JAVA 프로그래밍 숙달

작 성 일 : 20.02.17
작 성 자 : 도 원 진

1. HashMap , ArrayList, Vector 속도 비교

- Collection 클래스의 아이템 갯수
- **속도비교(벤치마킹)**

```
package com.bizsprint.benchmark;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Vector;


//결론, 삭제할 때 ArrayList와 Vector가 시간이 많이 걸린다.
public class compareHashMap_ArrayList_Vector {
    static Integer[] testArray = new Integer[5000000];
    Integer[] values = new Integer[4990];
    HashMap hashmap;
    ArrayList arrayList;
    Vector vecotr;
    public void addTest() {
        long start = System.nanoTime();
        hashmap = new HashMap();
        for(Integer integer : testArray){
            hashmap.put(integer, integer);
        }
        long end = System.nanoTime();
        System.out.println("입력");
        System.out.println("\tHashMap Test\t" + second(start, end) + "초");

        start = System.nanoTime();

        arrayList = new ArrayList();
        for(Integer integer : testArray){
            arrayList.add(integer);
        }

        end = System.nanoTime();

        System.out.println("\tArrayList\t" + second(start, end) + "초");

        start = System.nanoTime();

        vecotr = new Vector();
        for(Integer integer : testArray){
```

```
            vecotr.add(integer);
        }

        end = System.nanoTime();


        System.out.println("\tVecotr\t\t" + second(start, end) + "초");
    }

    public void getTest() {

        long start,end;

        start = System.nanoTime();
        for(Integer value : values){
            hashmap.get(value);
        }
        end = System.nanoTime();
        System.out.println("탐색 소요 시간");
        System.out.println("\thashmap\t\t" + second(start, end) + "초");

        start = System.nanoTime();
        for(Integer value : values){
            arrayList.get(value);
        }
        end = System.nanoTime();
        System.out.println("\tArrayList\t" + second(start, end) + "초");

        start = System.nanoTime();
        for(Integer value : values){
            vecotr.get(value);
        }
        end = System.nanoTime();
        System.out.println("\tVector\t\t" + second(start, end) + "초");
    }

    public void removeTest() {
        long start,end;
        start = System.nanoTime();
        for(Integer value : values){
            hashmap.remove(value);
        }
        end = System.nanoTime();
        System.out.println("삭제 소요시간");
        System.out.println("\tHashMap\t\t" + second(start, end) + "초");

        start = System.nanoTime();
        for(Integer value : values){
            arrayList.remove(value);
        }
        end = System.nanoTime();
        System.out.println("\tArrayList\t" + second(start, end) + "초");

        start = System.nanoTime();
        for(Integer value : values){
            vecotr.remove(value);
        }
        end = System.nanoTime();
```

```
        System.out.println("\tVector\t\t" + second(start, end) + "초");
    }

    private void prepare() {
        long start = System.nanoTime();
        for (int i = 0; i < testArray.length; i++) {
            testArray[i] = i;
        }
        long end = System.nanoTime();

        ArrayList temp = new ArrayList(1000);
        for(int i = 0 ; i < 4990 ; i++){
            temp.add(i * 1000);
        }
        temp.toArray(values);

        System.out.println(testArray.length + "개의 인스턴스 생성 시간 " +
                second(start, end) + "초");

    }

    private double second(long start, long end){
        return (end - start) / Math.pow(10, 9);
    }

    public void start() {
        prepare();
        addTest();
        getTest();
        removeTest();
    }

    public static void main(String[] args) {
        compareHashMap_ArrayList_Vector test = new
compareHashMap_ArrayList_Vector();
        test.start();
    }
}
```

결과

5000000개의 인스턴스 생성 시간 0.1023375초

--	HashMap	ArrayList	Vector
입력	2.6045415초	0.0989808초	0.0933573초
탐색/ 조회	**0.0015809초**	5.525E-4초	6.251E-4초
삭제	**0.0015886초**	**20.8032698초**	**21.0731901초**

정리

- 조회 , 삭제 할 경우 HashMap 이용, List 인터페이스 계열은 쓰지 말것
- 입력에서는 큰 차이를 보이지 않는다.

2. String, StringBuffer, StringBuilder 차이점과 속도 비교

2-1. 특징정리

	String	StringBuffer	StringBuilder
Mutable / IMutable	IM	M	M
연산시 내부적으로 char[ ] 이용	O	O	
조회연산(멀티 스레드 환경)	good		
수정/변경	bad	good	good
멀티쓰레드환경, 동기화		thread-safe	Bad
싱글쓰레드환경			good

2-2. 속도 비교 코드

```
public class StringPerformance {
    public static double mkLongStr(int len)
    {
        long start = System.nanoTime();

        String str = "";
        for(int i=0 ; i < len ; i++)    str += "a";

        long end = System.nanoTime();
        return (end-start)/(double)1_000_000_000;
    }

    public static double mkLongStrBuffer(int len)
    {
        long start = System.nanoTime();

        StringBuffer sb = new StringBuffer();
        for(int i=0 ; i < len ; i++)    sb.append("a");

        long end = System.nanoTime();
        return (end-start)/(double)1_000_000_000;
    }

    public static double mkLongStrBuilder(int len)
    {
        long start = System.nanoTime();

        StringBuilder sb = new StringBuilder();
        for(int i=0 ; i < len ; i++)    sb.append("a");

        long end = System.nanoTime();
        return (end-start)/(double)1_000_000_000;
    }
```

```
    public static void main(String[] args) {

        int[] lenList = new int[]{10_000, 20_000, 40_000, 80_000, 160_000,
320_000, 640_000,
                                  1_280_000, 2_560_000, 5_120_000,
10_240_000, 20_480_000, 40_960_000, 81_920_000, 163_840_000};

        for(int len : lenList)
        {
            System.out.println("String\t\t" + len + "\t" + mkLongStr(len));
            System.out.println("StringBuffer\t" + len + "\t" +
mkLongStrBuffer(len));
            System.out.println("StringBuilder\t" + len + "\t" +
mkLongStrBuilder(len));
        }
    }
}
```

연산횟수	String	StringBuffer	StringBuilder
10_000	0.041485	3.931E-4	1.886E-4
20_000	0.124186	6.21E-4	3.656E-4
40_000	0.4309068	0.001143	7.48E-4
80_000	1.1586242	0.0019471	8.073E-4
160_000	4.5415226	0.001448	0.0010108
320_000	19.2830811	0.0016736	0.001506
640_000	78.0562982	0.0032089	0.0029626