**Declaring A BigInteger Variable (4 Examples)** 

|  |
| --- |
| **Java:** |
| [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) bigInt0 = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).ZERO;  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) bigInt1 = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).ONE;  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) bigInt3 = **new** [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) ("3");  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) bigInt5 = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).valueOf(5); |

Note: You cannot pass an int/long into a BigInteger directly, the easiest way to do this it to pass it as a static BigInteger using valueOf, Thank You "OneOffDriveByPoster" for that method. 

|  |
| --- |
| **Java:** |
| long num = 9876543210;  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) bigInt123 = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).valueOf (num); |

**So, How Do I Use Them?**   
  
Well, BigIntegers contain all the regular math functions, plus more, the main difference is, it does not use symbols such as '\*', '+','=','>' etc. rather it uses words (multiply,add,equals,compareTo). Examples;   
  
To Multiply: 

|  |
| --- |
| **Java:** |
| bigInt1.multiply(bigInt3); |

Returns: A Big Integer With Value 3 (1\*3)   
Whereas, bigInt1 and bigInt3 are both BigInteger's   
  
To Subtract: 

|  |
| --- |
| **Java:** |
| bigInt1.subtract(bigInt3); |

Returns: A BigInteger with value -2 (3-1)   
Whereas, bigInt1 and bigInt3 are both BigInteger's   
  
To Compare: 

|  |
| --- |
| **Java:** |
| bigInt1.compareTo(bigInt3); |

Returns: An Integer with a value of -1 (Less Than), 0 (Equal), 1 (Greater Then), in this case, it will be -1 since 1 is less then 3.   
Whereas, bigInt1 and bigInt3 are both BigInteger's 

To Check If Equal: 

|  |
| --- |
| **Java:** |
| bigInt1.equals(bigInt3); |

Returns: A boolean with a value of false since 1 != 3   
Whereas, bigInt1 and bigInt3 are both BigInteger's   
  
Youll notice, its just like [DEALING[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://compsci.ca/v3/viewtopic.php?t=13193)](http://compsci.ca/v3/viewtopic.php?t=13193) with strings (equals, compareTo)...   
  
As well, you can create a new BigInteger inline (outputs 8) 

|  |
| --- |
| **Java:** |
| [**System**](http://www.google.com/search?q=allinurl%3ASystem+java.sun.com&bntl=1).out.println ("2\*\*3 = "+**new** [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1)("2").pow(3)); |

Note: I Used this as a special example, 'pow', raises the BigInteger '2' to the power of a regular integer '3'. Most of the math methods need BigIntegers as the initial and secondary values but this is a worthy exception to point out.   
  
**Can I Make An Array?**   
  
Sure, why not, arrays can be made of any object/class. Both arrays have a length of 10, both have been initiallized to value '1' 

|  |
| --- |
| **Java:** |
| *//------Example One-------//*  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) i = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).ONE;  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) [] biggg = {i,i,i,i,i,i,i,i,i,i};   *//------Example Two-------//*  [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) [] biggg = **new** [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1) [10];  for (int c = 0; c < biggg.length; c ++){      biggg [c] = [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1).ONE;  } |

**You Mentioned Primality Testing...**   
  
Checking if a number is prime is easy with BigIntegers, 

|  |
| --- |
| **Java:** |
| if (**new** [**BigInteger**](http://www.google.com/search?q=allinurl%3ABigInteger+java.sun.com&bntl=1)("17").isProbablePrime(5) == **true**){          [**System**](http://www.google.com/search?q=allinurl%3ASystem+java.sun.com&bntl=1).out.println("17 Is Prime!");  } |

Note: The 5 is the certainty that the number is prime, this value reflects how long it takes to complete, for small primes, a value of 1-5 works, but bigger numbers may return false positives unless certainty is >= 5. 

 o  **remainder**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) remainder([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val) throws [ArithmeticException](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.ArithmeticException.html#_top_)

Returns a BigInteger whose value is (this % val). Throws an ArithmeticException if val == 0.

 o  **divideAndRemainder**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_)[] divideAndRemainder([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val) throws [ArithmeticException](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.ArithmeticException.html#_top_)

Returns an array of two BigIntegers. The first ([0]) element of the return value is the quotient (this / val), and the second ([1]) element is the remainder (this % val). Throws an ArithmeticException if val == 0.

 o  **pow**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) pow(int exponent) throws [ArithmeticException](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.ArithmeticException.html#_top_)

Returns a BigInteger whose value is (this \*\* exponent). Throws an ArithmeticException if exponent < 0 (as the operation would yield a non-integer value). Note that exponent is an integer rather than a BigInteger.

 o  **gcd**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) gcd([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns a BigInteger whose value is the greatest common denominator of abs(this) and abs(val). Returns 0 if this == 0 && val == 0.

 o  **abs**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) abs()

Returns a BigInteger whose value is the absolute value of this number.

 o  **negate**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) negate()

Returns a BigInteger whose value is (-1 \* this).

 o  **mod**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) mod([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) m)

Returns a BigInteger whose value is this mod m. Throws an ArithmeticException if m <= 0.

 o  **modPow**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) modPow([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) exponent,

[BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) m)

Returns a BigInteger whose value is (this \*\* exponent) mod m. (If exponent == 1, the returned value is (this mod m). If exponent < 0, the returned value is the modular multiplicative inverse of (this \*\* -exponent).) Throws an ArithmeticException if m <= 0.

 o  **modInverse**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) modInverse([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) m) throws [ArithmeticException](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.ArithmeticException.html#_top_)

Returns modular multiplicative inverse of this, mod m. Throws an ArithmeticException if m <= 0 or this has no multiplicative inverse mod m (i.e., gcd(this, m) != 1).

 o  **shiftLeft**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) shiftLeft(int n)

Returns a BigInteger whose value is (this << n). (Computes floor(this \* 2\*\*n).)

 o  **shiftRight**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) shiftRight(int n)

Returns a BigInteger whose value is (this >> n). Sign extension is performed. (Computes floor(this / 2\*\*n).)

 o  **and**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) and([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns a BigInteger whose value is (this & val). (This method returns a negative number iff this and val are both negative.)

 o  **or**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) or([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns a BigInteger whose value is (this | val). (This method returns a negative number iff either this or val is negative.)

 o  **xor**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) xor([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns a BigInteger whose value is (this ^ val). (This method returns a negative number iff exactly one of this and val are negative.)

 o  **not**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) not()

Returns a BigInteger whose value is (~this). (This method returns a negative value iff this number is non-negative.)

 o  **andNot**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) andNot([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns a BigInteger whose value is (this & ~val). This method, which is equivalent to and(val.not()), is provided as a convenience for masking operations. (This method returns a negative number iff this is negative and val is positive.)

 o  **compareTo**

public int compareTo([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns -1, 0 or 1 as this number is less than, equal to, or greater than val. This method is provided in preference to individual methods for each of the six boolean comparison operators (<, ==, >, >=, !=, <=). The suggested idiom for performing these comparisons is: (x.compareTo(y) 0), where is one of the six comparison operators.

 o  **equals**

public boolean equals([Object](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.Object.html#_top_) x)

Returns true iff x is a BigInteger whose value is equal to this number. This method is provided so that BigIntegers can be used as hash keys.

**Overrides:**

[equals](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.Object.html#equals(java.lang.Object)) in class [Object](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.Object.html#_top_)

 o  **min**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) min([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns the BigInteger whose value is the lesser of this and val. If the values are equal, either may be returned.

 o  **max**

public [BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) max([BigInteger](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.math.BigInteger.html#_top_) val)

Returns the BigInteger whose value is the greater of this and val. If the values are equal, either may be returned.

 o  **toString**

public [String](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.String.html#_top_) toString(int radix)

Returns the string representation of this number in the given radix. If the radix is outside the range from Character.MIN\_RADIX(2) to Character.MAX\_RADIX(36) inclusive, it will default to 10 (as is the case for Integer.toString). The digit-to-character mapping provided by Character.forDigit is used, and a minus sign is prepended if appropriate. (This representation is compatible with the (String, int) constructor.)

 o  **toString**

public [String](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.String.html#_top_) toString()

Returns the string representation of this number, radix 10. The digit-to-character mapping provided by Character.forDigit is used, and a minus sign is prepended if appropriate. (This representation is compatible with the (String) constructor, and allows for string concatenation with Java's + operator.)

**Overrides:**

[toString](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.Object.html#toString()) in class [Object](http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.Object.html#_top_)