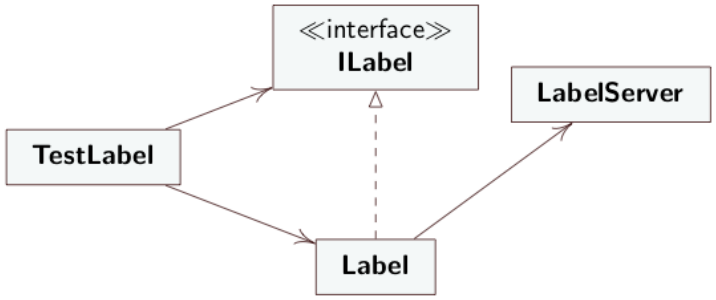
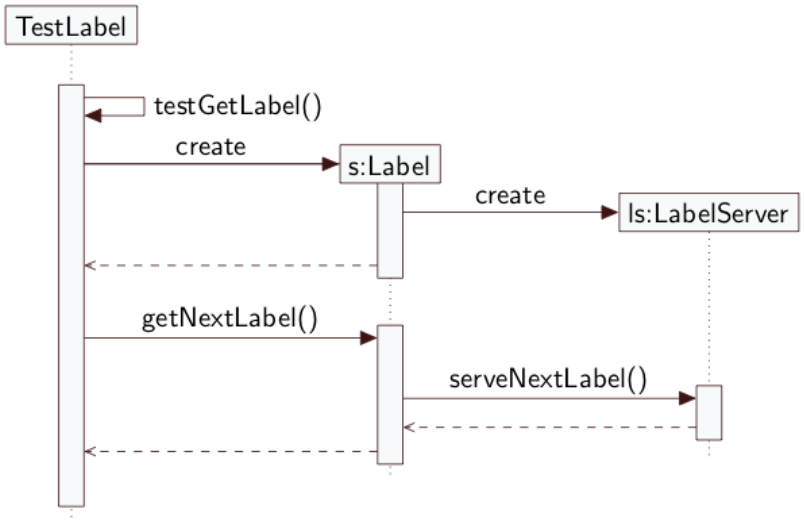


Design Pattern Adapter



Design Pattern Adapter

```
// Codice Java che implementa il design pattern Adapter


/** ILabel e' un Target */
public interface ILabel {
   public String getNextLabel();
   public boolean checkUsed(int k);
   public void insertTag(String s);
}


/** LabelServer e' un Adaptee */
public class LabelServer {
   private int num = 1;
   private String prefix;

   public LabelServer(String p) {
      prefix = p;
   }

   public String serveNextLabel() {
      return prefix + num++;
   }

   public int getCount() {
```

```
        return num;
    }

    public void change(String s) {
        prefix = s;
    }
}


import java.util.Arrays;
import java.util.List;

/** Label e' un Adapter */
public class Label implements ILabel {
    private List< String > l = Arrays.asList("cat", "dog", "sheep");
    private LabelServer ls;
    private String p;

    // si istanzia subito un Adaptee
    public Label(String prefix) {
        p = prefix;
        ls = new LabelServer(p);
    }

    // l'adattamento consiste nel chiamare un metodo con nome diverso sull'Adapte
    @Override
    public String getNextLabel() {
        return ls.serveNextLabel();
    }

    // l'adattamento consiste nel fornire una funzionalita' diversa rispetto a qu
    // metodo sull'Adaptee, che implementa solo parzialmente quanto richiesto dal
    @Override
    public boolean checkUsed(int k) {
        return (ls.getCount() >= k);
    }

    // qui, oltre a chiamare il corrispondente metodo dell'Adaptee, si verifica,
    // la prima condizione sul corpo del metodo, che la precondizione sia soddisf
    // ovvero non cambiare l'etichetta se non si usa un valore fra quelli permess
    @Override
    public void insertTag(String t) {
        if (l.contains(t)) ls.change(t);
    }
}



public class TestLabel {
    static public void main(String args[]) {
        testGetLabel();
        testChangeLabel();
```

```
        testNoChangeLabel();
        testUsed();
    }

    private static void testGetLabel() {
        ILabel s = new Label("LAB");
        if (s.getNextLabel().equals("LAB1")) System.out.println("OK Test get label
        else System.out.println("FAILED Test get label");
    }

    private static void testChangeLabel() {
        ILabel s = new Label("LAB");
        s.insertTag("cat");
        if (s.getNextLabel().equals("cat1")) System.out.println("OK Test change la
        else System.out.println("FAILED Test change label");
    }

    private static void testNoChangeLabel() {
        ILabel s = new Label("LAB");
        s.insertTag("zebra");
        if (s.getNextLabel().equals("LAB1")) System.out.println("OK Test no-change
        else System.out.println("FAILED Test no-change label");
    }

    private static void testUsed() {
        ILabel s = new Label("LAB");
        if (s.checkUsed(1) && !s.checkUsed(2)) System.out.println("OK Test used 1"
        else System.out.println("FAILED Test used 1");
        s.getNextLabel();
        if (s.checkUsed(2) && !s.checkUsed(3)) System.out.println("OK Test used 2"
        else System.out.println("FAILED Test used 2");
    }
}
```