import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.Scanner;

public class measurableDemo {

public static void main(String[] args) {

ArrayList<measurableString> words = new ArrayList<measurableString>();

String[] Ia = new String[5];

Employee[] Fa = new Employee[5];

Scanner in = new Scanner(System.in);

initialization(words, Ia, Fa);

binarySearcher<String> bSIa = new binarySearcher<String>(Ia);

binarySearcher<Employee> bSFa = new binarySearcher<Employee>(Fa);

boolean done = false;

do {

int choice = menu(in);

switch (choice) {

case 1:

System.out.println("-Testing largest-");

System.out.println("Largest word:" + measurable.largest(words));

System.out.println("Expected: little");

break;

case 2:

System.out.println("-Show sorted list of word-");

for (String x : Ia) {

System.out.println(x);

}

break;

case 3:

System.out.println("-Search list of word-");

System.out.print("Enter word: ");

String word = in.nextLine();

int pos1 = bSIa.search(word);

if (pos1 != -1) {

System.out.println(word + " existed at position: " + pos1);

} else {

System.out.println(word + " does not exist.");

}

break;

case 4:

System.out.println("-Show sorted list of employee-");

System.out.printf("%-10s %-10s %-3s\n", "First Name", "Last

Name", "ID");

for (Employee x : Fa) {

System.out.println(x);

}

break;

case 5:

System.out.println("-Search list of employee-");

System.out.print("Enter the first name of employee: ");

String name = in.nextLine();

int pos2 = bSFa.search(new Employee(name, "", 0));

if (pos2 != -1) {

System.out.println(name + " existed at position: " + pos2);

} else {

System.out.println(name + " does not exist.");

}

break;

case 6:

System.out.println("-Exiting-");

done = true;

default:

}

} while (!done);

}

public static int menu(Scanner in) {

int choice = 0;

boolean done = false;

System.out.println("1. Test largest");

System.out.println("2. Show sorted list of word");

System.out.println("3. Search list of word");

System.out.println("4. Show sorted employee list");

System.out.println("5. Search employee list by first name");

System.out.println("6. Exit");

do {

try {

choice = in.nextInt();

in.nextLine();

if (choice < 1 || choice > 6) {

throw new InputMismatchException();

}

done = true;

} catch (InputMismatchException ime) {

System.out.print("Invalid Input. Enter again: ");

in.nextLine();

}

} while (!done);

return choice;

}

public static void initialization(ArrayList<measurableString> words, String[] Ia, Employee[] Fa) {

words.add(new measurableString("Mary"));

words.add(new measurableString("had"));

words.add(new measurableString("a"));

words.add(new measurableString("little"));

words.add(new measurableString("lamb"));

Ia[0] = "Apple";

Ia[1] = "Orange";

Ia[2] = "Banna";

Ia[3] = "Grape";

Ia[4] = "Kiwi";

Fa[0] = new Employee("Bradon", "Tylor", 532);

Fa[1] = new Employee("Temmo", "Gar", 10);

Fa[2] = new Employee("Annie", "Tolaro", 753);

Fa[3] = new Employee("Yoey", "Malinda", 190);

Fa[4] = new Employee("Mitron", "Jake", 56);

insertionSort.sort(Ia, 5);

insertionSort.sort(Fa, 5);

}

}

import java.util.ArrayList;

public abstract class measurable {

public abstract double getMeasure();

public static <T extends measurable> T largest(ArrayList<T> obj) {

T v = obj.get(0);

for (T x : obj) {

if (v.getMeasure() < x.getMeasure()) {

v = x;

}

}

return v;

}

}

public class measurableString extends measurable {

private final String value;

public measurableString(String s) {

value = s;

}

public double getMeasure() {

return (double) value.length();

}

public String toString() {

return value;

}

}

public class Employee implements Comparable<Employee> {

private final String fName;

private final String lName;

private final Integer id;

public Employee(String f, String l, int i) {

fName = f;

lName = l;

id = i;

}

public String getFirstName() {

return fName;

}

public String getLastName() {

return lName;

}

public int getID() {

return id;

}

public int compareTo(Employee o) {

return fName.compareTo(o.getFirstName());

}

public String toString() {

return String.*format*("%-10s %-10s %-10d", fName, lName, id);

}

}

public class binarySearcher<T extends Comparable<T>> {

private T[] array;

public binarySearcher(T[] a) {

array = a;

}

public int search(T value) {

int low = 0;

int high = array.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int diff = array[mid].compareTo(value);

if (diff == 0) {

return mid;

} else if (diff < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

}

public class insertionSort {

public static <T extends Comparable<T>> void sort(T[] array, int size) {

T temp;

for (int i = 1; i < array.length; i++) {

for (int j = i; j > 0; j--) {

if (((Comparable<T>) array[j]).compareTo((T) array[j - 1]) < 0) {

temp = array[j];

array[j] = array[j - 1];

array[j - 1] = temp;

}

}

}

}

}

