实现 java8 分组时使用 BigDecimal

作者：os-tll
原文链接：https://ld246.com/article/1587523029008
来源网站：链滴

```java
package com.yss.util;

import java.math.BigDecimal;
import java.util.Collections;
import java.util.EnumSet;
import java.util.Set;
import java.util.function.BiConsumer;
import java.util.function.BinaryOperator;
import java.util.function.Function;
import java.util.function.Supplier;
import java.util.stream.Collector;


/**
 * 重写java- java-stream-util-Collectors 实现BigDecimal的summing
 *
 * @author tanglonglong
 * @version 1.0
 * @date 2020/4/22 9:45
 */
public class SummingUtil {

    static final Set<Collector.Characteristics> CH_CONCURRENT_ID
            = Collections.unmodifiableSet(EnumSet.of(Collector.Characteristics.CONCURRENT,
            Collector.Characteristics.UNORDERED,
            Collector.Characteristics.IDENTITY_FINISH));
    static final Set<Collector.Characteristics> CH_CONCURRENT_NOID
            = Collections.unmodifiableSet(EnumSet.of(Collector.Characteristics.CONCURRENT,
            Collector.Characteristics.UNORDERED));
    static final Set<Collector.Characteristics> CH_ID
            = Collections.unmodifiableSet(EnumSet.of(Collector.Characteristics.IDENTITY_FINISH));
    static final Set<Collector.Characteristics> CH_UNORDERED_ID
            = Collections.unmodifiableSet(EnumSet.of(Collector.Characteristics.UNORDERED,
            Collector.Characteristics.IDENTITY_FINISH));
    static final Set<Collector.Characteristics> CH_NOID = Collections.emptySet();

    @SuppressWarnings("unchecked")
    private static <I, R> Function<I, R> castingIdentity() {
        return i -> (R) i;
    }

    /**
     * Returns a {@code Collector} that produces the sum of a double-valued
     * function applied to the input elements.  If no elements are present,
     * the result is 0.
     *
     * <p>The sum returned can vary depending upon the order in which
     * values are recorded, due to accumulated rounding error in
     * addition of values of differing magnitudes. Values sorted by increasing
     * absolute magnitude tend to yield more accurate results.  If any recorded
     * value is a {@code NaN} or the sum is at any point a {@code NaN} then the
     * sum will be {@code NaN}.
     *
     * @param <T>    the type of the input elements
```

```
     * @param mapper a function extracting the property to be summed
     * @return a {@code Collector} that produces the sum of a derived property
     */
    public static <T> Collector<T, ?, BigDecimal>
    summingBigDecimal(ToBigDecimalFunction<? super T> mapper) {
        /*
         * In the arrays allocated for the collect operation, index 0
         * holds the high-order bits of the running sum, index 1 holds
         * the low-order bits of the sum computed via compensated
         * summation, and index 2 holds the simple sum used to compute
         * the proper result if the stream contains infinite values of
         * the same sign.
         */
        return new SummingUtil.CollectorImpl<>(
                () -> {
                    BigDecimal bArr[] = {new BigDecimal("0"), new BigDecimal("0"), new BigDecimal(
0")};
                    return bArr;
                },
                (a, t) -> {
                    sumWithCompensation(a, mapper.applyAsBigDecimal(t));
                    a[2].add(mapper.applyAsBigDecimal(t));
                },
                (a, b) -> {
                    sumWithCompensation(a, b[0]);
                    a[2].add(b[2]);
                    return sumWithCompensation(a, b[1]);
                },
                a -> computeFinalSum(a),
                CH_NOID);
    }

    static BigDecimal[] sumWithCompensation(BigDecimal[] intermediateSum, BigDecimal valu
) {
        BigDecimal tmp = value.subtract(intermediateSum[1]);
        BigDecimal sum = intermediateSum[0];
        BigDecimal velvel = sum.add(tmp); // Little wolf of rounding error
        intermediateSum[1] = (velvel.subtract(sum)).subtract(tmp);
        intermediateSum[0] = velvel;
        return intermediateSum;
    }

    /**
     * If the compensated sum is spuriously NaN from accumulating one
     * or more same-signed infinite values, return the
     * correctly-signed infinity stored in the simple sum.
     */
    static BigDecimal computeFinalSum(BigDecimal[] summands) {
        // Better error bounds to add both terms as the final sum
        BigDecimal tmp = summands[0].add(summands[1]);
        return tmp;
    }

    /**
```

```
 * Simple implementation class for {@code Collector}.
 *
 * @param <T> the type of elements to be collected
 * @param <R> the type of the result
 */
public static class CollectorImpl<T, A, R> implements Collector<T, A, R> {
    private final Supplier<A> supplier;
    private final BiConsumer<A, T> accumulator;
    private final BinaryOperator<A> combiner;
    private final Function<A, R> finisher;
    private final Set<Characteristics> characteristics;

    CollectorImpl(Supplier<A> supplier,
                  BiConsumer<A, T> accumulator,
                  BinaryOperator<A> combiner,
                  Function<A, R> finisher,
                  Set<Characteristics> characteristics) {
        this.supplier = supplier;
        this.accumulator = accumulator;
        this.combiner = combiner;
        this.finisher = finisher;
        this.characteristics = characteristics;
    }

    CollectorImpl(Supplier<A> supplier,
                  BiConsumer<A, T> accumulator,
                  BinaryOperator<A> combiner,
                  Set<Characteristics> characteristics) {
        this(supplier, accumulator, combiner, castingIdentity(), characteristics);
    }

    @Override
    public BiConsumer<A, T> accumulator() {
        return accumulator;
    }

    @Override
    public Supplier<A> supplier() {
        return supplier;
    }

    @Override
    public BinaryOperator<A> combiner() {
        return combiner;
    }

    @Override
    public Function<A, R> finisher() {
        return finisher;
    }

    @Override
    public Set<Characteristics> characteristics() {
        return characteristics;
```

```
        }
    }
}
```

原文链接：实现 java8 分组时使用 BigDecimal