@daebalprime

Effective Java 3/E Chapter 9

목차

목차

Google Guava

구아바를 사용할 만한 동기

1. 구글이 만들고 Effective Java도 개발에 기여함
2. 여러 패턴에 대한 쉬운 사용
3. 일관성 있는 API

Google Guava

구아바가 제공하는 기능

- Basic utilities: Make using the Java language more pleasant.
  - Using and avoiding null: `null` can be ambiguous, can cause confusing errors, and is sometimes just plain unpleasant. Many Guava utilities reject and fail fast on nulls, rather than accepting them blindly.
  - Preconditions: Test preconditions for your methods more easily.
  - Common object methods: Simplify implementing `Object` methods, like `hashCode()` and `toString()`.
  - Ordering: Guava's powerful "fluent `Comparator`" class.
  - Throwables: Simplify propagating and examining exceptions and errors.
- Collections: Guava's extensions to the JDK collections ecosystem. These are some of the most mature and popular parts of Guava.
  - Immutable collections, for defensive programming, constant collections, and improved efficiency.
  - New collection types, for use cases that the JDK collections don't address as well as they could: multisets, multimaps, tables, bidirectional maps, and more.
  - Powerful collection utilities, for common operations not provided in `java.util.Collections`.
  - Extension utilities: writing a `Collection` decorator? Implementing `Iterator`? We can make that easier.
- Graphs: a library for modeling graph-structured data, that is, entities and the relationships between them. Key features include:
  - Graph: a graph whose edges are anonymous entities with no identity or information of their own.
  - ValueGraph: a graph whose edges have associated non-unique values.
  - Network: a graph whose edges are unique objects.
  - Support for graphs that are mutable and immutable, directed and undirected, and several other properties.
- Caches: Local caching, done right, and supporting a wide variety of expiration behaviors.
- Functional idioms: Used sparingly, Guava's functional idioms can significantly simplify code.
- Concurrency: Powerful, simple abstractions to make it easier to write correct concurrent code.
  - ListenableFuture: Futures, with callbacks when they are finished.
  - Service: Things that start up and shut down, taking care of the difficult state logic for you.

Google Guava

concat(Iterable<Iterable>)

여러 Iterable 인스턴스를 합친 lazy view

실제로 Iterable을 합친 결과물을 들고 있지는 않음

```
Iterable<Integer> concatenated = Iterables.concat(
  Ints.asList(1, 2, 3),
  Ints.asList(4, 5, 6));
// concatenated has elements 1, 2, 3, 4, 5, 6

String lastAdded = Iterables.getLast(myLinkedHashSet);

String theElement = Iterables.getOnlyElement(thisSetIsDefinitelyASingleton);
  // if this set isn't a singleton, something is wrong!
```

General

Method	Description	See Also
`concat(Iterable<Iterable>)`	Returns a lazy view of the concatenation of several iterables.	`concat(Iterable...)`
`frequency(Iterable, Object)`	Returns the number of occurrences of the object.	Compare `Collections.frequency(Collection, Object)`; see `Multiset`
`partition(Iterable, int)`	Returns an unmodifiable view of the iterable partitioned into chunks of the specified size.	`Lists.partition(List, int)`, `paddedPartition(Iterable, int)`
`getFirst(Iterable, T default)`	Returns the first element of the iterable, or the default value if empty.	Compare `Iterable.iterator().next()`, `FluentIterable.first()`
`getLast(Iterable)`	Returns the last element of the iterable, or fails fast with a `NoSuchElementException` if it's empty.	`getLast(Iterable, T default)`, `FluentIterable.last()`
`elementsEqual(Iterable, Iterable)`	Returns true if the iterables have the same elements in the same order.	Compare `List.equals(Object)`
`unmodifiableIterable(Iterable)`	Returns an unmodifiable view of the iterable.	Compare `Collections.unmodifiableCollection(Collection)`
`limit(Iterable, int)`	Returns an `Iterable` returning at most the specified number of elements.	`FluentIterable.limit(int)`
`getOnlyElement(Iterable)`	Returns the only element in `Iterable`. Fails fast if the iterable is empty or has multiple elements.	`getOnlyElement(Iterable, T default)`

Google Guava

Guava 캐싱의 구현

- 캐싱할 데이터를 map처럼 사용
- LoadingCache 타입의 변수로 캐시 선언, CacheBuilder로 캐시 생성
- CacheLoader의 load 메서드를 재정의하여 캐시 miss-> 캐시 불러오는 로직 구현
- Size-based Eviction, Timed Eviction, Reference-based Eviction 정책(GC가 eviction을 수행)
- Guava의 캐시 클린업은 캐시에 접근하거나 캐시 load 시 실행

Google Guava

Guava 캐시 Size-based Eviction

```
LoadingCache<Key, Graph> graphs = CacheBuilder.newBuilder()
       .maximumSize(1000)
       .build(
           new CacheLoader<Key, Graph>() {
             public Graph load(Key key) throws AnyException {
               return createExpensiveGraph(key);
             }
           });

...
try {
  return graphs.get(key);
} catch (ExecutionException e) {
  throw new OtherException(e.getCause());
}
```

Google Guava

Guava 캐시 Timed Eviction

- expireAfterAccess(long, TimeUnit)
- expireAfterWrite(long, TimeUnit)

```
LoadingCache<Key, Graph> graphs = CacheBuilder.newBuilder()
       .expireAfterAccess(10, TimeUnit.MINUTES)
       .build(
           new CacheLoader<Key, Graph>() {
             public Graph load(Key key) { // no checked exception
               return createExpensiveGraph(key);
             }
           });

...
return graphs.getUnchecked(key);
```

Google Guava

기타 메서드

- expireAfterAccess(long, TimeUnit)
  - hitRate()
  - averageLoadPenalty()
  - evictionCount()
- Cache.invalidate(key)
- Cache.invalidateAll(keys)
- Cache.invalidateAll()

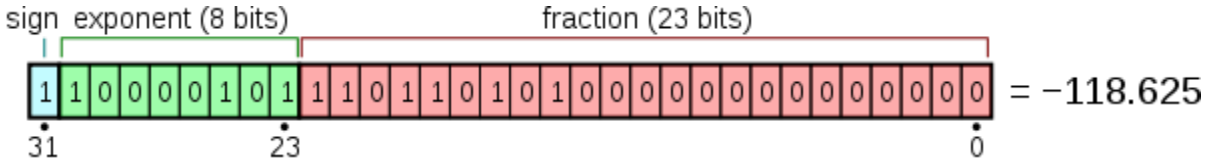
부동소숫점 오류의 원인

IEEE 754

- **IEEE 채택 부동소수점 표준**
- **sign bit(0-1, 음수 양수), exponent bit(-126~127), fraction bit**
- **Decimal 실수를 Binary 실수로 변환하여 저장**

−118.625(십진법)를 IEEE 754(32비트 단정밀도)로 표현해 보자.

- 음수이므로, 부호부는 1이 된다.
- 그 다음, 절댓값을 이진법으로 나타내면 $1110110.101_{(2)}$이 된다.
- 소수점을 왼쪽으로 이동시켜, 왼쪽에는 1만 남게 만든다. 예를 들면 $1110110.101_{(2)}=1.110110101_{(2)}\times 2^{6}$ 과 같다. 이것을 정규화된 부동소수점 수라고 한다.
- 가수부는 소수점의 오른쪽 부분으로, 부족한 비트 수 부분만큼 0으로 채워 23비트로 만든다. 결과는 11011010100000000000000 이 된다.
- 지수는 6이므로, Bias를 더해야 한다. 32비트 IEEE 754 형식에서는 Bias는 127이므로 6+127 = 133이 된다. 이진법으로 변환하면 $10000101_{(2)}$이 된다.



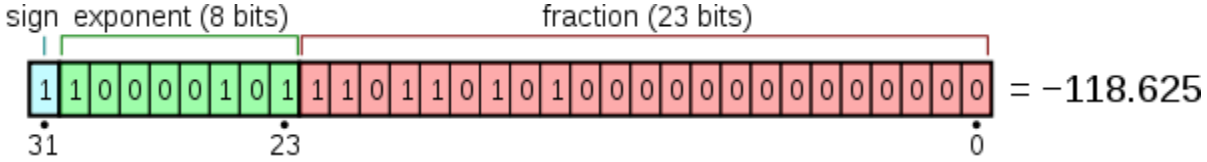
부동소숫점 오류의 원인

IEEE 754 표준의 저장을 자연어로 설명

임의의 실수를 (1+(1/2^n1)+(1/2^n2)+(1/2^n3)+...+(1/2^n23))*10^r로 표현

실수가 $\frac{1}{2^n}$ 의 합으로 표현이 가능할 때에는 부동소수점 오류가 발생하지 않음!

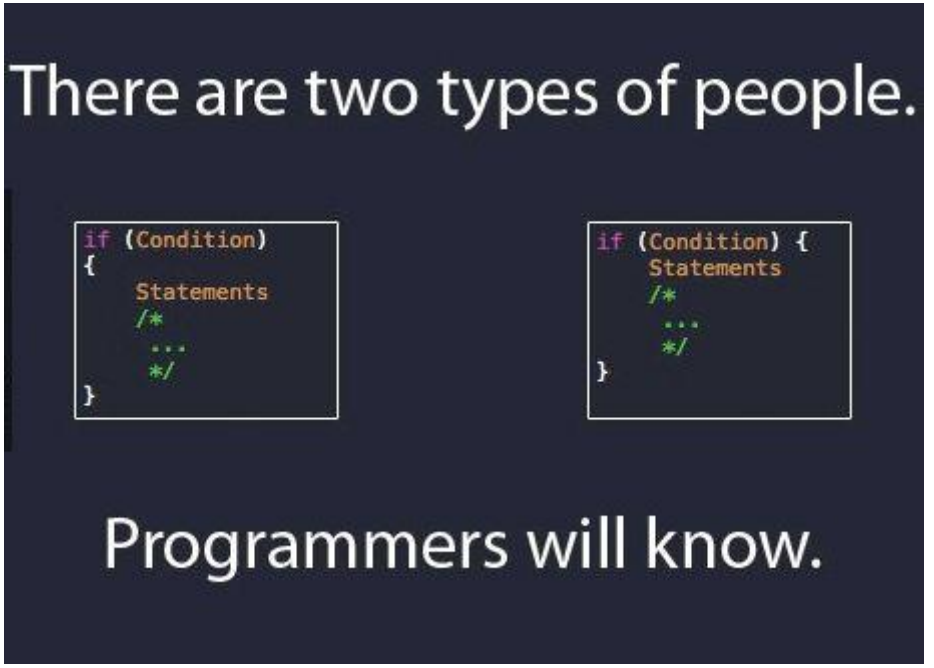
```
>>> 0.625 * 7
4.375
>>> 0.142857*7
0.9999990000000001
```



Google Java Guide

if, else, for, do, while -> body가 비어있거나 단 한 줄의 statement가 있더라도 괄호 사용한다. (“Egyptian Brackets”)

```
return new MyClass() {
  @Override public void method() {
    if (condition()) {
      try {
        something();
      } catch (ProblemException e) {
        recover();
      }
    } else if (otherCondition()) {
      somethingElse();
    } else {
      lastThing();
    }
  }
};
```



Google Java Guide

빈 블록은 줄바꿈이 선택사항이다.

멀티 블록 구문에서는 허용되지 않는다.

```
// 허용
void doNothing() {}

// 허용
void doNothingElse() {
}
```

```
// 허용되지 않음 : 멀티 블럭 구문에서는 간결한 빈 블럭을 사용할 수 없다.
try {
  doSomething();
} catch (Exception e) {}
```

StringBuilder 구현

StringBuilder extends AbstractStringBuilder. 대부분의 메서드 호출을 delegate

전체적인 구조는 ArrayList와 유사하다.

내부에 char 배열로 String을 저장한다.

초기 사이즈는 지정하지 않으면 16으로 주어진다.

```
public final class StringBuilder
    extends AbstractStringBuilder
    implements java.io.Serializable, CharSequence
{
    public StringBuilder() {
        super(16);
    }
…
```

StringBuilder 구현

AbstractStringBuilder 내부에 char 배열로 String을 저장한다.

```
abstract class AbstractStringBuilder implements Appendable,
CharSequence {

    char[] value;
    int count;

    AbstractStringBuilder(int capacity) {
        value = new char[capacity];
    }
```

StringBuilder 구현

내부 char 배열에 인수로 들어온 스트링을 copy하는 append()

```
public AbstractStringBuilder append(String str) {
    if (str == null)
        return appendNull();
    int len = str.length();
    ensureCapacityInternal(count + len);
    str.getChars(0, len, value, count);
    count += len;
    return this;
}
```

StringBuilder 구현

append() 오버로딩은 primitive type별로 되어있다.

```
    public AbstractStringBuilder append(int i) {
        if (i == Integer.MIN_VALUE) {
            append("-2147483648");
            return this;
        }
        int appendedLength = (i < 0) ? Integer.stringSize(-i) +
1
                                     : Integer.stringSize(i);
        int spaceNeeded = count + appendedLength;
        ensureCapacityInternal(spaceNeeded);
        Integer.getChars(i, spaceNeeded, value);
        count = spaceNeeded;
        return this;
    }
```

StringBuilder 구현

append() 오버로딩은 primitive type별로 되어있다.

```
public AbstractStringBuilder append(boolean b) {
        if (b) {
            ensureCapacityInternal(count + 4);
            value[count++] = 't';
            value[count++] = 'r';
            value[count++] = 'u';
            value[count++] = 'e';
        } else {
            ensureCapacityInternal(count + 5);
            value[count++] = 'f';
            value[count++] = 'a';
            value[count++] = 'l';
            value[count++] = 's';
            value[count++] = 'e';
        }
        return this;
    }
```

StringBuilder 구현

append 할 때 마다 공간이 충분한지 확인하고, 충분하지 않다면 value 크기를 늘린다.

```
private void ensureCapacityInternal(int minimumCapacity) {
        // overflow-conscious code
        if (minimumCapacity - value.length > 0) {
            value = Arrays.copyOf(value,
                    newCapacity(minimumCapacity));
        }
    }
private int newCapacity(int minCapacity) {
        // overflow-conscious code
        int newCapacity = (value.length << 1) + 2;
        if (newCapacity - minCapacity < 0) {
            newCapacity = minCapacity;
        }
        return (newCapacity <= 0 || MAX_ARRAY_SIZE - newCapacity < 0)
            ? hugeCapacity(minCapacity)
            : newCapacity;
    }
```

StringBuilder 구현

캐싱 + 방어적 복사 구현으로 toString()

```
@Override
    public synchronized String toString() {
        if (toStringCache == null) {
            toStringCache = Arrays.copyOfRange(value, 0, count);
        }
        return new String(toStringCache, true);
    }
```

Reference

https://ko.wikipedia.org/wiki/IEEE_754 (부동소수점 위키백과)

https://github.com/google/guava/wiki (Guava Docs)

https://velog.io/@new_wisdom/JAVA-Google-Java-Style-Guide-%EB%B2%88%EC%97%AD
(구글 자바 가이드 번역)

https://google.github.io/styleguide/javaguide.html (구글 자바 가이드 원문)

http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java/lang/AbstractStringBuilder.java
(StringBuilder 소스코드)