Christopher AboShear

COSC 3319 01 Data Structures and Algorithms

MWF 9:00 – 9:50am

Successfully Completed Option “A”

**Input\_b.txt**

IRC FORD 4

IRC FORD 2

IRC GMC 2

IRC RAM 2

IRC CHEVY 3

PS

PF

DC FORD

PS

PF

ILP BOEING 3 6

ILP PIPER 2 1

ILP CESSNA 4 4

PF

**Input\_c.txt**

IR 33

IR 57

IL 85

IL 62

IR 95

PR

PL

D 57

D 33

D 33

D 62

IR 22

D 95

PR

**Output:**

**Output for c:**

option=1

line=IR 33

val=33

Inserting: 33

line=IR 57

val=57

Inserting: 57

Insert right

line=IL 85

val=85

Inserting: 85

Insert Left

line=IL 62

val=62

Inserting: 62

Insert Left

line=IR 95

val=95

Inserting: 95

Insert right

line=PR

Right to Left

value: 95

value: 57

value: 33

value: 85

value: 62

line=PL

Left to Right

value: 62

value: 85

value: 33

value: 57

value: 95

line=D 57

val=57

Trying to find: 57

Value Deleted

line=D 33

val=33

Trying to find: 33

Value Deleted

line=D 33

val=33

Trying to find: 33

Error: Value not Found

line=D 62

val=62

Trying to find: 62

Value Deleted

line=IR 22

val=22

Inserting: 22

Insert right

line=D 95

val=95

Trying to find: 95

Value Deleted

line=PR

Right to Left

value: 22

value: 85

**output b**

option=2

line=IRC FORD 4

val=FORD 4

Inserting: num\_doors=4 manufacturer=FORD

line=IRC FORD 2

val=FORD 2

Inserting: num\_doors=2 manufacturer=FORD

Insert right

line=IRC GMC 2

val=GMC 2

Inserting: num\_doors=2 manufacturer=GMC

Insert right

line=IRC RAM 2

val=RAM 2

Inserting: num\_doors=2 manufacturer=RAM

Insert right

line=IRC CHEVY 3

val=CHEVY 3

Inserting: num\_doors=3 manufacturer=CHEVY

Insert right

line=PS

list\_size=5

line=PF

Left to Right

value: num\_doors=4 manufacturer=FORD

value: num\_doors=2 manufacturer=FORD

value: num\_doors=2 manufacturer=GMC

value: num\_doors=2 manufacturer=RAM

value: num\_doors=3 manufacturer=CHEVY

line=DC FORD

Trying to find: num\_doors=0 manufacturer=FORD

Value Deleted

val=FORD

line=PS

list\_size=4

line=PF

Left to Right

value: num\_doors=2 manufacturer=FORD

value: num\_doors=2 manufacturer=GMC

value: num\_doors=2 manufacturer=RAM

value: num\_doors=3 manufacturer=CHEVY

line=ILP BOEING 3 6

val=BOEING 3 6

Inserting: manufacturer=BOEING num\_doors=3 num\_engines=6

Insert right

line=ILP PIPER 2 1

val=PIPER 2 1

Inserting: manufacturer=PIPER num\_doors=2 num\_engines=1

Insert right

line=ILP CESSNA 4 4

val=CESSNA 4 4

Inserting: manufacturer=CESSNA num\_doors=4 num\_engines=4

Insert right

line=PF

Left to Right

value: num\_doors=2 manufacturer=FORD

value: num\_doors=2 manufacturer=GMC

value: num\_doors=2 manufacturer=RAM

value: num\_doors=3 manufacturer=CHEVY

value: manufacturer=BOEING num\_doors=3 num\_engines=6

value: manufacturer=PIPER num\_doors=2 num\_engines=1

value: manufacturer=CESSNA num\_doors=4 num\_engines=4

code:

**Base.java**

**package** containers;

/\*Abstract class for Car.java, Plane.java,

and Int\_Container.java\*/

**public** **abstract** **class** Base {

// overload for comparison

**abstract** **boolean** equal(Base b);

// to convert to strings

**abstract** String to\_string();

}

**Inheritance\_List.java**

**package** containers;

**public** **class** Inheritance\_List {

**private** **class** Node{

Node left,right;

Base val;

//initialize variables in Node class

Node(Base v,Node l, Node r){

val = v;

left = l;

right = r;

}

Node(Base v){

val = v;

}

}

**private** Node head;

//Constructor

**public** Inheritance\_List() {

head = **new** Node(**new** Int\_Container(0),head,head);

}

//Inserting

**public** **void** insert(**boolean** left, Base val) {

//temp node

Node temp = **new** Node(val);

System.***out***.println("Inserting: "+val.to\_string());

//Checking where to insert

**if**(((Int\_Container)head.val).get\_value()==0) {

temp.right = head;

temp.left = head;

head.right = temp;

head.left = temp;

}**else** {

//Inserting to the Left

**if**(left== **true**) {

System.***out***.println("Insert Left");

temp.left = head;

temp.right = head.right;

head.right.left = temp;

head.right = temp;

}**else** {

//Inserting to the right

System.***out***.println("Insert right");

temp.right = head;

temp.left = head.left;

head.left.right = temp;

head.left = temp;

}

}//Increment

((Int\_Container)head.val).set\_value(((Int\_Container)head.val).get\_value() + 1);

}

//Deletion

**public** **void** delete(Base val) {

Node temp = head.right;

System.***out***.println("Trying to find: "+val.to\_string());

//Finding where to delete

**for**(**int** i=0;i<((Int\_Container)head.val).get\_value();i++) {

**if**(temp.val.equal(val)) {

**break**;

}

temp = temp.right;

}

//Condition for when node not found

**if**(!temp.val.equal(val)) {

System.***out***.println("Error: Value not Found");

**return**;

}**else** {

//Delete when value is found

System.***out***.println("Value Deleted");

temp.left.right = temp.right;

temp.right.left = temp.left;

//Decrement

((Int\_Container)head.val).set\_value(((Int\_Container)head.val).get\_value() - 1);

}

}

//Returns new size of list

**public** **int** list\_size() {

**return** ((Int\_Container)head.val).get\_value();

}

//Printing the list from right to left

**public** **void** print(**boolean** right\_to\_left) {

**if**(right\_to\_left == **true**) {

System.***out***.println("Right to Left");

Node temp = head.left;

**for**(**int** i=0;i<((Int\_Container)head.val).get\_value();i++) {

System.***out***.println("value: "+temp.val.to\_string());

temp = temp.left;

}

}**else** {

//Printing list from left to right

System.***out***.println("Left to Right");

Node temp = head.right;

**for**(**int** i=0;i<((Int\_Container)head.val).get\_value();i++) {

System.***out***.println("value: "+temp.val.to\_string());

temp = temp.right;

}

}

}

}

**Driver.java**

package containers;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Driver {

public static void main(String[] args) throws FileNotFoundException {

Scanner sc = new Scanner(System.in);

System.out.println("Please enter input file: "); //Get file name

String input\_file\_name = sc.nextLine();

System.out.println("Enter 1 for (int) or 2 for (else): "); // Enter 1 for integer ,2 for anything else

String temp = sc.nextLine();

int option = Integer.parseInt(temp);

System.out.println("input\_file\_name="+input\_file\_name);

System.out.println("option="+option);

Inheritance\_List il = new Inheritance\_List();

File f = new File("C:\\"+input\_file\_name);

Scanner fs = new Scanner(f);

// reads file

while(fs.hasNextLine()) {

String line = fs.nextLine();

System.out.println("line="+line);

// option 1 is list of integers

if(option==1) {

// insertion

if(line.charAt(0)=='I') {

int v = Integer.parseInt(line.substring(3));

// value to read

Int\_Container val = new Int\_Container(v);

System.out.println("val="+val.to\_string());

// R for Rear

if(line.charAt(1)=='R') {

il.insert(false, val);

// F for front

}else {

il.insert(true, val);

}

// delete

}else if(line.charAt(0)=='D') {

int v = Integer.parseInt(line.substring(2));

Int\_Container val = new Int\_Container(v);

System.out.println("val="+val.to\_string());

il.delete(val);

// print list

}else {

// print from rear to front

if(line.charAt(1)=='R') {

il.print(true);

// print front to rear

}else {

il.print(false);

}

}

}else {

// insert

if(line.charAt(0)=='I') {

String val = line.substring(4);

System.out.println("val="+val);

String[] a = val.split(" ");

// rear

if(line.charAt(1)=='R') {

// car

if(line.charAt(2)=='C') {

Car v = new Car(a[0],Integer.parseInt(a[1]));

il.insert(false, v);

// plane

}else {

Plane p = new Plane(a[0],Integer.parseInt(a[1]),Integer.parseInt(a[2]));

il.insert(false, p);

}

// front

}else {

// car

if(line.charAt(2)=='C') {

Car v = new Car(a[0],Integer.parseInt(a[1]));

il.insert(true, v);

// plane

}else {

Plane p = new Plane(a[0],Integer.parseInt(a[1]),Integer.parseInt(a[2]));

il.insert(false, p);

}

}

// delete

}else if(line.charAt(0)=='D') {

String val = line.substring(3);

if(line.charAt(1)=='C') {

Car c = new Car(val);

il.delete(c);

}else {

Plane p = new Plane(val);

il.delete(p);

}

System.out.println("val="+val);

// print

}else {

if(line.charAt(1)=='R') {

il.print(true);

}else if(line.charAt(1)=='F'){

il.print(false);

}else {

System.out.println("list\_size="+il.list\_size());

}

}

}

}

fs.close();

sc.close();

}

}

**Int\_Containers**

**package** containers;

**public** **class** Int\_Container **extends** Base{

**private** **int** value;

**public** Int\_Container(**int** v) {

value = v;

}

**public** **int** get\_value() {

**return** value;

}

**public** **void** set\_value(**int** v) {

value = v;

}

@Override

// allow for comparison

**boolean** equal(Base b) {

Int\_Container i = (Int\_Container)b;

**if**(i.value == value) {

**return** **true**;

}**else** {

**return** **false**;

}

}

@Override

// convert to string

String to\_string() {

**return** "" + value;

}

}

**Car.java**

**package** containers;

**public** **class** Car **extends** Base{

**private** String manufacturer; //

**private** **int** num\_doors;

//initialization of manufacture

Car(String m){

manufacturer = m;

}

//initialization of manufacture and num\_doors

Car(String m, **int** d){

manufacturer = m;

num\_doors = d;

}

//Comparisonss

@Override

**boolean** equal(Base b) {

Car c = (Car)b;

**if**(c.manufacturer.equals(manufacturer) || (c.manufacturer.equals(manufacturer) && c.num\_doors==num\_doors)) {

**return** **true**;

}**else** {

**return** **false**;

}

}

//Override return string

@Override

String to\_string() {

**return** "num\_doors=" + num\_doors+" manufacturer="+manufacturer;

}

}

**Plane.java**

**package** containers;

// creating sub class

**public** **class** Plane **extends** Base{

**private** String manufacturer;

**private** **int** num\_doors;

**private** **int** num\_engines;

Plane(String m){

manufacturer = m;

}

//Initialize manufacture, num\_doors, and num\_engines

Plane(String m, **int** d, **int** e){

manufacturer = m;

num\_doors = d;

num\_engines = e;

}

//Compare

@Override

**boolean** equal(Base b) {

Plane p = (Plane)b;

**if**(p.manufacturer.equals(manufacturer)

||(p.manufacturer.equals(manufacturer) &&

p.num\_doors == num\_doors &&

p.num\_engines == num\_engines)) {

**return** **true**;

}**else** {

**return** **false**;

}

}

//Override return String

@Override

String to\_string() {

**return** "manufacturer="+manufacturer+" num\_doors="+num\_doors+" num\_engines="+num\_engines;

}

}