```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 * methods.
 *
 * @author Put your name here
 *
 */
public abstract class NaturalNumberTest {

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @return the new number
     * @ensures constructorTest = 0
     */
    protected abstract NaturalNumber constructorTest();

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param i
     *            {@code int} to initialize from
     * @return the new number
     * @requires i >= 0
     * @ensures constructorTest = i
     */
    protected abstract NaturalNumber constructorTest(int i);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param s
     *            {@code String} to initialize from
     * @return the new number
     * @requires there exists n: NATURAL (s = TO_STRING(n))
     * @ensures s = TO_STRING(constructorTest)
     */
    protected abstract NaturalNumber constructorTest(String s);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param n
     *            {@code NaturalNumber} to initialize from
     * @return the new number
     * @ensures constructorTest = n
     */
    protected abstract NaturalNumber constructorTest(NaturalNumber n);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * reference implementation and returns the result.
     *
     * @return the new number
     * @ensures constructorRef = 0
     */
    protected abstract NaturalNumber constructorRef();

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * reference implementation and returns the result.
     *
     * @param i
     *            {@code int} to initialize from
     * @return the new number
     * @requires i >= 0
     * @ensures constructorRef = i
     */
    protected abstract NaturalNumber constructorRef(int i);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * reference implementation and returns the result.
     *
     * @param s
     *            {@code String} to initialize from
     * @return the new number
     * @requires there exists n: NATURAL (s = TO_STRING(n))
     * @ensures s = TO_STRING(constructorRef)
     */
    protected abstract NaturalNumber constructorRef(String s);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * reference implementation and returns the result.
     *
     * @param n
     *            {@code NaturalNumber} to initialize from
     * @return the new number
     * @ensures constructorRef = n
     */
    protected abstract NaturalNumber constructorRef(NaturalNumber n);

    // TODO - add test cases for four constructors, multiplyBy10, divideBy10, isZero

    @Test
    public final void testForDefaultConstructor() {
        NaturalNumber n = this.constructorTest();
        NaturalNumber nExp = this.constructorRef();

        assertEquals(n, nExp);
    }

    @Test
    public final void testForIntConstructorZero() {
        int i = 0;

        NaturalNumber n = this.constructorTest(i);
        NaturalNumber nExp = this.constructorRef(i);

        assertEquals(n, nExp);
    }
```

```java

    @Test
    public final void testForIntConstructor() {
        int i = 5;

        NaturalNumber n = this.constructorTest(i);
        NaturalNumber nExp = this.constructorRef(i);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForStringConstructor() {
        String s = "5";

        NaturalNumber n = this.constructorTest(s);
        NaturalNumber nExp = this.constructorRef(s);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForNaturalNumberConstructor() {
        NaturalNumber natN = new NaturalNumber1L(3);

        NaturalNumber n = this.constructorTest(natN);
        NaturalNumber nExp = this.constructorRef(natN);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10Zero() {

        NaturalNumber n = this.constructorTest(0);
        NaturalNumber nExp = this.constructorRef(0);

        n.multiplyBy10(0);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10LeadingZero() {

        NaturalNumber n = this.constructorTest(0);
        NaturalNumber nExp = this.constructorRef(7);

        n.multiplyBy10(7);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10Int() {

        NaturalNumber n = this.constructorTest(123);
        NaturalNumber nExp = this.constructorRef(1234);

        n.multiplyBy10(4);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10MaxInt() {

        NaturalNumber n = this.constructorTest(Integer.MAX_VALUE);
        NaturalNumber nExp = this.constructorTest("21474836470");

        n.multiplyBy10(0);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10String() {

        NaturalNumber n = this.constructorTest("678");
        NaturalNumber nExp = this.constructorRef("6789");

        n.multiplyBy10(9);

        assertEquals(n, nExp);
    }

    @Test
    public final void testForMultiplyBy10NN() {

        NaturalNumber nInitial = this.constructorRef(4567);
        NaturalNumber nFinal = this.constructorRef(45678);

        NaturalNumber n = this.constructorTest(nInitial);
        NaturalNumber nExp = this.constructorRef(nFinal);
        n.multiplyBy10(8);

        assertEquals(n, nExp);
    }

    @Test
    public final void testDivideBy10Zero() {
        NaturalNumber n = this.constructorTest(0);
        NaturalNumber nExp = this.constructorRef(0);

        int dig = n.divideBy10();

        assertEquals(n, nExp);
        assertEquals(0, dig);
    }

    @Test
    public final void testDivideBy10OneDigit() {
        NaturalNumber n = this.constructorTest(7);
        NaturalNumber nExp = this.constructorRef(0);

        int dig = n.divideBy10();

        assertEquals(n, nExp);
        assertEquals(7, dig);
    }

    @Test
    public final void testDivideBy10Int() {
        NaturalNumber n = this.constructorTest(12345);
        NaturalNumber nExp = this.constructorRef(1234);

        int dig = n.divideBy10();

        assertEquals(n, nExp);
        assertEquals(5, dig);
    }

    @Test
    public final void testDivideBy10String() {
        NaturalNumber n = this.constructorTest("98765");
        NaturalNumber nExp = this.constructorRef("9876");

        int dig = n.divideBy10();

        assertEquals(n, nExp);
        assertEquals(5, dig);
    }

    @Test
    public final void testDivideBy10NN() {
        NaturalNumber nInitial = this.constructorRef(45678);
        NaturalNumber nFinal = this.constructorRef(4567);

        NaturalNumber n = this.constructorTest(nInitial);
        NaturalNumber nExp = this.constructorRef(nFinal);

        int dig = n.divideBy10();

        assertEquals(n, nExp);
        assertEquals(8, dig);
    }

    @Test
    public final void testForIsZeroWhenZero() {
        NaturalNumber n = this.constructorTest();

        assertEquals(true, n.isZero());
    }

    @Test
    public final void testForIsZeroWhenNonZero() {
        NaturalNumber n = this.constructorTest(13);

        assertEquals(false, n.isZero());
    }

}
```