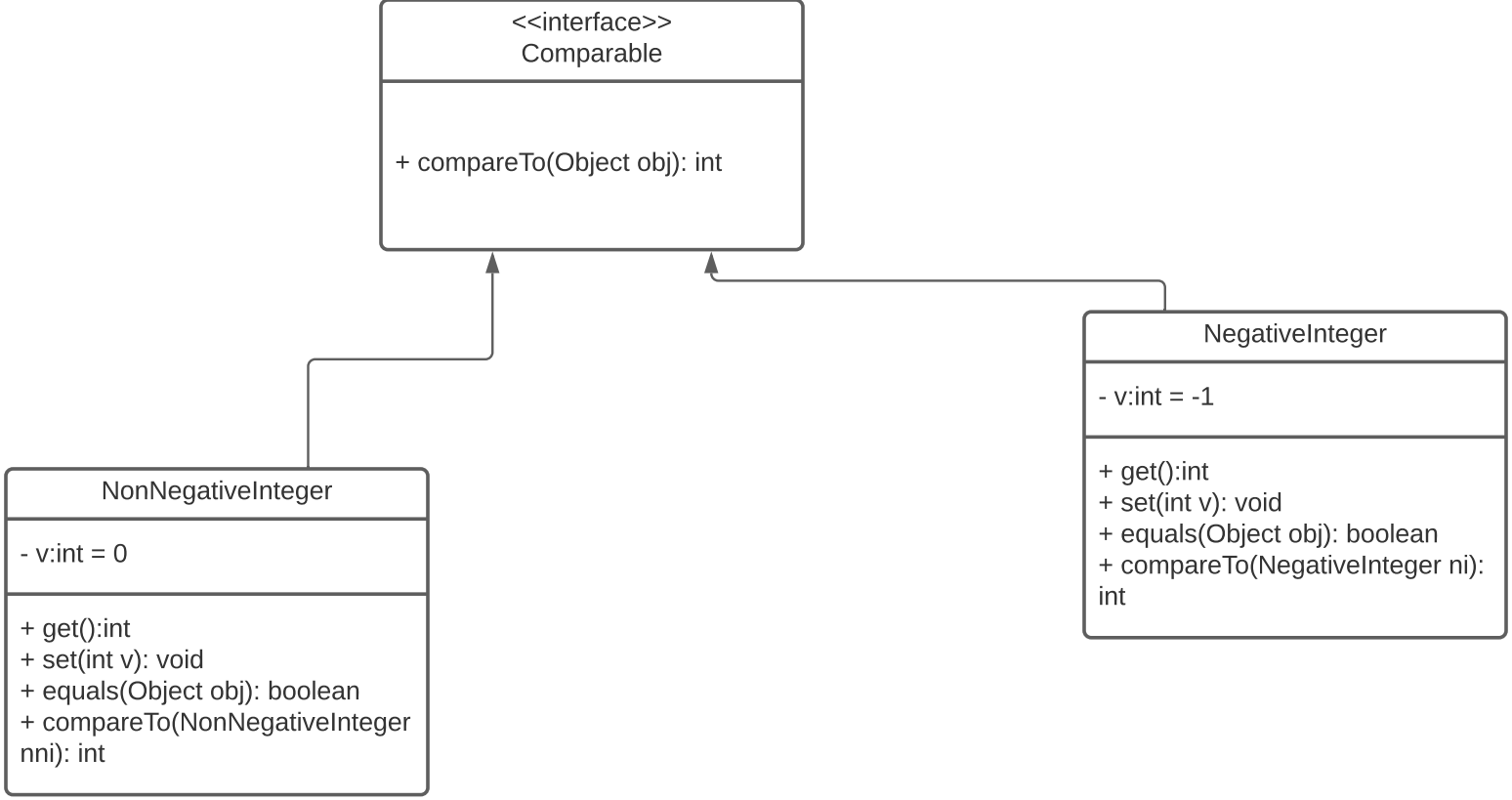
```
public class NonNegativeInteger implements Comparable<NonNegativeInteger> {
    public NonNegativeInteger() {
        v = 0;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof NonNegativeInteger)) return false;
        NonNegativeInteger nni = (NonNegativeInteger)obj;
        return v == nni.get();
    }

    @Override
    public int hashCode() {
        int result = Integer.hashCode(v);
        return 31 * result;
    }

    public int compareTo(NonNegativeInteger nni) {
        return Integer.compare(v, nni.v);
    }

    public void set(int v) throws IllegalArgumentException {
        if (v < 0) {
            throw new IllegalArgumentException("" +
                    "You cannot pass a negative integer to this method.");
        }
        this.v = v;
    }

    public int get() {
        return v;
    }

    private int v;
}


import org.junit.Test;

import static org.junit.Assert.*;

public class NonNegativeIntegerTest {
    @Test
    public void getTest() {
        NonNegativeInteger nni = new NonNegativeInteger();
        NonNegativeInteger nni2 = new NonNegativeInteger();

        assertEquals(nni.get(), 0);

        nni.set(5);
        assertEquals(nni.get(), 5);
```

```
        try {
            nni2.set(-1);
        } catch (IllegalArgumentException e) {
            assertEquals(nni2.get(), 0);
            nni2.set(5);
            assertEquals(nni2.get(), 5);
        }
    }

    @Test
    public void setTest() {
        NonNegativeInteger nni = new NonNegativeInteger();

        try {
            nni.set(-1);
        } catch (IllegalArgumentException e) {
            nni.set(5);
            assertEquals(nni.get(), 5);
            nni.set(25);
            assertEquals(nni.get(), 25);
        }
    }

    @Test
    public void equalsTest() {
        NonNegativeInteger nni = new NonNegativeInteger();
        NonNegativeInteger nni2 = new NonNegativeInteger();

        assertTrue(nni.equals(nni2));
        assertTrue(nni2.equals(nni));

        nni.set(5);
        nni2.set(5);

        assertTrue(nni.equals(nni2));
        assertTrue(nni2.equals(nni));

        nni.set(3);
        nni2.set(25);

        assertFalse(nni.equals(nni2));
        assertFalse(nni2.equals(nni));
    }
}

public class NegativeInteger implements Comparable<NegativeInteger> {
    public NegativeInteger() {
        v = -1;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
```

```
        if (!(obj instanceof NegativeInteger)) return false;
        NegativeInteger ni = (NegativeInteger)obj;
        return v == ni.get();
    }

    @Override
    public int hashCode() {
        int result = Integer.hashCode(v);
        return 31 * result;
    }

    public int compareTo(NegativeInteger ni) {
        return Integer.compare(v, ni.v);
    }

    public void set(int v) throws IllegalArgumentException {
        if (v >= 0) {
            throw new IllegalArgumentException("" +
                    "You cannot pass a positive integer to this method.");
        }
        this.v = v;
    }

    public int get() {
        return v;
    }

    private int v;
}
```


<<interface>>
Comparable
+ compareTo(Object obj): int
NonNegativeInteger
- v:int = 0
+ get():int
+ set(int v): void
+ equals(Object obj): boolean
+ compareTo(NonNegativeInteger nni): int
NegativeInteger
- v:int = -1
+ get():int
+ set(int v): void
+ equals(Object obj): boolean
+ compareTo(NegativeInteger ni): int