q", 11);
        rankCode.put("K", 12);
        rankCode.put("A", 13);
    }

    public static void main(String[] args) {

        // 1. initialize deck
        List<Card> deck = new ArrayList<>();
        for (String suit : Card.SUITS) {
            for (String rank : Card.RANKS) {
                deck.add(new Card(suit, rank));
            }
        }
        System.out.println("Initial:\t" + deck + "\n");

        // 2. Shuffle the deck.
        Collections.shuffle(deck);
```

```
60          System.out.println("After shuffle:\t" + deck + "\n");
61
62          // 3. deal four hands
63          int numHands = 4;
64          int cardsPerHand = 13;
65          for (int i = 0; i < numHands; i++) {
66              List<Card> handView = deck.subList(cardsPerHand * i, cardsPerHand * (i +
    1));
67              List<Card> hand = new ArrayList<>(handView);
68              // handView.clear();
69
70              /*
71               * new Comparator<Card>() {
72               *
73               * @Override public int compare(Card c1, Card c2) { if
    (c1.suit.equals(c2.suit))
74               * { return rankCode.get(c1.rank) - rankCode.get(c2.rank); } else {
    return
75               * suitCode.get(c2.suit) - suitCode.get(c1.suit); }
76               *
77               * } };
78               */
79
80              Collections.sort(hand, new MyComparator());
81
82              System.out.println("Hand " + (i + 1) + ":\t" + hand);
83          }
84      }
85
86      static class MyComparator implements Comparator<Card> {
87
88          @Override
89          public int compare(Card c1, Card c2) {
90              if (c1.suit.equals(c2.suit)) {
91                  return rankCode.get(c1.rank) - rankCode.get(c2.rank);
92              } else {
93                  return suitCode.get(c2.suit) - suitCode.get(c1.suit);
94              }
95
96          }
97      }
98
99  }
100
```



```
1   /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package lab5.solution;
7
8   import java.util.TreeSet;
9   import java.util.Set;
10
11  /**
12   *
13   * @author Van
14   */
15  public class Fruits {
16
17      public static void main(String[] args) {
18
19          Set<String> setA = new TreeSet<>();
20          setA.add("apple");
21          setA.add("banana");
22          setA.add("durian");
23          setA.add("grape");
24          setA.add("papaya");
25          System.out.print("Set A: ");
26          System.out.println(setA);
27
28          Set<String> setB = new TreeSet<>();
29          setB.add("banana");
30          setB.add("mango");
31          setB.add("papaya");
32          setB.add("tomato");
33          setB.add("watermelon");
34          System.out.print("Set B: ");
35          System.out.println(setB);
36
37          System.out.print("Union: ");
38          Set<String> union = new TreeSet<>();
39          union.addAll(setA);
40          union.addAll(setB);
41          System.out.println(union);
42
43          System.out.print("Intersection: ");
44          Set<String> intersection = new TreeSet<>();
45          intersection.addAll(setA);
46          intersection.retainAll(setB);
47          System.out.println(intersection);
48
49          System.out.print("Complement of A: ");
50          Set<String> compA = new TreeSet<>(setB);
51          compA.removeAll(setA);
52          System.out.println(compA);
53
54          System.out.print("Complement of B: ");
55          Set<String> compB = new TreeSet<>(setA);
56          compB.removeAll(setB);
57          System.out.println(compB);
58
59      }
```

```
60
61 }
62
```



```java
package lab5.solution;

import java.util.ArrayList;
import java.util.List;
import lab5.solution.given.Stack;

/**
 *
 * @author Van Ting
 * @param <E> type of stack element
 */
public class MyStack<E> implements Stack<E> {

    private final List<E> list = new ArrayList<>();

    @Override
    public void push(E e) {
        list.add(e);
    }

    @Override
    public E pop() {
        if (list.size() > 0) {
            return list.remove(list.size() - 1);
        } else
            return null;
    }

    @Override
    public E peek() {
        if (list.size() > 0) {
            return list.get(list.size() - 1);
        } else
            return null;
    }

    @Override
    public int size() {
        return list.size();
    }

    @Override
    public boolean isEmpty() {
        return list.isEmpty();
    }

}

```