Math Cheat Sheets

**Primes**

**int** x, y, c = 0;

**for**( x = 2; x < 1000; x++ )

{

**if**( x % 2 != 0 || x == 2 )

{

**for**( y = 2; y <= x / 2; y++ )

{

**if**( x % y == 0 )

{

**break**;

}

}

//count the number of primes

**if**( y > x / 2 )

{

System.*out*.println( x );

c++;

}

}

}

System.*out*.println( "\nTotal: " + c );

}

**Efficient**

**import** java.util.\*;

**public** **class** Sieve

{

**private** BitSet sieve;

**public** **static** **void** main(String[] args)

{

Sieve theSieve = **new** Sieve(10);

List<Integer> theListOfPrimes = Sieve.*sieve\_of\_eratosthenes*(100);

**for**(**int** i = 0; i < theListOfPrimes.size(); i++)

{

System.*out*.println(theListOfPrimes.get(i));

}

}

**public** Sieve(**int** size) {

sieve = **new** BitSet((size+1)/2);

}

**public** **boolean** is\_composite(**int** k)

{

**assert** k >= 3 && (k % 2) == 1;

**return** sieve.get((k-3)/2);

}

**public** **void** set\_composite(**int** k)

{

**assert** k >= 3 && (k % 2) == 1;

sieve.set((k-3)/2);

}

**public** **static** List<Integer> sieve\_of\_eratosthenes(**int** max)

{

Sieve sieve = **new** Sieve(max + 1); // +1 to include max itself

**for** (**int** i = 3; i\*i <= max; i += 2) {

**if** (sieve.is\_composite(i))

**continue**;

// We increment by 2\*i to skip even multiples of i

**for** (**int** multiple\_i = i\*i; multiple\_i <= max; multiple\_i += 2\*i)

sieve.set\_composite(multiple\_i);

}

List<Integer> primes = **new** ArrayList<Integer>();

primes.add(2);

**for** (**int** i = 3; i <= max; i += 2)

**if** (!sieve.is\_composite(i))

primes.add(i);

**return** primes;

}

}

**Sets**

// Create the set

Set set = new HashSet();

// Add elements to the set

set.add(*"a"*);

set.add(*"b"*);

set.add(*"c"*);

// Remove elements from the set

set.remove(*"c"*);

// Get number of elements in set

int size = set.size(); // 2

// Adding an element that already exists in the set has no effect

set.add(*"a"*);

size = set.size(); // 2

// Determining if an element is in the set

boolean b = set.contains(*"a"*); // true

b = set.contains(*"c"*); // false

// Iterating over the elements in the set

Iterator it = set.iterator();

while (it.hasNext()) {

// Get element

Object element = it.next();

}

// Create an array containing the elements in the set (in this case a String array)

*String*[] array = (*String*[])set.toArray(new *String*[set.size()]);

**Operating on Sets**

// Create the sets

Set set1 = new HashSet();

Set set2 = new HashSet();

// Add elements to the sets ...

// Copy all the elements from set2 to set1 (set1 += set2)

// set1 becomes the union of set1 and set2

set1.addAll(set2);

// Remove all the elements in set1 from set2 (set1 -= set2)

// set1 becomes the asymmetric difference of set1 and set2

set1.removeAll(set2);

// Get the intersection of set1 and set2

// set1 becomes the intersection of set1 and set2

set1.retainAll(set2);

// Remove all elements from a set

set1.clear();

**Sets remaining in order**

Set set = new LinkedHashSet();

// Add some elements

set.add(*"1"*);

set.add(*"2"*);

set.add(*"3"*);

set.add(*"2"*);

// List the elements

for (Iterator it=set.iterator(); it.hasNext(); ) {

Object o = it.next();

}

// [1, 2, 3]