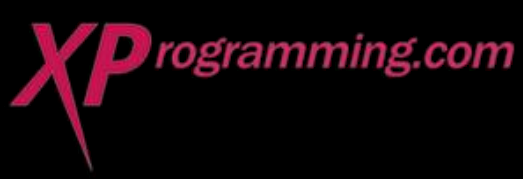
Prime Factors Kata

Object Mentor, Inc.

www.objectmentor.com

blog.objectmentor.com

fitnesse.org



www.junit.org

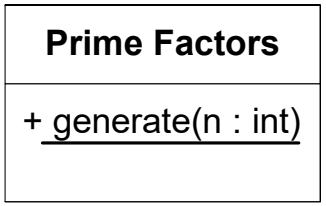
Generating Prime Factors.

Although quite short, this kata is fascinating in the way it shows how ‘if’ statements become ‘while’ statements as the number of test cases increase. It’s also a wonderful example of how algorithms sometimes become simpler as they become more general.

I stumbled upon this little kata one evening when my son was in 7th grade. He had just discovered that all numbers can be broken down into a product of primes and was interested in exploring this further. So I wrote a little ruby program, test-first, and was stunned by how the algorithm evolved.

I have done this particular kata in Java 5.0. This should give you a feel for the power and convenience of some of the new features.

The Requirements.



- Write a class named “PrimeFactors” that has one static method: generate.
  - The generate method takes an integer argument and returns a List<Integer>. That list contains the prime factors in numerical sequence.

Begin.

- Create a project named PrimeFactors
- Create a package named primeFactors
- Create a unit test named PrimeFactorsTest

```
package primeFactors;

import junit.framework.TestCase;

public class PrimeFactorsTest extends TestCase {
}
```

No tests found in primeFactors.PrimeFactorsTest

The first test.

```
package primeFactors;

import junit.framework.TestCase;

public class PrimeFactorsTest extends TestCase {
  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }
}
```

The first test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.List;

public class PrimeFactorsTest extends TestCase {
  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  private List<Integer> list() {
    return null;
  }
}
```

The first test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.List;

public class PrimeFactorsTest extends TestCase {
  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  private List<Integer> list() {
    return null;
  }
}
```

```
package primeFactors;

public class PrimeFactors {
}
```

The first test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.List;

public class PrimeFactorsTest extends TestCase {
  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  private List<Integer> list() {
    return null;
  }
}
```

```
expected:<null> but was:<[]>
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    return new ArrayList<Integer>();
  }
}
```

The first test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.*;

public class PrimeFactorsTest extends TestCase {
  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  private List<Integer> list() {
    return new ArrayList<Integer>();
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    return new ArrayList<Integer>();
  }
}
```

The first test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list() {
    return new ArrayList<Integer>();
  }

  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    return new ArrayList<Integer>();
  }
}
```

The Second Test

The Second test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list() {
    return new ArrayList<Integer>();
  }

  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),PrimeFactors.generate(2));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    return new ArrayList<Integer>();
  }
}
```

The Second test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),PrimeFactors.generate(2));
  }
}
```

varargs

```
expected:<[2]> but was:<[]>
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    return new ArrayList<Integer>();
  }
}
```

The Second test.

```
package primeFactors;

import junit.framework.TestCase;

import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),PrimeFactors.generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),PrimeFactors.generate(2));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      primes.add(2);
    }
    return primes;
  }
}
```

The Second test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      primes.add(2);
    }
    return primes;
  }
}
```

The Third Test

The Third test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }
}
```

expected:<[3]> but was:<[2]>

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      primes.add(2);
    }
    return primes;
  }
}
```

The Third test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      primes.add(n);
    }
    return primes;
  }
}
```

The Fourth Test

The Fourth test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      primes.add(n);
    }
    return primes;
  }
}
```

expected:<[2, 2]> but was:<[4]>

The Fourth test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      if (n%2 == 0) {
        primes.add(2);
        n /= 2;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

The Fifth Test

The Fifth test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      if (n%2 == 0) {
        primes.add(2);
        n /= 2;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

The Sixth Test

The Sixth test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      if (n%2 == 0) {
        primes.add(2);
        n /= 2;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

expected:<[2, 2, 2]> but was:<[2, 4]>

The Sixth test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      while (n%2 == 0) {
        primes.add(2);
        n /= 2;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

The Seventh Test

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      while (n%2 == 0) {
        primes.add(2);
        n /= 2;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

expected:<[3, 3]> but was:<[9]>

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      int candidate = 2;
      while (n%candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
      if (n > 1)
        primes.add(n);
    }
    return primes;
  }
}
```

expected:<[3, 3]> but was:<[9]>

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    if (n > 1) {
      int candidate = 2;
      while (n % candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
    }
    if (n > 1)
      primes.add(n);
    return primes;
  }
}
```

expected:<[3, 3]> but was:<[9]>

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    int candidate = 2;
    if (n > 1) {
      while (n % candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
    }
    if (n > 1)
      primes.add(n);
    return primes;
  }
}
```

expected:<[3, 3]> but was:<[9]>

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    int candidate = 2;
    if (n > 1) {
      while (n % candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
    }
    if (n > 1)
      primes.add(n);
    return primes;
  }
}
```

expected:<[3, 3]> but was:<[9]>

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    int candidate = 2;
    while (n > 1) {          <-- !!!
      while (n % candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
      candidate++;
    }
    if (n > 1)
      primes.add(n);
    return primes;
  }
}
```

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    int candidate = 2;
    while (n > 1) {
      while (n % candidate == 0) {
        primes.add(candidate);
        n /= candidate;
      }
      candidate++;
    }
    return primes;
  }
}
```

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();
    int candidate = 2;
    while (n > 1) {
      for (; n%candidate == 0; n/=candidate)
        primes.add(candidate);

      candidate++;
    }
    return primes;
  }
}
```

The Seventh test.

```
package primeFactors;

import static primeFactors.PrimeFactors.generate;
import junit.framework.TestCase;
import java.util.*;

public class PrimeFactorsTest extends TestCase {
  private List<Integer> list(int... ints) {
    List<Integer> list = new ArrayList<Integer>();
    for (int i : ints)
      list.add(i);
    return list;
  }

  public void testOne() throws Exception {
    assertEquals(list(),generate(1));
  }

  public void testTwo() throws Exception {
    assertEquals(list(2),generate(2));
  }

  public void testThree() throws Exception {
    assertEquals(list(3),generate(3));
  }

  public void testFour() throws Exception {
    assertEquals(list(2,2),generate(4));
  }

  public void testSix() throws Exception {
    assertEquals(list(2,3),generate(6));
  }

  public void testEight() throws Exception {
    assertEquals(list(2,2,2),generate(8));
  }

  public void testNine() throws Exception {
    assertEquals(list(3,3),generate(9));
  }
}
```

```
package primeFactors;

import java.util.*;

public class PrimeFactors {
  public static List<Integer> generate(int n) {
    List<Integer> primes = new ArrayList<Integer>();

    for (int candidate = 2; n > 1; candidate++)
      for (; n%candidate == 0; n/=candidate)
        primes.add(candidate);

    return primes;
  }
}
```

The algorith is three lines of code!

END