Java 核心技术

第十章 Java数据结构

第三节 列表List

华东师范大学 陈良育

List(1)

- List：列表
  - 有序的Collection
  - 允许重复元素
  - {1，2，4，{5，2}，1，3}
- List 主要实现
  - ArrayList(非同步的)
  - LinkedList(非同步)
  - Vector(同步)

List(2)

- ArrayList：
  - 以数组实现的列表，不支持同步
    - List list = Collections.synchronizedList(new ArrayList(...));
  - 利用索引位置可以快速定位访问
  - 不适合指定位置的插入、删除操作
  - 适合变动不大，主要用于查询的数据
  - 和Java数组相比，其容量是可动态调整的
  - ArrayList在元素填满容器时会自动扩充容器大小的50%
  - 查看程序ArrayListTest.java

List(3)

- LinkedList:
  - 以双向链表实现的列表，不支持同步
    - List list = Collections.synchronizedList(new LinkedList(...));
  - 可被当作堆栈、队列和双端队列进行操作
  - 顺序访问高效，随机访问较差，中间插入和删除高效
  - 适用于经常变化的数据
  - 查看程序LinkedListTest.java

List(4)

- Vector(同步)
  - 和ArrayList类似，可变数组实现的列表
  - Vector同步，适合在多线程下使用
  - 原先不属于JCF框架，属于Java最早的数据结构，性能较差
  - 从JDK1.2开始，Vector被重写，并纳入到JCF
  - 官方文档建议在非同步情况下，优先采用ArrayList
  - 查看程序VectorTest.java

List(5)

- 总结
  - ArrayList/LinkedList/Vector
  - 同步采用Vector
  - 非同步情况下，根据数据操作特点选取ArrayList/LinkedList

代码(1) ArrayListTest.java


```
public class ArrayListTest {
    public static void main(String[] a) {
        ArrayList<Integer> al = new ArrayList<Integer>();
        al.add(3);
        al.add(2);
        al.add(1);
        al.add(4);
        al.add(5);
        al.add(6);
        al.add(new Integer(6));

        System.out.print("The third element is  ");
        System.out.println(al.get(3));
        al.remove(3);   //删除第四个元素，后面元素往前挪动
        al.add(3, 9);   //将9插入到第4个元素，后面元素往后挪动
```

代码(2) ArrayListTest.java


```
        System.out.println("=======遍历方法==============");

        ArrayList<Integer> as = new ArrayList<Integer>(100000);
        for (int i=0; i<100000; i++)
        {
            as.add(i);
        }
        traverseByIterator(as);
        traverseByIndex(as);
        traverseByFor(as);
    }
    public static void traverseByIterator(ArrayList<Integer> al)
    {
        long startTime = System.nanoTime();
        System.out.println("==============迭代器遍历==============");
        Iterator<Integer> iter1 = al.iterator();
        while(iter1.hasNext()){
            iter1.next();
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
```

代码(3) ArrayListTest.java


```
    public static void traverseByIndex(ArrayList<Integer> al)
    {
        long startTime = System.nanoTime();
        System.out.println("=============随机索引值遍历==============");
        for(int i=0;i<al.size();i++)
        {
            al.get(i);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
    public static void traverseByFor(ArrayList<Integer> al)
    {
        long startTime = System.nanoTime();
        System.out.println("=============for循环遍历==============");
        for(Integer item : al)
        {
            ;
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

代码(4) LinkedListTest.java


```
public class LinkedListTest {

    public static void main(String[] args) {
        LinkedList<Integer> ll = new LinkedList<Integer>();
        ll.add(3);
        ll.add(2);
        ll.add(5);
        ll.add(6);
        ll.add(6);
        System.out.println(ll.size());
        ll.addFirst(9);   //在头部增加9
        ll.add(3, 10);    //将10插入到第四个元素，四以及后续的元素往后挪动
        ll.remove(3);     //将第四个元素删除
```

代码(5) LinkedListTest.java


```
        LinkedList<Integer> list = new LinkedList<Integer>();
        for (int i=0; i<100000; i++)
        {
            list.add(i);
        }
        traverseByIterator(list);
        traverseByIndex(list);
        traverseByFor(list);

    }

    public static void traverseByIterator(LinkedList<Integer> list)
    {
        long startTime = System.nanoTime();
        System.out.println("==============迭代器遍历==============");
        Iterator<Integer> iter1 = list.iterator();
        while(iter1.hasNext()){
            iter1.next();
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
```

代码(6) LinkedListTest.java


```
    public static void traverseByIndex(LinkedList<Integer> list)
    {
        long startTime = System.nanoTime();
        System.out.println("==============随机索引值遍历==============");
        for(int i=0;i<list.size();i++)
        {
            list.get(i);
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
    public static void traverseByFor(LinkedList<Integer> list)
    {
        long startTime = System.nanoTime();
        System.out.println("==============for循环遍历==============");
        for(Integer item : list)
        {
            ;
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

代码(7) ListCompareTest.java


```
public class ListCompareTest {

    public static void main(String[] args) {
        int times = 10 * 1000;
        // times = 100 * 1000;
        // times = 1000 * 1000;

        ArrayList<Integer> arrayList = new ArrayList<Integer>();
        LinkedList<Integer> linkedList = new LinkedList<Integer>();

        System.out.println("Test times = " + times);
        System.out.println("------------------------");
```

代码(8) ListCompareTest.java


```
// ArrayList add
long startTime = System.nanoTime();

for (int i = 0; i < times; i++) {
    arrayList.add(0,i);
}
long endTime = System.nanoTime();
long duration = endTime - startTime;
System.out.println(duration + " <--ArrayList add");

// LinkedList add
startTime = System.nanoTime();

for (int i = 0; i < times; i++) {
    linkedList.add(0,i);
}
endTime = System.nanoTime();
duration = endTime - startTime;
System.out.println(duration + " <--LinkedList add");
System.out.println("----------------------------");
```

代码(9) ListCompareTest.java


```
// ArrayList get
startTime = System.nanoTime();

for (int i = 0; i < times; i++) {
    arrayList.get(i);
}
endTime = System.nanoTime();
duration = endTime - startTime;
System.out.println(duration + " <--ArrayList get");

// LinkedList get
startTime = System.nanoTime();

for (int i = 0; i < times; i++) {
    linkedList.get(i);
}
endTime = System.nanoTime();
duration = endTime - startTime;
System.out.println(duration + " <--LinkedList get");
System.out.println("----------------------------");
```

代码(10) ListCompareTest.java


```
        // ArrayList remove
        startTime = System.nanoTime();

        for (int i = 0; i < times; i++) {
            arrayList.remove(0);
        }
        endTime = System.nanoTime();
        duration = endTime - startTime;
        System.out.println(duration + " <--ArrayList remove");

        // LinkedList remove
        startTime = System.nanoTime();

        for (int i = 0; i < times; i++) {
            linkedList.remove(0);
        }
        endTime = System.nanoTime();
        duration = endTime - startTime;
        System.out.println(duration + " <--LinkedList remove");
    }
}
```

代码(11) VectorTest.java


```
public class VectorTest {

    public static void main(String[] args) {
        Vector<Integer> v = new Vector<Integer>();
        v.add(1);
        v.add(2);
        v.add(3);
        v.remove(2);
        v.add(1, 5);
        System.out.println(v.size());

        System.out.println("=======遍历方法==============");

        Vector<Integer> v2 = new Vector<Integer>(100000);
        for (int i = 0; i < 100000; i++) {
            v2.add(i);
        }
        traverseByIterator(v2);
        traverseByIndex(v2);
        traverseByFor(v2);
        traverseByEnumeration(v2);
    }
```

代码(12) VectorTest.java


```
public static void traverseByIterator(Vector<Integer> v) {
    long startTime = System.nanoTime();
    System.out.println("==============迭代器遍历==============");
    Iterator<Integer> iter1 = v.iterator();
    while (iter1.hasNext()) {
        iter1.next();
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}

public static void traverseByIndex(Vector<Integer> v) {
    long startTime = System.nanoTime();
    System.out.println("==============随机索引值遍历==============");
    for (int i = 0; i < v.size(); i++) {
        v.get(i);
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println(duration + "纳秒");
}
```

代码(13) VectorTest.java


```
    public static void traverseByFor(Vector<Integer> v) {
        long startTime = System.nanoTime();
        System.out.println("=============for循环遍历==============");
        for (Integer item : v) {
            ;
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }

    public static void traverseByEnumeration(Vector<Integer> v) {
        long startTime = System.nanoTime();
        System.out.println("=============Enumeration遍历==============");
        for (Enumeration<Integer> enu = v.elements(); enu.hasMoreElements();) {
            enu.nextElement();
        }
        long endTime = System.nanoTime();
        long duration = endTime - startTime;
        System.out.println(duration + "纳秒");
    }
}
```

谢 谢！