**// ArrayList.java**

import java.util.Scanner;

import arraylist.DynamicArray;

import arraylist.MyArrayList;

import arraylist.MyLinkedList;

public class ArrayList {

public static void PrintMenu(){

System.out.println("1. 在列表最后位置添加数据（输入格式：1 数据）");

System.out.println("2. 在第i个位置上添加数据（输入格式：2 位置 数据）");

System.out.println("3. 获取第i个位置上的数据（输入格式：3 位置）");

System.out.println("4. 删除第i个位置上的数据（输入格式：4 位置）");

System.out.println("5. 修改第i个位置上的数据（输入格式：5 位置 数据）");

System.out.println("0. 退出程序");

}

public static void main(String[] args) {

MyArrayList myList = new MyArrayList();

MyLinkedList myLinkList = new MyLinkedList();

Scanner reader = new Scanner(System.in);

System.out.print("请输入初始化列表的大小：");

int initialSize;

while (true){

while (!reader.hasNextInt()){

System.out.println("请输入整数！");

System.out.print("请输入初始化列表的大小：");

reader.next();

}

initialSize = reader.nextInt();

if (initialSize < 0){

System.out.println("初始化大小不能小于0！");

System.out.print("请输入初始化列表的大小：");

continue;

}

break;

}

myList = new MyArrayList(initialSize);

control controller = new control();

PrintMenu();

while (reader.hasNext()){

if (!reader.hasNextInt()){

System.out.println("请输入正确的模式序号！");

reader.next();

continue;

}

int mode = reader.nextInt();

if (mode == 1){

Object x = reader.next();

controller.Mode\_1(myList, x);

controller.Mode\_1(myLinkList, x);

}

else if (mode == 2){

int x = reader.nextInt();

Object y = reader.next();

controller.Mode\_2(myList, x, y);

controller.Mode\_2(myLinkList, x, y);

}

else if (mode == 3){

int x = reader.nextInt();

controller.Mode\_3(myList, x);

controller.Mode\_3(myLinkList, x);

}

else if (mode == 4){

int x = reader.nextInt();

controller.Mode\_4(myList, x);

controller.Mode\_4(myLinkList, x);

}

else if (mode == 5){

int x = reader.nextInt();

Object y = reader.next();

controller.Mode\_5(myList, x, y);

controller.Mode\_5(myLinkList, x, y);

}

else if (mode == 0){

break;

}

else{

System.out.println("请输入正确的模式序号！");

continue;

}

}

reader.close();

}

}

class control {

public void Mode\_1(DynamicArray myList, Object x){

myList.add(x);

myList.printList();

}

public void Mode\_2(DynamicArray myList, int x, Object y){

myList.add(x, y);

myList.printList();

}

public void Mode\_3(DynamicArray myList, int x){

System.out.println(myList.get(x));

System.out.println("");

}

public void Mode\_4(DynamicArray myList, int x){

myList.remove(x);

myList.printList();

}

public void Mode\_5(DynamicArray myList, int x, Object y){

myList.set(x, y);

myList.printList();

}

}

**// DynamicArray.java**

package arraylist;

public interface DynamicArray {

void add(Object element);

void add(int index, Object element);

Object get(int index);

void remove(int index);

void set(int index, Object element);

int size();

void printList();

}

**// MyArrayList.java**

package arraylist;

public class MyArrayList implements DynamicArray {

private Object[] data;

private int size;

public MyArrayList() {

data = new Object[10];

size = 0;

}

public MyArrayList(int initialCapacity) {

data = new Object[initialCapacity];

size = 0;

}

public void add(Object element) {

if (size == data.length) {

ensureCapacity();

}

data[size] = element;

size++;

}

public void add(int index, Object element) {

if (index < 0 || index > size) {

System.out.println("无效位置/超出范围");

return;

}

if (size == data.length) {

ensureCapacity();

}

for (int i = size; i > index; i--) {

data[i] = data[i - 1];

}

data[index] = element;

size++;

}

public Object get(int index) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return null;

}

return data[index];

}

public void remove(int index) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return;

}

for (int i = index; i < size - 1; i++) {

data[i] = data[i + 1];

}

data[size - 1] = null;

size--;

}

public void set(int index, Object element) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return;

}

data[index] = element;

}

public int size() {

return size;

}

public void printList(){

System.out.print("List: ");

for (int i = 0; i < size; i++){

System.out.print(data[i] + " ");

}

System.out.println("");

}

private void ensureCapacity() {

int newCapacity = data.length \* 2;

Object[] newData = new Object[newCapacity];

System.arraycopy(data, 0, newData, 0, size);

data = newData;

}

}

**// MyLinkedList.java**

package arraylist;

public class MyLinkedList implements DynamicArray {

private Node head;

private Node tail;

private int size;

public MyLinkedList() {

head = null;

tail = null;

size = 0;

}

private class Node {

Object data;

Node next;

Node(Object data) {

this.data = data;

this.next = null;

}

}

public void add(Object element) {

Node newNode = new Node(element);

if (size == 0) {

head = newNode;

tail = newNode;

}

else {

tail.next = newNode;

tail = newNode;

}

size++;

}

public void add(int index, Object element) {

if (index < 0 || index > size) {

System.out.println("无效位置/超出范围");

return;

}

Node newNode = new Node(element);

if (index == 0) {

newNode.next = head;

head = newNode;

}

else {

Node current = head;

for (int i = 0; i < index - 1; i++) {

current = current.next;

}

newNode.next = current.next;

current.next = newNode;

}

size++;

}

public Object get(int index) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return null;

}

Node current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

return current.data;

}

public void remove(int index) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return;

}

if (index == 0) {

head = head.next;

} else {

Node current = head;

for (int i = 0; i < index - 1; i++) {

current = current.next;

}

current.next = current.next.next;

}

size--;

}

public void set(int index, Object element) {

if (index < 0 || index >= size) {

System.out.println("无效位置/超出范围");

return;

}

Node current = head;

for (int i = 0; i < index; i++) {

current = current.next;

}

current.data = element;

}

public int size() {

return size;

}

public void printList() {

Node current = head;

System.out.print("List: ");

while (current != null) {

System.out.print(current.data + " ");

current = current.next;

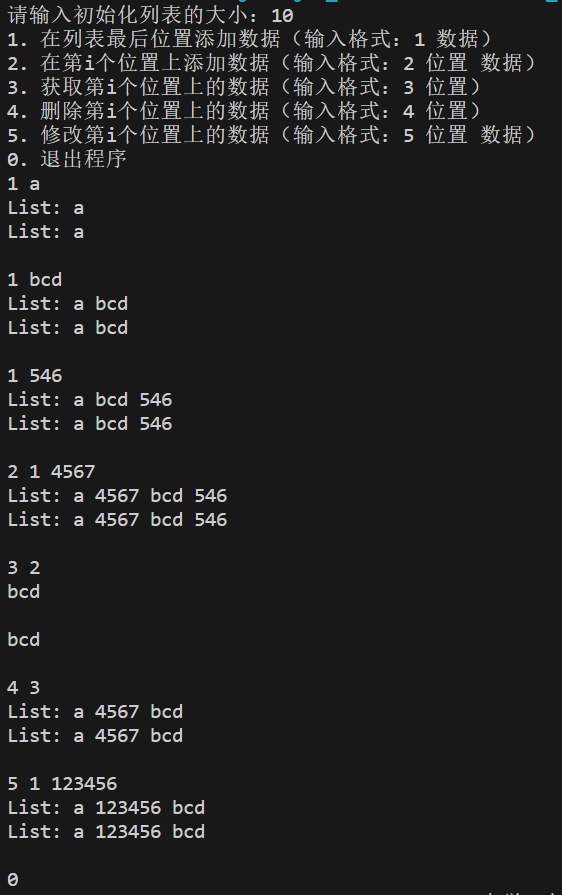
}

System.out.println();

System.out.println();

}

}



// MyArrayList 方法操作

// MyLinkedList 方法操作

**MyArrayList:**

优点：

1. 随机访问元素效率高，时间复杂度低。
2. 使用连续的内存存储，节省内存空间。

缺点：

1. 在中间插入或删除元素时，需要移动元素，效率较低，平均时间复杂度较高。
2. 当数组容量不足时，需要重新分配内存和复制元素，可能引发性能开销。

**MyLinkedList:**

优点：

1. 在中间插入或删除元素时，效率较高，平均时间复杂度为O(1)。
2. 不需要预先分配大块连续内存，更加灵活。

缺点：

1. 随机访问元素效率较低，平均时间复杂度为O(n)。
2. 每个元素都需要额外的链表节点，可能占用更多内存。