APIs

`public class System.out/StdOut/Out`

	`Out(String name)`	*create output stream from* `name`
`void`	`print(String s)`	*print* `s`
`void`	`println(String s)`	*print* `s`*, followed by newline*
`void`	`println()`	*print a new line*
`void`	`printf(String f, ... )`	*formatted print*

Note: *Methods are static and constructor does not apply for* `System.out/StdOut`.

`public class Math`

`double`	`abs(double a)`	*absolute value of a*
`double`	`max(double a, double b)`	*maximum of a and b*
`double`	`min(double a, double b)`	*minimum of a and b*

Note 1: `abs()`, `max()`, *and* `min()` *are defined also for* `int`, `long`, *and* `float`.

`double`	`sin(double theta)`	*sine function*
`double`	`cos(double theta)`	*cosine function*
`double`	`tan(double theta)`	*tangent function*

Note 2: Angles are expressed in radians. Use `toDegrees()` *and* `toRadians()` *to convert.*
Note 3: Use `asin()`, `acos()`, *and* `atan()` *for inverse functions.*

`double`	`exp(double a)`	*exponential* (e^a)
`double`	`log(double a)`	*natural log* ($\log_e a$, *or* $\ln a$)
`double`	`pow(double a, double b)`	*raise a to the bth power* (a^b)
`long`	`round(double a)`	*round to the nearest integer*
`double`	`random()`	*random number in* $[0, 1)$
`double`	`sqrt(double a)`	*square root of a*
`double`	`E`	*value of e (constant)*
`double`	`PI`	*value of* π *(constant)*

`public class StdIn/In`

	`In(String name)`	*create input stream from* `name`
`boolean`	`isEmpty()`	`true` *if no more values, else* `false`
`int`	`readInt()`	*read a value of type* `int`
`double`	`readDouble()`	*read a value of type* `double`
`long`	`readLong()`	*read a value of type long*
`boolean`	`readBoolean()`	*read a value of type* `boolean`
`char`	`readChar()`	*read a value of type* `char`
`String`	`readString()`	*read a value of type* `String`
`String`	`readLine()`	*read the rest of the line*
`String`	`readAll()`	*read the rest of the text*

Note: *Methods are static and constructor does not apply for* `StdIn`.

`public class String`

	`String(String s)`	*create a string with the same value as* `s`
`int`	`length()`	*string length*
`char`	`charAt(int i)`	`i`*th character*
`String`	`substring(int i, int j)`	`i`*th through* (`j-1`)*st characters*
`boolean`	`contains(String sub)`	*does string contain* `sub` *as a substring?*
`boolean`	`startsWith(String pre)`	*does string start with* `pre`*?*
`boolean`	`endsWith(String post)`	*does string end with* `post`*?*
`int`	`indexOf(String p)`	*index of first occurrence of* `p`
`int`	`indexOf(String p, int i)`	*index of first occurrence of* `p` *after* `i`
`String`	`concat(String t)`	*this string with* `t` *appended*
`int`	`compareTo(String t)`	*string comparison*
`String`	`replaceAll(String a, String b)`	*result of changing* `a`*s to* `b`*s*
`String[]`	`split(String delim)`	*strings between occurrences of* `delim`
`boolean`	`equals(String t)`	*is this string's value the same as* `t`*'s?*

`public class StdDraw/Draw`

	`Draw()`	*create a new* `Draw` *object*
`void`	`line(double x0, double y0, double x1, double y1)`	
`void`	`point(double x, double y)`	
`void`	`text(double x, double y, String s)`	
`void`	`circle(double x, double y, double r)`	
`void`	`filledCircle(double x, double y, double r)`	
`void`	`square(double x, double y, double r)`	
`void`	`filledSquare(double x, double y, double r)`	
`void`	`polygon(double[] x, double[] y)`	
`void`	`filledPolygon(double[] x, double[] y)`	
`void`	`setXscale(double x0, double x1)`	*reset x range to* (x_0, x_1)
`void`	`setYscale(double y0, double y1)`	*reset y range to* (y_0, y_1)
`void`	`setPenRadius(double r)`	*set pen radius to* `r`
`void`	`setPenColor(Color c)`	*set pen color to* `c`
`void`	`setFont(Font f)`	*set text font to* `f`
`void`	`setCanvasSize(int w, int h)`	*set canvas to* `w`*-by-*`h` *window*
`void`	`clear(Color c)`	*clear the canvas; color it* `c`
`void`	`show(int dt)`	*show all; pause* `dt` *milliseconds*
`void`	`save(String filename)`	*save to a* `.jpg` *or* `.png` *file*

Note: *Methods are static and constructor does not apply for* `StdDraw`.

`public class StdAudio`

`void`	`play(String file)`	*play the given* .wav *file*
`void`	`play(double[] a)`	*play the given sound wave*
`void`	`play(double x)`	*play sample for 1/44100 second*
`void`	`save(String file, double[] a)`	*save to a* .wav *file*
`double[]`	`read(String file)`	*read from a* .wav *file*

public class StdRandom

`int`	`uniform(int N)`	*integer between* `0` *and* `N-1`
`double`	`uniform(double lo, double hi)`	*real between* `lo` *and* `hi`
`boolean`	`bernoulli(double p)`	*true with probability* `p`
`double`	`gaussian()`	*normal, mean* `0`*, standard deviation* `1`
`double`	`gaussian(double m, double s)`	*normal, mean* `m`*, standard deviation* `s`
`int`	`discrete(double[] a)`	`i` *with probability* `a[i]`
`void`	`shuffle(double[] a)`	*randomly shuffle the array* `a[]`

public class StdArrayIO

`double[]`	`readDouble1D()`	*read a one-dimensional array of* `double` *values*
`double[][]`	`readDouble2D()`	*read a two-dimensional array of* `double` *values*
`void`	`print(double[] a)`	*print a one-dimensional array of* `double` *values*
`void`	`print(double[][] a)`	*print a two-dimensional array of* `double` *values*

Note 1. 1D *format is an integer N followed by N values.*

Note 2. 2D *format is two integers M and N followed by M×N values in row-major order.*

Note 3. Methods for `int` *and* `boolean` *are also included.*

public class StdStats

`double`	`max(double[] a)`	*largest value*
`double`	`min(double[] a)`	*smallest value*
`double`	`mean(double[] a)`	*average*
`double`	`var(double[] a)`	*sample variance*
`double`	`stddev(double[] a)`	*sample standard deviation*
`double`	`median(double[] a)`	*median*
`void`	`plotPoints(double[] a)`	*plot points at* `(i, a[i])`
`void`	`plotLines(double[] a)`	*plot lines connecting* `(i, a[i])`
`void`	`plotBars(double[] a)`	*plot bars to points at* `(i, a[i])`

Note: overloaded implementations are included for all numeric types `public class Picture`

	`Picture(String name)`	*create a picture from a file*
	`Picture(int w, int h)`	*create a blank* `w`*-by-*`h` *picture*
`int`	`width()`	*return the width of the picture*
`int`	`height()`	*return the height of the picture*
`Color`	`get(int i, int j)`	*return the color of pixel (i, j)*
`void`	`set(int i, int j, Color c)`	*set the color of pixel (i, j) to* `c`
`void`	`show()`	*display the image in a window*
`void`	`save(String name)`	*save the image to a file*

`public class Stopwatch`

	`Stopwatch()`	*create a new stopwatch and start it running*
`double`	`elapsedTime()`	*return the elapsed time since creation, in seconds*

`public class Histogram`

	`Histogram(int N)`	*create a dynamic histogram for the N integer values in [0, N)*
`double`	`addDataPoint(int i)`	*add an occurrence of the value i*

`public class Turtle`

	`Turtle(double x0, double y0, double a0)`	*create a new turtle at (x_0, y_0) facing a_0 degrees counterclockwise from x-axis*
`void`	`turnLeft(double delta)`	*rotate* `delta` *degrees counterclockwise*
`void`	`goForward(double step)`	*move distance* `step`*, drawing a line*

public class Counter		
	`Counter(String id, int max)`	*create a counter, initialized to* `0`
`void`	`increment()`	*increment counter unless its value is* `max`
`int`	`value()`	*return the value of the counter*
`String`	`toString()`	*string representation*

public class Complex		
	`Complex(double real, double imag)`	
`Complex`	`plus(Complex b)`	*sum of this number and* `b`
`Complex`	`times(Complex b)`	*product of this number and* `b`
`double`	`abs()`	*magnitude*
`double`	`re()`	*real part*
`double`	`im()`	*imaginary part*
`String`	`toString()`	*string representation*

public class Vector		
	`Vector(double[] a)`	*create a vector with the given Cartesian coordinates*
`Vector`	`plus(Vector b)`	*sum of this vector and* `b`
`Vector`	`minus(Vector b)`	*difference of this vector and* `b`
`Vector`	`times(double t)`	*scalar product of this vector and* `t`
`double`	`dot(Vector b)`	*dot product of this vector and* `b`
`double`	`magnitude()`	*magnitude of this vector*
`Vector`	`direction()`	*unit vector with same direction as this vector*
`double`	`cartesian(int i)`	`i`*th cartesian coordinate of this vector*
`String`	`toString()`	*string representation*

`public class Stack<Item>`

	`Stack<Item>`	*create an empty stack*
`boolean`	`isEmpty()`	*is the stack empty?*
`void`	`push(Item item)`	*push an item onto the stack*
`Item`	`pop()`	*pop the stack*

`public class Queue<Item>`

	`Queue<Item>()`	*create an empty queue*
`boolean`	`isEmpty()`	*is the queue empty?*
`void`	`enqueue(Item item)`	*enqueue an item*
`Item`	`dequeue()`	*dequeue an item*
`int`	`length()`	*queue length*

`public class ST<Key extends Comparable<Key>, Value>`

	`ST()`	*create a symbol table*
`void`	`put(Key key, Value v)`	*put key-value pair into the table*
`Value`	`get(Key key)`	*return value paired with key,* `null` *if key not in table*
`boolean`	`contains(Key key)`	*is there a value paired with key?*

`public class SET<Key extends Comparable<Key>>`

	`SET()`	*create a set*
`boolean`	`isEmpty()`	*is the set empty?*
`void`	`add(Key key)`	*add* `key` *to the set*
`boolean`	`contains(Key key)`	*is* `key` *in the set?*

`public class Graph`

	`Graph()`	*create an empty graph*
	`Graph(In in, String delim)`	*read graph from input stream*
`void`	`addEdge(String v, String w)`	*add edge* `v-w`
`int`	`V()`	*number of vertices*
`int`	`E()`	*number of edges*
`Iterable<String>`	`vertices()`	*vertices in the graph*
`Iterable<String>`	`adjacentTo(String v)`	*neighbors of* `v`
`int`	`degree(String v)`	*number of neighbors of* `v`
`boolean`	`hasVertex(String v)`	*is* `v` *a vertex in the graph?*
`boolean`	`hasEdge(String v, String w)`	*is* `v-w` *an edge in the graph?*

`public class PathFinder`

	`PathFinder(Graph G, String s)`	*create an object that finds paths in* `G` *from* `s`
`int`	`distanceTo(String v)`	*length of shortest path from* `s` *to* `v` *in* `G`
`Iterable<String>`	`pathTo(String v)`	*shortest path from* `s` *to* `v` *in* `G`