Java 核心技术(进阶)

第五章 Java多线程和并发编程

第七节 Java并发数据结构

华东师范大学 陈良育

并发数据结构(1)

- 常用的数据结构是线程不安全的
  - ArrayList, HashMap, HashSet 非同步的
  - 多个线程同时读写，可能会抛出异常或数据错误
- 传统Vector，Hashtable等同步集合性能过差
- 并发数据结构：数据添加和删除
  - 阻塞式集合：当集合为空或者满时，等待
  - 非阻塞式集合：当集合为空或者满时，不等待，返回null或异常

并发数据结构(2)

- List
  - Vector 同步安全，写多读少
  - ArrayList 不安全
  - Collections.synchronizedList(List list) 基于synchronized，效率差
  - CopyOnWriteArrayList 读多写少，基于复制机制，非阻塞
- Set
  - HashSet 不安全
  - Collections.synchronizedSet(Set set) 基于synchronized，效率差
  - CopyOnWriteArraySet (基于CopyOnWriteArrayList实现) 读多写少，非阻塞

并发数据结构(3)

- Map
  - Hashtable 同步安全，写多读少
  - HashMap 不安全
  - Collections.synchronizedMap(Map map) 基于synchronized，效率差
  - ConcurrentHashMap 读多写少，非阻塞
- Queue & Deque (队列，JDK 1.5 提出)
  - ConcurrentLinkedQueue 非阻塞
  - ArrayBlockingQueue/LinkedBlockingQueue 阻塞

总结

- 了解数据结构并发读写的问题
- 根据业务特点，使用正确的并发数据结构

代码(1) ListTest.java


```
public class ListTest {

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

        //线程不安全
        List<String> unsafeList = new ArrayList<String>();
        //线程安全
        List<String> safeList1 = Collections.synchronizedList(new ArrayList<String>());
        //线程安全
        CopyOnWriteArrayList<String> safeList2 = new CopyOnWriteArrayList<String>();

        ListThread t1 = new ListThread(unsafeList);
        ListThread t2 = new ListThread(safeList1);
        ListThread t3 = new ListThread(safeList2);
```

代码(2) ListTest.java


```
for(int i = 0; i < 10; i++){
    Thread t = new Thread(t1, String.valueOf(i));
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t2, String.valueOf(i));
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t3, String.valueOf(i));
    t.start();
}

//等待子线程执行完
Thread.sleep(2000);

System.out.println("listThread1.list.size() = " + t1.list.size());
System.out.println("listThread2.list.size() = " + t2.list.size());
System.out.println("listThread3.list.size() = " + t3.list.size());
```

代码(3) ListTest.java


```
//输出list中的值
System.out.println("unsafeList: ");
for(String s : t1.list){
     if(s == null){
          System.out.print("null   ");
     }
     else
     {
          System.out.print(s + "   ");
     }
}
System.out.println();
System.out.println("safeList1: ");
for(String s : t2.list){
     if(s == null){
          System.out.print("null   ");
     }
     else
     {
          System.out.print(s + "   ");
     }
}
```

代码(4) ListTest.java


```
        System.out.println();
        System.out.println("safeList2: ");
        for(String s : t3.list){
            if(s == null){
                System.out.print("null  ");
            }
            else
            {
                System.out.print(s + "  ");
            }
        }
    }
}
```

代码(5) ListThread.java


```
class ListThread implements Runnable{
    public List<String> list;

    public ListThread(List<String> list){
        this.list = list;
    }

    @Override
    public void run() {
        int i = 0;
        while(i<10)
        {
            try {
                Thread.sleep(10);
            }catch (InterruptedException e){
                e.printStackTrace();
            }
            //把当前线程名称加入list中
            list.add(Thread.currentThread().getName());
            i++;
        }
    }
}
```

代码(6) SetTest.java


```
public class SetTest{

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

        //线程不安全
        Set<String> unsafeSet = new HashSet<String>();
        //线程安全
        Set<String> safeSet1 = Collections.synchronizedSet(new HashSet<String>());
        //线程安全
        CopyOnWriteArraySet<String> safeSet2 = new CopyOnWriteArraySet<String>();

        SetThread t1 = new SetThread(unsafeSet);
        SetThread t2 = new SetThread(safeSet1);
        SetThread t3 = new SetThread(safeSet2);
```

代码(7) SetTest.java


```
//unsafeSet的运行测试
for(int i = 0; i < 10; i++){
    Thread t = new Thread(t1, String.valueOf(i));
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t2, String.valueOf(i));
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t3, String.valueOf(i));
    t.start();
}

//等待子线程执行完
Thread.sleep(2000);

System.out.println("setThread1.set.size() = " + t1.set.size());
System.out.println("setThread2.set.size() = " + t2.set.size());
System.out.println("setThread3.set.size() = " + t3.set.size());
```

代码(8) SetTest.java


```
//输出set中的值
System.out.println("unsafeSet: ");
for(String element:t1.set){
     if(element == null){
          System.out.print("null   ");
     }
     else
     {
          System.out.print(element + "   ");
     }
}
System.out.println();
System.out.println("safeSet1: ");
for(String element:t2.set){
     if(element == null){
          System.out.print("null   ");
     }
     else
     {
          System.out.print(element + "   ");
     }
}
```

代码(9) SetTest.java


```
            System.out.println();
            System.out.println("safeSet2: ");
            for(String element:t3.set){
                if(element == null){
                    System.out.print("null  ");
                }
                else
                {
                    System.out.print(element + "  ");
                }
            }
        }
}
```

代码(10) SetThread.java


```
class SetThread implements Runnable{
    public Set<String> set;

    public SetThread(Set<String> set){
        this.set = set;
    }

    @Override
    public void run() {
        int i = 0;
        while(i<10)
        {
            i++;
            try {
                Thread.sleep(10);
            }catch (InterruptedException e){
                e.printStackTrace();
            }
            //把当前线程名称加入list中
            set.add(Thread.currentThread().getName() + i);
        }
    }
}
```

代码(11) MapTest.java


```
public class MapTest{

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

        //线程不安全
        Map<Integer,String> unsafeMap = new HashMap<Integer,String>();
        //线程安全
        Map<Integer,String> safeMap1 = Collections.synchronizedMap(new HashMap<Integer,String>());
        //线程安全
        ConcurrentHashMap<Integer,String> safeMap2 = new ConcurrentHashMap<Integer,String>();

        MapThread t1 = new MapThread(unsafeMap);
        MapThread t2 = new MapThread(safeMap1);
        MapThread t3 = new MapThread(safeMap2);
```

代码(12) MapTest.java


```
//unsafeMap的运行测试
for(int i = 0; i < 10; i++){
    Thread t = new Thread(t1);
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t2);
    t.start();
}
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(t3);
    t.start();
}

//等待子线程执行完
Thread.sleep(2000);

System.out.println("mapThread1.map.size() = " + t1.map.size());
System.out.println("mapThread2.map.size() = " + t2.map.size());
System.out.println("mapThread3.map.size() = " + t3.map.size());
```

代码(13) MapTest.java


```
//输出set中的值
System.out.println("unsafeMap: ");
Iterator iter = t1.map.entrySet().iterator();
while(iter.hasNext()) {
    Map.Entry<Integer,String> entry = (Map.Entry<Integer,String>)iter.next();
    // 获取key
    System.out.print(entry.getKey() + ":");
    // 获取value
    System.out.print(entry.getValue() + " ");
}
System.out.println();

System.out.println("safeMap1: ");
iter = t2.map.entrySet().iterator();
while(iter.hasNext()) {
    Map.Entry<Integer,String> entry = (Map.Entry<Integer,String>)iter.next();
    // 获取key
    System.out.print(entry.getKey() + ":");
    // 获取value
    System.out.print(entry.getValue() + " ");
}
```

代码(14) MapTest.java


```
        System.out.println();
        System.out.println("safeMap2: ");
        iter = t3.map.entrySet().iterator();
        while(iter.hasNext()) {
            Map.Entry<Integer,String> entry = (Map.Entry<Integer,String>)iter.next();
            // 获取key
            System.out.print(entry.getKey() + ":");
            // 获取value
            System.out.print(entry.getValue() + " ");
        }
        System.out.println();
        System.out.println("mapThread1.map.size() = " + t1.map.size());
        System.out.println("mapThread2.map.size() = " + t2.map.size());
        System.out.println("mapThread3.map.size() = " + t3.map.size());
    }
}
```

代码(15) MapThread.java


```
class MapThread implements Runnable
{
    public Map<Integer,String> map;

    public MapThread(Map<Integer,String> map){
        this.map = map;
    }

    @Override
    public void run() {
        int i=0;

        while(i<100)
        {
            //把当前线程名称加入map中
            map.put(i++,Thread.currentThread().getName());
            try {
                Thread.sleep(10);
            }catch (InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}
```

代码(16) QueueTest.java


```
public class QueueTest {

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

        //线程不安全
        Deque<String> unsafeQueue = new ArrayDeque<String>();
        //线程安全
        ConcurrentLinkedDeque<String> safeQueue1 = new ConcurrentLinkedDeque<String>();

        ArrayBlockingQueue<String> safeQueue2 = new ArrayBlockingQueue<String>(100);

        QueueThread t1 = new QueueThread(unsafeQueue);
        QueueThread t2 = new QueueThread(safeQueue1);
        QueueThread t3 = new QueueThread(safeQueue2);
```

代码(17) QueueTest.java


```
for(int i = 0; i < 10; i++){
    Thread thread1 = new Thread(t1, String.valueOf(i));
    thread1.start();
}
for(int i = 0; i < 10; i++) {
    Thread thread2 = new Thread(t2, String.valueOf(i));
    thread2.start();
}
for(int i = 0; i < 10; i++) {
    Thread thread3 = new Thread(t3, String.valueOf(i));
    thread3.start();
}

//等待子线程执行完
Thread.sleep(2000);

System.out.println("queueThread1.queue.size() = " + t1.queue.size());
System.out.println("queueThread2.queue.size() = " + t2.queue.size());
System.out.println("queueThread3.queue.size() = " + t3.queue.size());
```

代码(18) QueueTest.java


```
        //输出queue中的值
        System.out.println("unsafeQueue: ");
        for(String s:t1.queue)
        {
            System.out.print(s + " ");
        }
        System.out.println();
        System.out.println("safeQueue1: ");
        for(String s:t2.queue)
        {
            System.out.print(s + " ");
        }
        System.out.println();
        System.out.println("safeQueue2: ");
        for(String s:t3.queue)
        {
            System.out.print(s + " ");
        }
    }
}
```

代码(19) QueueThread.java


```
class QueueThread implements Runnable{
    public Queue<String> queue;

    public QueueThread(Queue<String> queue){
        this.queue = queue;
    }

    @Override
    public void run() {
        int i = 0;
        while(i<10)
        {
            i++;
            try {
                Thread.sleep(10);
            }catch (InterruptedException e){
                e.printStackTrace();
            }
            //把当前线程名称加入list中
            queue.add(Thread.currentThread().getName());
        }
    }
}
```

谢 谢！