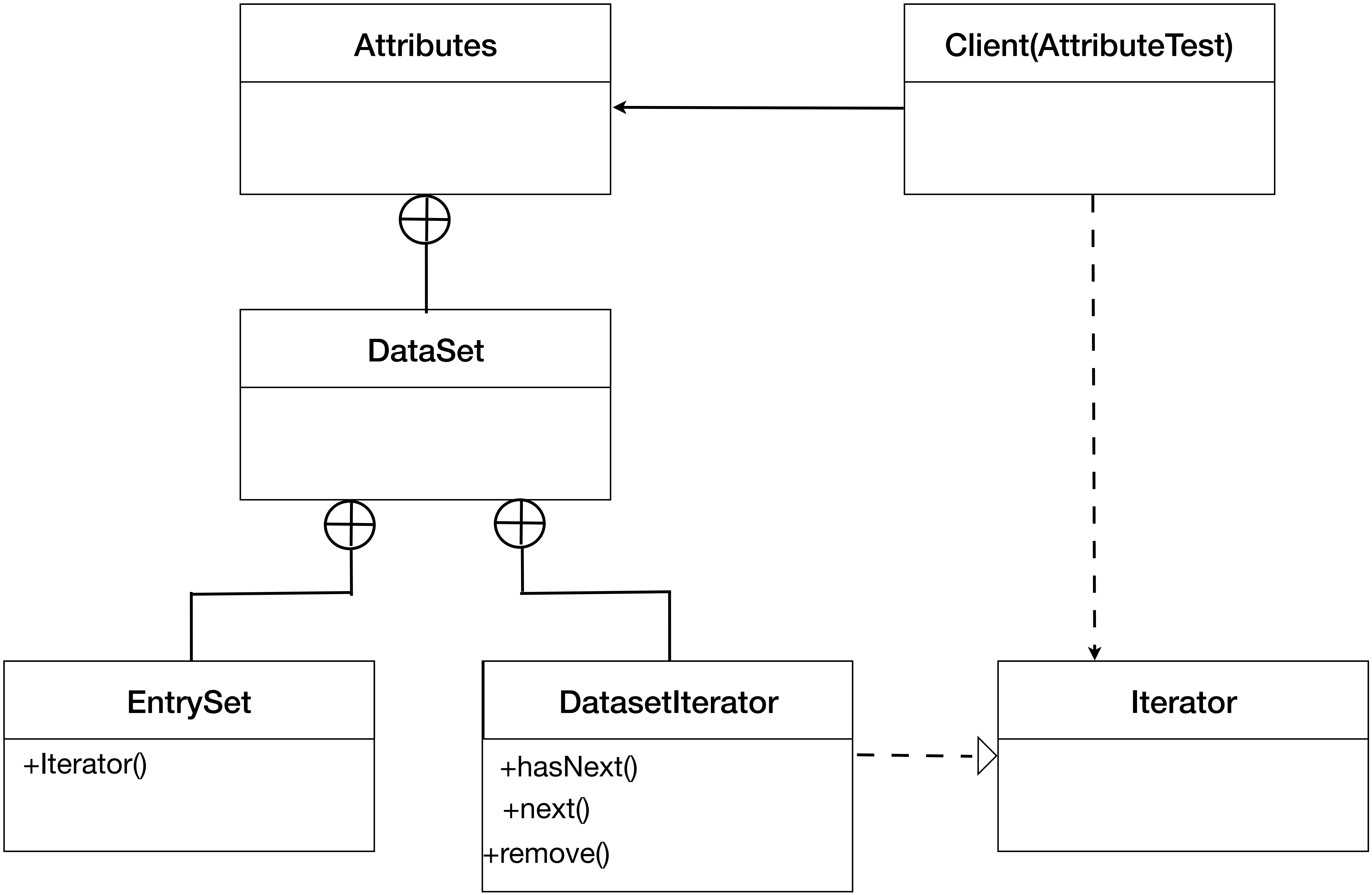


Attributes

Client(AttributeTest)

DataSet

EntrySet

+Iterator()

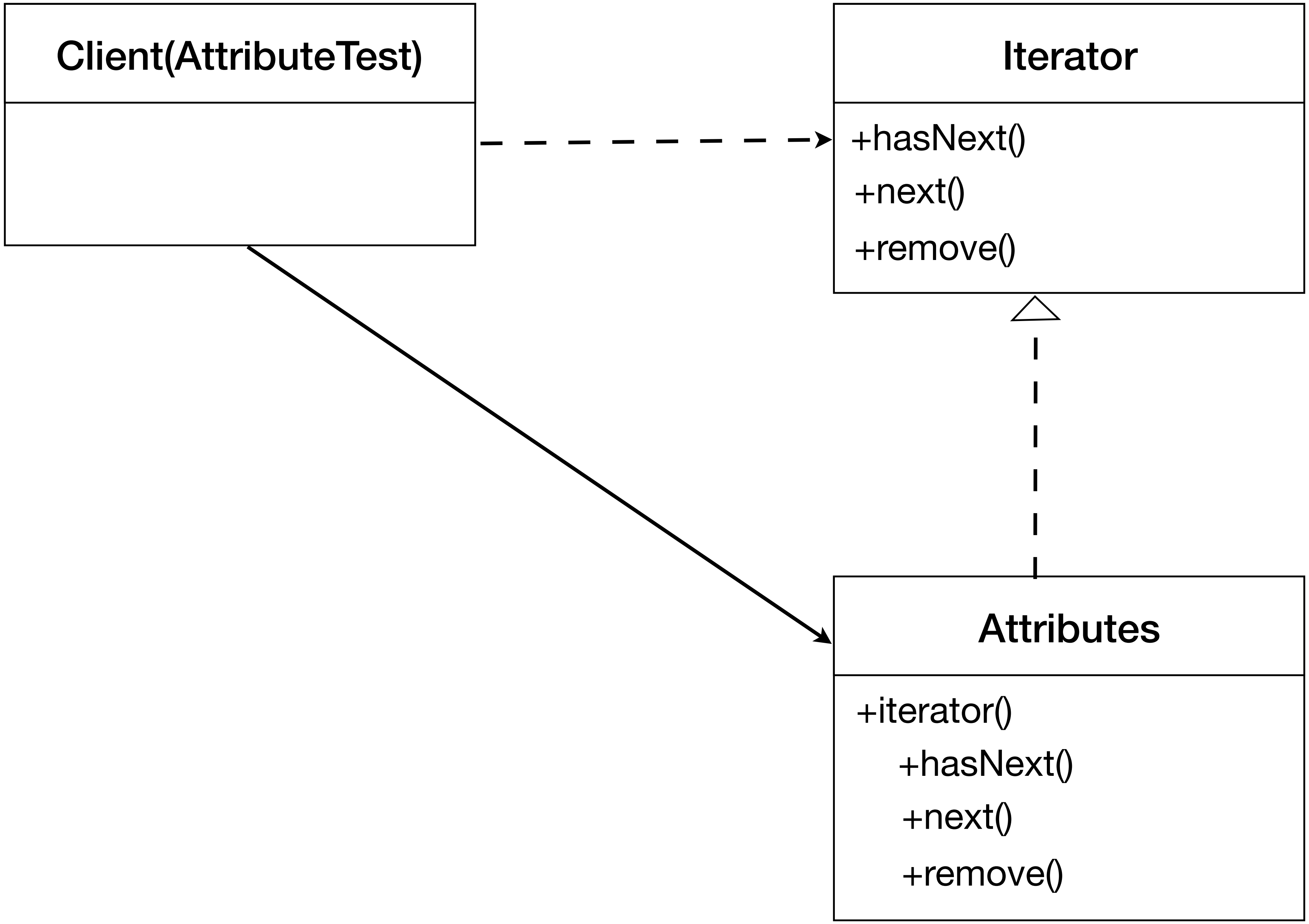
DatasetIterator

+hasNext()

+next()

+remove()

Iterator



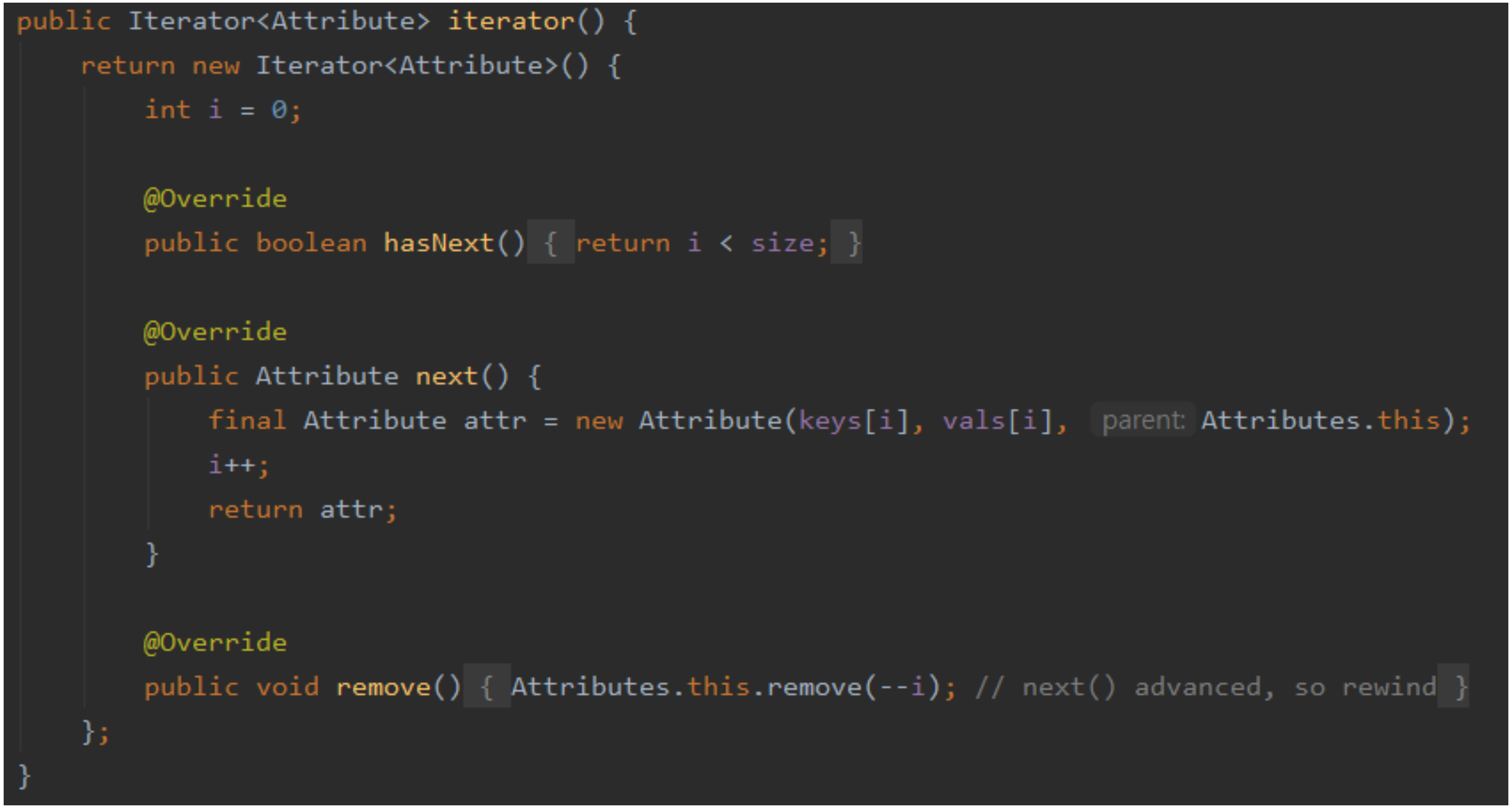
Client(AttributeTest)

Iterator

+hasNext()
+next()
+remove()

Attributes

+iterator()
+hasNext()
+next()
+remove()

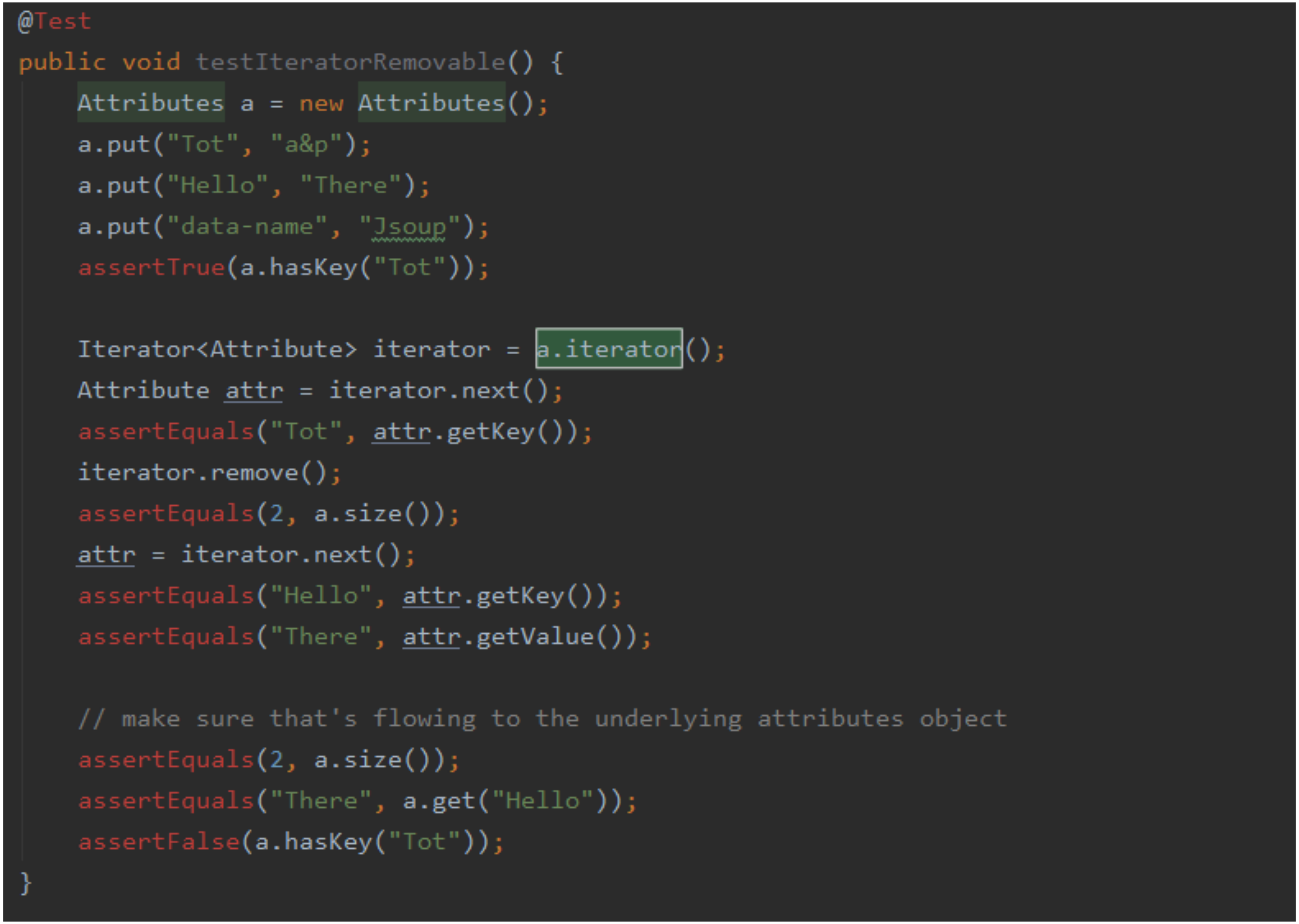


```
public Iterator<Attribute> iterator() {
    return new Iterator<Attribute>() {
        int i = 0;

        @Override
        public boolean hasNext() { return i < size; }

        @Override
        public Attribute next() {
            final Attribute attr = new Attribute(keys[i], vals[i],  parent: Attributes.this);
            i++;
            return attr;
        }

        @Override
        public void remove() { Attributes.this.remove(--i); // next() advanced, so rewind }
    };
}
```



```java
@Test
public void testIteratorRemovable() {
    Attributes a = new Attributes();
    a.put("Tot", "a&p");
    a.put("Hello", "There");
    a.put("data-name", "Jsoup");
    assertTrue(a.hasKey("Tot"));

    Iterator<Attribute> iterator = a.iterator();
    Attribute attr = iterator.next();
    assertEquals("Tot", attr.getKey());
    iterator.remove();
    assertEquals(2, a.size());
    attr = iterator.next();
    assertEquals("Hello", attr.getKey());
    assertEquals("There", attr.getValue());

    // make sure that's flowing to the underlying attributes object
    assertEquals(2, a.size());
    assertEquals("There", a.get("Hello"));
    assertFalse(a.hasKey("Tot"));
}
```