```java
 *  Licensed to the Apache Software Foundation (ASF) under one or more
package com.java;

import static org.testng.Assert.*;

import static java.lang.System.*;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;

import org.testng.annotations.Test;

/**
 * <b>Templates</b> are a structured description of <b>coding patterns</b>.
 *
 * @author  Bert Lee
 * @version 2013-6-2
 */
public class Templates {

    /**
     * <b>arrayadd</b> Template.
     *
     * <pre>
     * 参考{@link Arrays#copyOf(Object[], int)}实现.
     * </pre>
     *
     * @param a
     * @param e
     */
    public void arrayadd(Object[] a, Object e) {
        Object[] result = new Object[a.length + 1];
        // array copy - faster
        System.arraycopy(a, 0, result, 0, a.length);
        result[a.length] = e;
    }

    private String name;

    /**
     * Generate <b>hashCode()</b> and <b>equals(Object)</b>.
     *
     * <pre>
     * 参考{@link Arrays#hashCode(Object[])}实现.
     * </pre>
     */
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }

    /**
```

```java
     * <b>equals</b>, <b>instanceof</b>, <b>cast</b> Template.
     *
     * <pre>
     * 参考《Effective Java》实现, p-36
     * </pre>
     */
    @Override
    public boolean equals(Object obj) {
        // 性能优化：比较操作有可能很昂贵
        if (this == obj)
            return true;

        if (! (obj instanceof Templates))
            return false;

        // cast
        Templates other = (Templates) obj;
        return name == null ? other.name == null : name.equals(other.name);
    }

    /*
     * Generate
     */
//  @Override
//  public boolean equals(Object obj) {
//      // 性能优化：比较操作有可能很昂贵
//      if (this == obj)
//          return true;
//
//      if (obj == null)
//          return false;
//      if (getClass() != obj.getClass())
//          return false;
//
//      Templates other = (Templates) obj;
//      return name == null ? other.name == null : name.equals(other.name);
//  }

    /**
     * <b>try</b> or <b>catch</b> Template.
     *
     * @param s
     * @param defaultValue
     * @return
     */
    public static int parseInt(String s, int defaultValue) {
        try {
            int i = Integer.parseInt(s, 10);
            return i;
        } catch (NumberFormatException nfe) {
            return defaultValue;
        }
    }

    private String email;
    private String phone;

    /**
     * <b>toString</b> Template.
     */
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Templates [")
          .append("name=").append(name)
          .append(", email=").append(email)
          .append(", phone=").append(phone)
          .append(']');
        return sb.toString();
    }

    /*
     * Generate
     */
//  @Override
//  public String toString() {
//      StringBuilder sb = new StringBuilder();
//      sb.append("Templates [")
//        .append("name=").append(name)
//        .append(", email=").append(email)
//        .append(", phone=").append(phone)
//        .append(']');
//      return sb.toString();
//  }

    /**
     * <b>foreach</b> Template.
     *
     * @param a
     * @return
     */
    public static int sum(int[] a) {
        int sum = 0;
        for (int i : a) {
            sum += i;
        }
        return sum;
    }

    /**
     * <b>static_method</b>, <b>while</b>, <b>finally</b>, <b>if</b> Template.
     *
     * @param fileName
     * @return
     * @throws IOException
     */
    public static String bufferedRead(String fileName) throws IOException {
        BufferedReader in = null;

        try {
            in = new BufferedReader(new FileReader(fileName));

            StringBuilder content = new StringBuilder(1024);  // 2KB
            String l = null;
            while ((l = in.readLine()) != null) {
                content.append(l);
            }
            return content.toString();
        } finally {
            if (in != null) {
                in.close();
            }
        }
    }

    /**
     * <b>main</b>, <b>sysout</b> Template.
     *
     * @param args
     */
    public static void main(String[] args) {
        out.println("Hello world!");
    }

    /**
     * <b>method</b>, <b>new</b>, <b>list</b>, <b>for-iterate</b> Template.
     */
    public void loop() {
        List<String> list = new ArrayList<String>();
        list.add("1");
        list.add("2");
        list.add("3");

        for (Iterator<String> it = list.iterator(); it.hasNext();) {
            String s = it.next();
            if ("2".equals(s)) {
                it.remove(); // 只有需要删除操作时，才会使用迭代器
            }
        }
    }

    /**
     * <b>static_final</b> Template.
     */
    public static final long INVALID_ID = -1;

    /**
     * <b>switch</b> Template.
     *
     * @param s
     */
    public void select(Selector s) {
        switch (s) { // Ctrl + 1
        case BLUE:
            // do what
            break;
        case GREEN:
            //
            break;
        case RED:
            //
            break;
        default:
            //
            break;
        }
    }
    static enum Selector {
        RED("red"),
        GREEN("green"),
        BLUE("blue"),
        ;

        @Override
        public String toString() {
            return value;
        }

        private String value;

        Selector(String value) {
            this.value = value;
        }
    }

    /**
     * <b>test</b>, <b>array</b>, <b>for</b> Template.
     */
    @Test
    public void unitTest() {
        List<Integer> expected = new ArrayList<Integer>(3);
        for (int i = 0; i < expected.size(); i++) {
            expected.add(Integer.valueOf(i));
        }

        Integer[] actual = toArray(expected);

        assertEquals(actual.length, expected.size());
        // for
        for (int i = 0; i < actual.length; i++) {
            assertEquals(actual[i], expected.get(i));
        }
    }

    /**
     * <b>toarray</b> Template.
     *
     * @param c
     * @return
     */
    public static Integer[] toArray(Collection<Integer> c) {
        return c.toArray(new Integer[c.size()]);
    }
}

```