Versión final

Como sabemos que no es muy recomendable usar demasiado a menudo la palabra reservada `static`, con unos pequeños cambios convertimos `GeneradorDePrimos` en una clase de la que se puedan crear distintos objetos.

En primer lugar, modificamos los casos de prueba con los que comprobaremos el funcionamiento de nuestro generador de primos:

```
import junit.framework.*;

public class GeneradorDePrimosTest extends TestCase
{
  GeneradorDePrimos generador;
  int[]             primos;

  public static void main(String args[])
  {
   junit.swingui.TestRunner.main(
     new String[] {"GeneradorDePrimosTest"});
  }

  public GeneradorDePrimosTest(String name)
  {
   super(name);
  }

  public void testPrimos0()
  {
    generador = new GeneradorDePrimos(0);
    primos = generador.getPrimos();
    assertEquals( primos.length, 0);
  }

  public void testPrimos2()
  {
    generador = new GeneradorDePrimos(2);
    primos = generador.getPrimos();
    assertEquals(primos.length, 1);
    assertEquals(primos[0], 2);
  }
```

```
  public void testPrimos3()
  {
    generador = new GeneradorDePrimos(3);
    primos = generador.getPrimos();
    assertEquals(primos.length, 2);
    assertEquals(primos[0], 2);
    assertEquals(primos[1], 3);
  }

  public void testPrimos100()
  {
    generador = new GeneradorDePrimos(100);
    primos = generador.getPrimos();
    assertEquals(primos.length, 25);
    assertEquals(primos[24], 97);
  }
}
```

A continuación, creamos un constructor para `GeneradorDePrimos` (que construye el vector de números primos) y añadimos un método `getPrimos()` con el cual acceder a este vector desde el exterior…

```
/**
 * Esta clase genera todos los números primos de 1
 * hasta un número máximo especificado por el usuario
 * utilizando la criba de Eratóstenes.
 * <p>
 * Dado un vector de enteros empezando en 2, se tachan
 * todos los múltiplos de 2. A continuación, se
 * encuentra el siguiente entero no tachado y se
 * tachan sus múltiplos. Los números que queden sin
 * tachar al final son los números primos entre 1 y N.
 *
 * @author Fernando Berzal
 * @version 3.0 Enero'2005 (FB)
 */

public class GeneradorDePrimos
{
  private boolean esPrimo[];
  private int primos[];
```

```
public GeneradorDePrimos (int max)
{
  if (max < 2) {

      primos = new int[0];  // Vector vacío

  } else {

     inicializarCandidatos(max);
     eliminarMultiplos();
     obtenerCandidatosNoEliminados();
  }
}


public int[] getPrimos ()
{
  return primos;
}


private void inicializarCandidatos (int max)
{
  int i;

  esPrimo = new boolean[max+1];

  esPrimo[0] = esPrimo[1] = false;

  for (i=2; i<esPrimo.length; i++)
      esPrimo[i] = true;
}


private void eliminarMultiplos ()
{
  int i;

  for (i=2; i<maxFactorPrimo(); i++)
      if (esPrimo[i])
         eliminarMultiplosDe(i);
}
```

```
  private int maxFactorPrimo ()
  {
    return (int) Math.sqrt(esPrimo.length) + 1;
  }


  private void eliminarMultiplosDe (int i)
  {
    int multiplo;

    for ( multiplo=2*i;
          multiplo<esPrimo.length;
          multiplo+=i )
        esPrimo[multiplo] = false;
  }


  private void obtenerCandidatosNoEliminados ()
  {
    int i, j;

    primos = new int[numPrimos()];

    for (i=0, j=0; i<esPrimo.length; i++) {
        if (esPrimo[i])
           primos[j++] = i;
    }
  }


  private int numPrimos ()
  {
    int i;
    int cuenta = 0;

    for (i=0; i<esPrimo.length; i++) {
        if (esPrimo[i])
           cuenta++;
    }

    return cuenta;
  }
}
```