* Chương 1
* Viết chương trình
* Mùa thu năm 2011

Tổng quan về lớp học

* Thông tin khóa học
  + Trên trang web và Bảng đen
  + [www.uncp.edu/home/lilliec/](https://translate.google.com/translate?hl=vi&prev=_t&sl=auto&tl=vi&u=http://www.uncp.edu/home/lilliec/)
  + giáo trình
  + Bài tập
  + Bài tập về nhà
  + Bài kiểm tra
  + Chính sách tham dự
* Sách giáo khoa
  + Tsui & Karam, Yếu *tố cần thiết của Kỹ thuật phần mềm*

Mục tiêu

* Phân tích các vấn đề cho các chương trình đơn giản
  + Yêu cầu
  + Thiết kế ràng buộc
  + Kiểm tra
  + Dự toán lỗi
  + Chi tiết thực hiện
* Hiểu chuỗi hoạt động
* Xem trước các chủ đề trong tương lai

Yêu cầu

* Yêu cầu - xác định và đủ điều kiện hệ thống
  + Được xác định bởi khách hàng, với sự giúp đỡ từ kỹ sư
  + Chức năng - xác định những gì phải được thực hiện
  + Không chức năng - đủ điều kiện
* Hạn chế thiết kế
  + Về thiết kế hoặc thực hiện
  + Ngôn ngữ lập trình, nền tảng, v.v.
* Yêu cầu - Sức mạnh và Tình yêu
  + Hàng hóa của chúng tôi
  + Năng lực - năng lực
  + Không có tài năng
* Văn phòng
  + Hạ gục
  + Hiện tại

Một vấn đề đơn giản

* Đưa ra một tập hợp các dòng văn bản (chuỗi) được lưu trữ trong một tệp, sắp xếp chúng theo thứ tự bảng chữ cái và ghi chúng vào một tệp khác
* Đây là yêu cầu

Yêu cầu chức năng

* định dạng đầu vào
  + Kích thước nhân vật
  + Đường phân cách
* Chỉ định sắp xếp
  + Số
  + Chữ hoa / chữ thường
* Trường hợp đặc biệt
* Ranh giới
* Điều kiện lỗi
* Kho dữ liệu
* Phần cứng dòng
* Chỉ
* Vì thế
* Trẻ em
* Trường tổng hợp
* Ranh giới
* Tôi ăn

Những yêu cầu vô lý

* Hiệu suất
* Thời gian thực ?
* Sửa đổi
* GIÁM SÁT
* Thời gian thực?
* Năng lực thay đổi

Thiết kế ràng buộc

* Giao diện người dùng
  + GUI, CLI, Web
* Đầu vào và kích thước điển hình
* Nền tảng
* Lịch trình

Quyết định thiết kế

* Ngôn ngữ lập trình
* Thuật toán

Kiểm tra

* White-Box - kiểm tra mã như đã viết
* Hộp đen - giả sử không có kiến ​​thức về mã
* Kiểm thử đơn vị - bởi lập trình viên, trên mỗi mảnh
* Kiểm thử tích hợp - Đặt các đơn vị lại với nhau thành hệ thống lớn hơn
* Kiểm tra chấp nhận - nếu thất bại, khách hàng từ chối chương trình

Dự toán

* Cần bao nhiêu nỗ lực?
  + Thường được thực hiện trong tháng
* Giá cả
  + Một khi biết nỗ lực có thể ước tính chi phí
* Thời gian / lên lịch
  + Một khi biết nỗ lực có thể ước tính lịch trình

Quy tắc thực hiện

* Hãy kiên định
* Chọn tên cẩn thận
* Kiểm tra trước khi sử dụng
  + Kiểm tra, kiểm tra, kiểm tra
* Biết thư viện của bạn
* Làm mã đánh giá

Thiết kế cơ bản

* Lớp StringSorter
  + Đọc
  + Sắp xếp
  + Viết
  + Wrapper để làm Đọc rồi Sắp xếp rồi Viết
* Sẽ kiểm tra đơn vị từng phương pháp
* Sẽ sử dụng ArrayList để giữ các dòng

Triển khai thực hiện

* nhập java.io. \*; // cho Reader (s), Writer (s), IOException
* nhập java.util. \*; // cho Danh sách, ArrayList, Iterator
* lớp công khai StringSorter {
* Dòng ArrayList;
* công khai void readFromStream (Reader r) ném IOException
* {
* BufferedReader br = new BufferedReader (r);
* dòng = new ArrayList ();
* trong khi (đúng) {
* Chuỗi đầu vào = br.readLine ();
* if (đầu vào == null)
* phá vỡ;
* dòng.add (đầu vào);
* }
* }

Kiểm tra

* lớp công khai TestStringSorter mở rộng TestCase {
* riêng ArrayList make123 () {
* ArrayList l = new ArrayList ();
* l.add ("một");
* l.add ("hai");
* l.add ("ba");
* trả lại l;
* }
* công khai void testReadFromStream () ném IOException {
* Trình đọc trong = Trình tạo tệp mới ("in.txt");
* StringSorter ss = new StringSorter ();
* ArrayList l = make123 ();
* ss.readFromStream (trong);
* assertEquals (l, ss.lines);
* }
* Hình 1.5: GUI Junit

Triển khai thực hiện

* hoán đổi khoảng trống tĩnh (Danh sách l, int i1, int i2) {
* Đối tượng tmp = l.get (i1);
* l.set (i1, l.get (i2));
* l.set (i2, tmp);
* }

Kiểm tra

* công khai void testSwap () {
* ArrayList l1 = make123 ();
* ArrayList l2 = new ArrayList ();
* l2.add ("một");
* l2.add ("ba");
* l2.add ("hai");
* StringSorter.swap (l1,1,2);
* assertEquals (l1, l2);
* }

Triển khai thực hiện

* static int findIdxBiggest (Danh sách l, int từ, int đến) {
* Chuỗi lớn nhất = (Chuỗi) l.get (0);
* int idxBigest = from;
* for (int i = from + 1; i <= to; ++ i) {
* if (most.compareTo (l.get (i)) <0) {// nó lớn hơn
* lớn nhất = (Chuỗi) l.get (i);
* idxBigest = i;
* }
* }
* trả lại idxBigest;
* }
* Hình 1.8: phương thức findIdxBigest

Kiểm tra

* công khai void testFindIdxBiggest () {
* StringSorter ss = new StringSorter ();
* ArrayList l = make123 ();
* int i = StringSorter.findIdxBiggest (l, 0, l.size () - 1);
* assertEquals (i, 1);
* }
* Hình 1.9: phương thức testFindIdxBigest

Triển khai thực hiện

* công khai void sort () {
* for (int i = lines.size () - 1; i> 0; --i) {
* int big = findIdxBiggest (dòng, 0, i); hoán đổi (dòng, i, lớn);
* }
* }
* Hình 1.10: phương thức sắp xếp

Kiểm tra

* công khai void testSort1 () {
* StringSorter ss = new StringSorter ();
* ss.lines = make123 ();
* ArrayList l2 = new ArrayList ();
* l2.add ("một");
* l2.add ("ba");
* l2.add ("hai");
* ss.sort ();
* assertEquals (l2, ss.lines);
* }
* Hình 1.11 phương thức testSort1

Biết thư viện của bạn

* void sort () {
* java.util.Collections.sort (dòng);
* }
* Một thói quen sắp xếp đã tồn tại trong java (và hầu hết các ngôn ngữ khác)

Triển khai thực hiện

* công khai void writeToStream (Nhà văn w) ném IOException {
* PrintWriter pw = new PrintWriter (w);
* Trình lặp i = lines.iterator ();
* while (i.hasNext ()) {
* pw.println ((Chuỗi) (i.next ()));
* }
* }
* Hình 1.13: phương thức writeToStream

Kiểm tra

* công khai void testWriteToStream () ném IOException {
* // viết ra một giá trị đã biết
* StringSorter ss1 = new StringSorter ();
* ss1.lines = make123 ();
* Writer out = new FileWriter ("test.out");
* ss1.writeToStream (ra);
* out.c Đóng (); // sau đó đọc nó và so sánh
* Trình đọc trong = Trình tạo tệp mới ("in.txt");
* StringSorter ss2 = new StringSorter ();
* ss2.readFromStream (trong);
* assertEquals (ss1.lines, ss2.lines);
* }

Triển khai thực hiện

* sắp xếp khoảng trống công khai (Chuỗi inputFileName,
* Chuỗi outputFileName)
* ném IOException
* {
* Trình đọc trong = Trình tạo tệp mới (inputFileName);
* Writer out = new FileWriter (outputFileName);
* StringSorter ss = new StringSorter ();
* ss.readFromStream (trong);
* ss.sort ();
* ss.writeToStream (ra);
* ghim theo();
* out.c Đóng ();
* }

Kiểm tra

* công khai void testSort2 () ném IOException {
* // viết ra một giá trị đã biết
* StringSorter ss1 = new StringSorter ();
* ss1.sort ("in.txt", "test2.out");
* ArrayList l = new ArrayList ();
* l.add ("một");
* l.add ("ba");
* l.add ("hai");
* // sau đó đọc nó và so sánh
* Trình đọc trong = Trình tạo tệp mới ("test2.out");
* StringSorter ss2 = new StringSorter ();
* ss2.readFromStream (trong);
* assertEquals (l, ss2.lines);
* }
* Hình 1.16: phương thức testSort2

Giao diện dòng lệnh

* nhập java.io.IOException;
* lớp công khai StringSorterCommandLine {
* public static void main (String args []) ném IOException {
* if (args.length! = 2) {
* System.out.println ("Sử dụng: cmd inputfile outputfile");
* } khác {
* StringSorter ss = new StringSorter ();
* ss.sort (args [0], args [1]);
* }
* }
* }

GUI xấu

* lớp công khai StringSorterBadGUI {
* public static void main (String args []) ném IOException {
* thử {
* StringSorter ss = new StringSorter ();
* Chuỗi inFileName = JOptionPane.showInputDialog
* ("Vui lòng nhập tên tệp đầu vào");
* Chuỗi outFileName = JOptionPane.showInputDialog
* ("Vui lòng nhập tên tệp đầu ra");
* ss.sort (inFileName, outFileName);
* } cuối cùng {
* System.exit (1);
* }
* }

Giao diện tốt hơn

GUI tốt hơn

* Nhấp vào nút bất kỳ, để có được hộp thoại mở