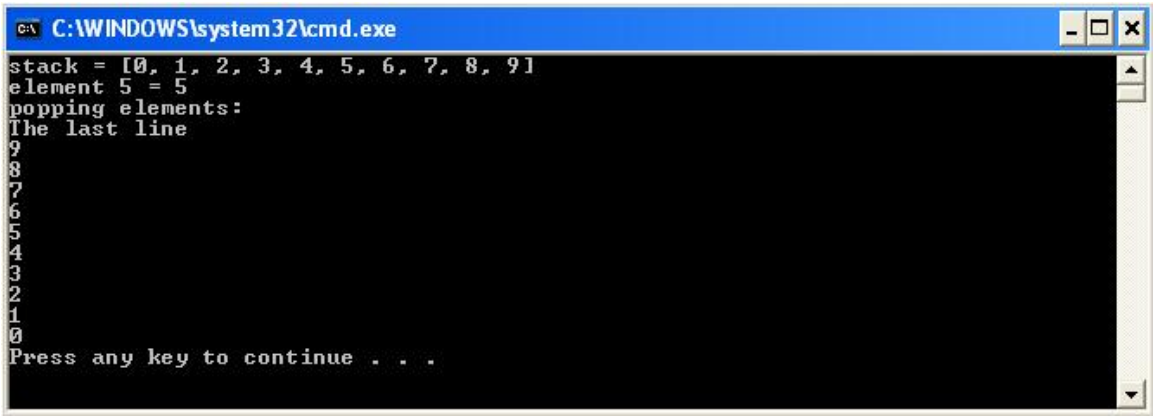
1. Demonstration of Stack class.

```
import java.util.Stack;

public class Stacks {

  public static void main(String[] args) {
    Stack stack = new Stack();
    for (int i = 0; i < 10; i++)
      stack.push(new Integer(i));
    System.out.println("stack = " + stack);
    // Treating a stack as a Vector:
    stack.addElement("The last line");
    System.out.println("element 5 = " + stack.elementAt(5));
    System.out.println("popping elements:");
    while (!stack.empty())
      System.out.println(stack.pop());
  }
}
```

The result:



2. Finding elements in a Vector.

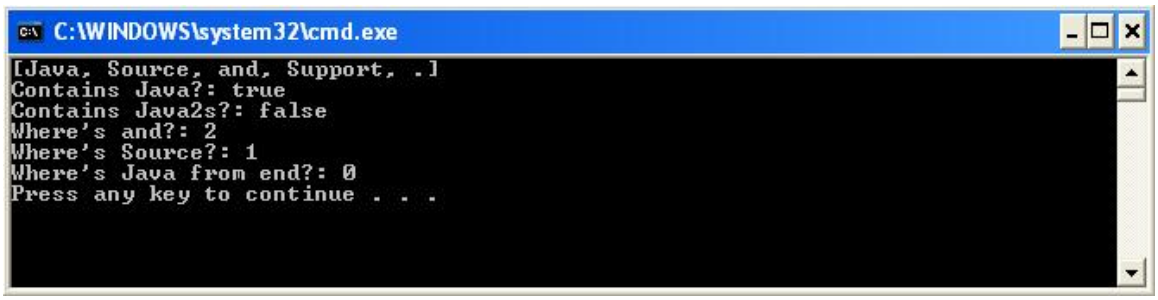
```
import java.util.Vector;

public class FindVector {

  public static void main(String args[]) {
    String data[] = { "Java", "Source", "and", "Support", "."};

    Vector v = new Vector();
    for (int i = 0, n = data.length; i < n; i++) {
      v.add(data[i]);
    }
    System.out.println(v);
    System.out.println("Contains Java?: " + v.contains("Java"));
    System.out.println("Contains Java2s?: " + v.contains("Java2s"));
    System.out.println("Where's and?: " + v.indexOf("and"));
    System.out.println("Where's Source?: " + v.indexOf("Source"));
    System.out.println("Where's Java from end?: "
        + v.lastIndexOf("Java"));
  }
}
```

The result:



3. Using the Collection interface

```
import java.awt.Color;
import java.util.*;

public class CollectionTest {
   private String colors[] = { "red", "white", "blue" };
   // create ArrayList, add objects to it and manipulate it
   public CollectionTest()
   {
      ArrayList list = new ArrayList();

      // add objects to list
      list.add( Color.magenta ); // add a color object
      for ( int count = 0; count < colors.length; count++ )
         list.add( colors[ count ] );
      list.add( Color.cyan ); // add a color object

      // output list contents
      System.out.println( "\nArrayList: " );
      for ( int count = 0; count < list.size(); count++ )
         System.out.print( list.get( count ) + " " );

      // remove all String objects
      removeStrings( list );

      // output list contents
      System.out.println( "\n\nArrayList after calling" + "
removeStrings: " );
      for ( int count = 0; count < list.size(); count++ )
      System.out.print( list.get( count ) + " " );
   }

   // remove String objects from Collection
   public void removeStrings( Collection collection )
   {
      // get iterator
      Iterator iterator = collection.iterator();
      // loop while collection has items
      while ( iterator.hasNext() )
         if ( iterator.next() instanceof String )
      iterator.remove(); // remove String object
   }

   // execute application
   public static void main( String args[] )
   {
      new CollectionTest();
   }
} // end class CollectionTest
```

The result:

```
C:\WINDOWS\system32\cmd.exe
ArrayList:
java.awt.Color[r=255,g=0,b=255] red white blue java.awt.Color[r=0,g=255,b=255]

ArrayList after calling removeStrings:
java.awt.Color[r=255,g=0,b=255] java.awt.Color[r=0,g=255,b=255] Press any key to
 continue . . .
```

4. Working with key-value pairs in a Hashtable

```
import java.util.Enumeration;
import java.util.Hashtable;

public class PlanetDiameters {
  public static void main(String args[]) {
    String names[] = { "Mercury", "Venus", "Earth", "Mars", "Jupiter",
        "Saturn", "Uranus", "Neptune", "Pluto" };
    float diameters[] = { 4800f, 12103.6f, 12756.3f, 6794f, 142984f,
        120536f, 51118f, 49532f, 2274f };
    Hashtable hash = new Hashtable();
    for (int i = 0, n = names.length; i < n; i++) {
      hash.put(names[i], new Float(diameters[i]));
    }
    Enumeration e = hash.keys();
    Object obj;
    while (e.hasMoreElements()) {
      obj = e.nextElement();
      System.out.println(obj + ": " + hash.get(obj));
    }
  }
}
```

The result:

```
C:\WINDOWS\system32\cmd.exe
ArrayList:
java.awt.Color[r=255,g=0,b=255] red white blue java.awt.Color[r=0,g=255,b=255]

ArrayList after calling removeStrings:
java.awt.Color[r=0,g=255,b=255] java.awt.Color[r=0,g=255,b=255] Press any key to
 continue . . .
```

Do It Yourself

4.1. Do workshop of the module 5

4.2. Redo the exercise 3.2 but the student objects would be stored in a collection.

4.3. Write an application to count amount of each word in a text file. Use the class Hashtable to store words and their amount.

4.4. Write a simple dictionary program. Each word in the dictionary is corresponding to a line in data file. Each line includes two fields: word and its definition.

The program has the menu:

```
Menu
-------------------------
    1. Save new word
    2. List word by alphabet
    3. Exit
    Your choice: _
```

+ Save new word: add a word into the data file
+ List word by alphabet: input a character, load all words that start with that character into a SortedSet, and display them.

References

+ Java tutorials

+ Javadoc

+ Java2s.com

+ Javapassion.com

+ Java almanac
http://www.exampledepot.com