Libraries and Data Types

This document describes[1] the Application Programming Interface (API) for the supporting libraries and data types used throughout the book *Computer Science: An Interdisciplinary Approach* by Robert Sedgewick and Kevin Wayne. The corresponding code is part of a package called `stdlib`.

BinaryIn	
`BinaryIn()`	constructs a binary input stream from standard input
`BinaryIn(InputStream in)`	constructs a binary input stream from an input stream
`BinaryIn(Socket socket)`	constructs a binary input stream from a socket
`BinaryIn(URL url)`	constructs a binary input stream from an URL
`BinaryIn(String name)`	constructs a binary input stream from a file or an URL with the given name
`boolean exists()`	returns `true` if this binary input stream exists, and `false` otherwise
`boolean isEmpty()`	returns `true` if this binary input stream is empty, and `false` otherwise
`boolean readBoolean()`	reads and returns the next bit of data from this binary input stream as a boolean
`char readChar()`	reads and returns the next 8 bits of data from this binary input stream as a char
`char readChar(int r)`	reads and returns the next `r` bits of data from this binary input stream as an `r`-bit char
`String readString()`	reads and returns the remaining bytes of data from this binary input stream as a string
`short readShort()`	reads and returns the next 16 bits of data from this binary input stream as a short
`int readInt()`	reads and returns the next 32 bits of data from this binary input stream as an int
`int readInt(int r)`	reads and returns the next `r` bits of data from this binary input stream as an `r`-bit int
`long readLong()`	reads and returns the next 64 bits of data from this binary input stream as a long
`double readDouble()`	reads and returns the next 64 bits of data from this binary input stream as a double
`float readFloat()`	reads and returns the next 32 bits of data from this binary input stream as a float
`byte readByte()`	reads and returns the next 8 bits of data from this binary input stream as a byte

BinaryOut	
`BinaryOut()`	constructs a binary output stream from standard output
`BinaryOut(OutputStream in)`	constructs a binary output stream from an output stream
`BinaryOut(String name)`	constructs a binary output stream from a file with the given name
`BinaryOut(Socket socket)`	constructs a binary output stream from a socket
`void flush()`	flushes this binary output stream, padding 0s if number of bits written is not a multiple of 8
`void close()`	flushes and closes this binary output stream
`void write(boolean x)`	writes the bit to this binary output stream
`void write(byte x)`	writes the 8-bit byte to this binary output stream
`void write(int x)`	writes the 32-bit int to this binary output stream
`void write(int x, int r)`	writes the `r`-bit int to this binary output stream
`void write(double x)`	writes the 64-bit double to this binary output stream
`void write(long x)`	writes the 64-bit long to this binary output stream
`void write(float x)`	writes the 32-bit float to this binary output stream
`void write(short x)`	writes the 16-bit short to this binary output stream
`void write(char x)`	writes the 8-bit char to this binary output stream
`void write(char x, int r)`	writes the `r`-bit char to this binary output stream
`void write(String x)`	writes the string of 8-bit characters to this binary output stream
`void write(String x, int r)`	writes the string of `r`-bit characters to this binary output stream

[1]A data type name in italics denotes an interface.

BinaryStdIn	
`static void close()`	closes standard input and releases any associated resources
`static boolean isEmpty()`	returns `true` if standard input is empty, and `false` otherwise
`static boolean readBoolean()`	reads and returns the next bit of data from standard input as a boolean
`static char readChar()`	reads and returns the next 8 bits of data from standard input as a char
`static char readChar(int r)`	reads and returns the next `r` bits of data from standard input as an `r`-bit char
`static String readString()`	reads and returns the remaining bytes of data from standard input as a string
`static short readShort()`	reads and returns the next 16 bits of data from standard input as a short
`static int readInt()`	reads and returns the next 32 bits of data from standard input as an int
`static int readInt(int r)`	reads and returns the next `r` bits of data from standard input as an `r`-bit int
`static long readLong()`	reads and returns the next 64 bits of data from standard input as a long
`static double readDouble()`	reads and returns the next 64 bits of data from standard input as a double
`static float readFloat()`	reads and returns the next 32 bits of data from standard input as a float
`static byte readByte()`	reads and returns the next 8 bits of data from standard input as a byte

BinaryStdOut	
`static void flush()`	flushes standard output, padding 0s if number of bits written is not a multiple of 8
`static void close()`	flushes and closes standard output
`static void write(boolean x)`	writes the bit to standard output
`static void write(byte x)`	writes the 8-bit byte to standard output
`static void write(int x)`	writes the 32-bit int to standard output
`static void write(int x, int r)`	writes the `r`-bit int to standard output
`static void write(double x)`	writes the 64-bit double to standard output
`static void write(long x)`	writes the 64-bit long to standard output
`static void write(float x)`	writes the 32-bit float to standard output
`static void write(short x)`	writes the 16-bit short to standard output
`static void write(char x)`	writes the 8-bit char to standard output
`static void write(char x, int r)`	writes the `r`-bit char to standard output
`static void write(String x)`	writes the string of 8-bit characters to standard output
`static void write(String x, int r)`	writes the string of `r`-bit characters to standard output

Draw	
`static Color BLACK`	represents black
`static Color BLUE`	represents blue
`static Color CYAN`	represents cyan
`static Color DARK_GRAY`	represents dark gray
`static Color GREEN`	represents green
`static Color LIGHT_GRAY`	represents light gray
`static Color MAGENTA`	represents magenta
`static Color ORANGE`	represents orange
`static Color PINK`	represents pink
`static Color RED`	represents red
`static Color WHITE`	represents white
`static Color YELLOW`	represents yellow
`static Color BOOK_BLUE`	represents the blue from Introduction to Programming in Java by Sedgewick et al
`static Color BOOK_LIGHT_BLUE`	represents the light blue from Introduction to Programming in Java by Sedgewick et al
`static Color BOOK_RED`	represents the red from Algorithms by Sedgewick et al
`static Color PRINCETON_ORANGE`	represents the orange used in Princeton's identity
`Draw(String name)`	constructs an empty canvas with the given name
`Draw()`	constructs an empty canvas
`void setLocationOnScreen(int x, int y)`	sets the upper-left corner of this canvas to `(x, y)`
`void setDefaultCloseOperation(int v)`	sets the default close operation of this canvas to `v`
`void setCanvasSize(int w, int h)`	sets the width and height of this canvas to `w` and `h` pixels
`void setXscale()`	sets the x-scale of this canvas to the default ($[0, 1]$)
`void setYscale()`	sets the y-scale of this canvas to the default ($[0, 1]$)
`void setXscale(double min, double max)`	sets the x-scale of this canvas to `[min, max]`
`void setYscale(double min, double max)`	sets the y-scale of this canvas to `[min, max]`
`void clear()`	clears this canvas to the default (white) color
`void clear(Color c)`	clears this canvas to the color `c`
`double getPenRadius()`	returns the current pen radius of this canvas
`void setPenRadius()`	sets the pen radius of this canvas to the default (0.002)
`void setPenRadius(double r)`	sets the pen radius of this canvas to `r`
`Color getPenColor()`	returns the current pen color of this canvas
`void setPenColor()`	sets the pen color of this canvas to the default (black) color
`void setPenColor(Color c)`	sets the pen color of this canvas to the color `c`
`void setPenColor(int r, int g, int b)`	sets the pen color of this canvas to the `(r, g, b)` color
`void xorOn()`	turns on xor mode on this canvas
`void xorOff()`	turns off xor mode on this canvas
`JLabel getLabel()`	returns the current label of this canvas for use in some other GUI
`void getFont()`	returns the current font for this canvas
`void setFont()`	sets the font for this canvas to the default (sans serif, 16 point) font
`void setFont(Font f)`	sets the font for this canvas to the font `f`
`void drawLine(double x0, double y0, double x1, double y1)`	draws a line on this canvas from `(x0, y0)` to `(x1, y1)`
`void draw(double x, double y)`	draws one pixel on this canvas at `(x, y)`
`void point(double x, double y)`	draws a point on this canvas at `(x, y)`
`void circle(double x, double y, double r)`	draws a circle on this canvas of radius `r`, centered at `(x, y)`
`void filledCircle(double x, double y, double r)`	draws a filled circle on this canvas of radius `r`, centered at `(x, y)`
`void ellipse(double x, double y, double maj, double min)`	draws an ellipse on this canvas with semimajor and semiminor axes `maj` and `min`, centered at `(x, y)`

Draw	
`void filledEllipse(double x, double y, double maj, double min)`	draws a filled ellipse on this canvas with semimajor and semiminor axes `maj` and `min`, centered at `(x, y)`
`void arc(double x, double y, double r, double a1, double a2)`	draws an arc on this canvas with radius `r`, centered at `(x, y)`, from `a1` to `a2` (in degrees)
`void square(double x, double y, double l)`	draws a square on this canvas of side length `2l`, centered at `(x, y)`
`void filledSquare(double x, double y, double l)`	draws a filled square on this canvas of side length `2l`, centered at `(x, y)`
`void rectangle(double x, double y, double w, double h)`	draws a rectangle on this canvas of width `2w` and height `2h`, centered at `(x, y)`
`void filledRectangle(double x, double y, double w, double h)`	draws a filled rectangle on this canvas of width `2w` and height `2h`, centered at `(x, y)`
`void polygon(double[] x, double[] y)`	draws a polygon on this canvas with vertices whose coordinates are given by `x` and `y`
`void filledPolygon(double[] x, double[] y)`	draws a filled polygon on this canvas with vertices whose coordinates are given by `x` and `y`
`void picture(double x, double y, String name)`	draws the image with the given name on this canvas, centered at `(x, y)`
`void picture(double x, double y, String name, double a)`	draws the image with the given name on this canvas, centered at `(x, y)`, rotated by `a` (in degrees)
`void picture(double x, double y, String name, double w, double h)`	draws the image with the given name on this canvas, centered at `(x, y)`, rescaled to a `w x h` bounding box
`void picture(double x, double y, String name, double w, double h, double a)`	draws the image with the given name on this canvas, centered at `(x, y)`, rotated by `a` (in degrees), rescaled to a `w x h` bounding box
`void text(double x, double y, String s)`	draws the text `s` on this canvas, centered at `(x, y)`
`void text(double x, double y, String s, double a)`	draws the text `s` on this canvas, centered at `(x, y)`, rotated by `a` (in degrees)
`void textLeft(double x, double y, String s)`	draws the text `s` on this canvas, left-aligned at `(x, y)`
`void textRight(double x, double y, String s)`	draws the text `s` on this canvas, right-aligned at `(x, y)`
`void pause(int t)`	pauses this canvas for `t` milliseconds
`void show()`	copies offscreen buffer to onscreen buffer on this canvas
`void enableDoubleBuffering()`	enables double buffering on this canvas
`void disableDoubleBuffering()`	disables double buffering on this canvas
`void save(String name)`	saves the drawing on this canvas to a file with the given name
`void addListener(DrawListener l)`	adds a listener to this canvas to listen to keyboard and mouse events
`boolean isMousePressed()`	returns `true` if the mouse is being pressed on this canvas, and `false` otherwise
`double mouseX()`	returns the x-coordinate of the mouse pointer on this canvas
`double mouseY()`	returns the y-coordinate of the mouse pointer on this canvas
`boolean hasNextKeyTyped()`	returns `true` if the user has typed a key on this canvas, and `false` otherwise
`char nextKeyTyped()`	returns the next key typed by the user on this canvas
`boolean isKeyPressed(int c)`	returns `true` if the keycode `c` is being pressed on this canvas, and `false` otherwise

DrawListener	
`void mousePressed(double x, double y)`	invoked when the mouse has been pressed, with `(x, y)` denoting the mouse coordinates
`void mouseDragged(double x, double y)`	invoked when the mouse has been dragged, with `(x, y)` denoting the mouse coordinates
`void mouseReleased(double x, double y)`	invoked when the mouse has been released, with `(x, y)` denoting the mouse coordinates
`void mouseClicked(double x, double y)`	invoked when the mouse has been clicked, with `(x, y)` denoting the mouse coordinates
`void keyTyped(char c)`	invoked when a key has been typed, with `c` denoting the character typed
`void keyPressed(int c)`	invoked when a key has been pressed, with `c` denoting the key combination pressed
`void keyReleased(int c)`	invoked when a key has been released, with `c` denoting the key combination released

GrayscalePicture	
`GrayscalePicture(int w, int h)`	constructs a grayscale picture of width `w` and height `h`
`GrayscalePicture(GrayscalePicture pic)`	constructs a grayscale picture that is a deep copy of `pic`
`GrayscalePicture(String name)`	constructs a grayscale picture from an image with the given name
`static Color toGray(Color c)`	retuns a grayscale version of the color `c`
`JLabel getLabel()`	returns the current label of this picture for use in some other GUI
`void setOriginUpperLeft()`	sets the origin to be the upper left pixel (default)
`void setOriginLowerLeft()`	sets the origin to be the lower left pixel
`void show()`	displays this picture on the screen
`int height()`	returns the height of this picture
`int width()`	returns the width of this picture
`Color get(int col, int row)`	returns the grayscale value of pixel `(col, row)` as a `Color` object
`int getGrayscale(int col, int row)`	returns the grayscale value of pixel `(col, row)` as an int
`void set(int col, int row, Color c)`	sets the color of the pixel `(col, row)` to the given value (a `Color` object)
`void setGrayscale(int col, int row, int gray)`	sets the color of the pixel `(col, row)` to the given value (an int from $[0, 255]$)
`boolean equals(Object other)`	returns `true` if this picture is equal to `other`, and `false` otherwise
`String toString()`	returns a string representation of this picture
`void save(String name)`	saves this picture to a file with the given name
`void actionPerformed(ActionEvent e)`	opens a save dialog box when the user selects “Save As” from the menu

In	
`In()`	constructs an input stream from standard input
`In(Socket socket)`	constructs an input stream from a socket
`In(URL url)`	constructs an input stream from an URL
`In(String name)`	constructs an input stream from a file with the given name
`In(Scanner scanner)`	constructs an input stream from a scanner
`boolean exists()`	returns `true` if this input stream exists, and `false` otherwise
`boolean isEmpty()`	returns `true` if this input stream is empty, and `false` otherwise
`boolean hasNextLine()`	returns `true` if this input stream has a next line, and `false` otherwise
`boolean hasNextChar()`	returns `true` if this input stream has a next char, and `false` otherwise
`String readLine()`	reads and returns the next line from this input stream
`char readChar()`	reads and returns the next char from this input stream
`String readAll()`	reads and returns the remainder of this input stream as a string
`String readString()`	reads and returns the next token from this input stream as a string
`int readInt()`	reads and returns the next int from this input stream
`double readDouble()`	reads and returns the next double from this input stream
`float readFloat()`	reads and returns the next float from this input stream
`long readLong()`	reads and returns the next long from this input stream
`short readShort()`	reads and returns the next short from this input stream
`byte readByte()`	reads and returns the next byte from this input stream
`boolean readBoolean()`	reads and returns the next boolean from this input stream
`String[] readAllStrings()`	reads and returns all the remaining tokens from this input stream as an array of strings
`String[] readAllLines()`	reads and returns all the remaining lines from this input stream as an array of strings
`int[] readAllInts()`	reads and returns all the remaining tokens from this input stream as an array of ints
`long[] readAllLongs()`	reads and returns all the remaining tokens from this input stream as an array of longs
`double[] readAllDoubles()`	reads and returns all the remaining tokens from this input stream as an array of doubles
`void close()`	closes this input stream

Out	
`Out()`	constructs an output stream from standard output
`Out(Socket socket)`	constructs an output stream from a socket
`Out(String name)`	constructs an output stream from a file with the given name
`void close()`	closes this output stream
`void println()`	prints a newline to this output stream
`void println(Object x)`	prints an object and a newline to this output stream
`void println(boolean x)`	prints a boolean and a newline to this output stream
`void println(char x)`	prints a char and a newline to this output stream
`void println(double x)`	prints a double and a newline to this output stream
`void println(float x)`	prints a float and a newline to this output stream
`void println(int x)`	prints an int and a newline to this output stream
`void println(long x)`	prints a long and a newline to this output stream
`void println(byte x)`	prints a byte and a newline to this output stream
`void print()`	flushes this output stream
`void print(Object x)`	prints an object to this output stream
`void print(boolean x)`	prints a boolean to this output stream
`void print(char x)`	prints a char to this output stream
`void print(double x)`	prints a double to this output stream
`void print(float x)`	prints a float to this output stream
`void print(int x)`	prints an int to this output stream
`void print(long x)`	prints a long to this output stream
`void print(byte x)`	prints a byte to this output stream
`void printf(String fmt, Object... args)`	prints `args` to this output stream using format string `fmt`
`void printf(Locale loc, String fmt, Object... args)`	prints `args` to this output stream using locale `loc` and format string `fmt`

Picture	
`Picture(int w, int h)`	constructs a picture of width `w` and height `h`
`Picture(Picture pic)`	constructs a picture that is a deep copy of `pic`
`Picture(String name)`	constructs a picture from an image with the given name
`JLabel getLabel()`	returns the current label of this picture for use in some other GUI
`void setOriginUpperLeft()`	sets the origin to be the upper left pixel (default)
`void setOriginLowerLeft()`	sets the origin to be the lower left pixel
`void show()`	displays this picture on the screen
`int height()`	returns the height of this picture
`int width()`	returns the width of this picture
`Color get(int col, int row)`	returns the color of pixel `(col, row)` as a `Color` object
`int getRGB(int col, int row)`	returns the color of pixel `(col, row)` as an int
`void set(int col, int row, Color c)`	sets the color of the pixel `(col, row)` to the given value (a `Color` object)
`void setRGB(int col, int row, int rgb)`	sets the color of the pixel `(col, row)` to the given value
`boolean equals(Object other)`	returns `true` if this picture is equal to `other`, and `false` otherwise
`String toString()`	returns a string representation of this picture
`void save(String name)`	saves this picture to a file with the given name
`void actionPerformed(ActionEvent e)`	opens a save dialog box when the user selects “Save As” from the menu

StdArrayIO	
`static double[] readDouble1D()`	reads an integer n from standard input, and then reads n doubles also from standard input and returns them as a 1D array of size n
`static void print(double[] a)`	prints the size and elements of the 1D array `a` to standard output
`static double[][] readDouble2D()`	reads integers m and n from standard input, and then reads mn doubles also from standard input and returns them as a 2D array of size $m \times n$
`static void print(double[][] a)`	prints the size and elements of the 2D array `a` to standard output
`static int[] readInt1D()`	reads an integer n from standard input, and then reads n ints also from standard input and returns them as a 1D array of size n
`static void print(int[] a)`	prints the size and elements of the 1D array `a` to standard output
`static int[][] readInt2D()`	reads integers m and n from standard input, and then reads mn ints also from standard input and returns them as a 2D array of size $m \times n$
`static void print(int[][] a)`	prints the size and elements of the 2D array `a` to standard output
`static boolean[] readBoolean1D()`	reads an integer n from standard input, and then reads n booleans also from standard input and returns them as a 1D array of size n
`static void print(boolean[] a)`	prints the size and elements of the 1D array `a` to standard output
`static boolean[][] readBoolean2D()`	reads integers m and n from standard input, and then reads mn booleans also from standard input and returns them as a 2D array of size $m \times n$
`static void print(boolean[][] a)`	prints the size and elements of the 2D array `a` to standard output

StdAudio	
`static void close()`	closes standard audio
`static void play(double sample)`	plays one sample (between -1.0 and +1.0) using standard audio
`static void play(double[] samples)`	plays an array of samples (between -1.0 and +1.0) using standard audio
`static double[] read(String name)`	reads and returns samples from a file (`.au` or `.wav` format) with the given name
`static void save(String name, double[] samples)`	saves the samples as a file (`.au` or `.wav` format) with the given name
`static void play(String name)`	plays a file (`.au`, `.mid`, or `.au` format) with the given name in the background using standard audio
`static void loop(String name)`	loops a file (`.au`, `.mid`, or `.au` format) with the given name in the background using standard audio

StdDraw	
`static Color BLACK`	represents black
`static Color BLUE`	represents blue
`static Color CYAN`	represents cyan
`static Color DARK_GRAY`	represents dark gray
`static Color GREEN`	represents green
`static Color LIGHT_GRAY`	represents light gray
`static Color MAGENTA`	represents magenta
`static Color ORANGE`	represents orange
`static Color PINK`	represents pink
`static Color RED`	represents red
`static Color WHITE`	represents white
`static Color YELLOW`	represents yellow
`static Color BOOK_BLUE`	represents the blue from Introduction to Programming in Java by Sedgewick et al
`static Color BOOK_LIGHT_BLUE`	represents the light blue from Introduction to Programming in Java by Sedgewick et al
`static Color BOOK_RED`	represents the red from Algorithms by Sedgewick et al
`static Color PRINCETON_ORANGE`	represents the orange used in Princeton's identity
`static void setCanvasSize()`	sets the width and height of the canvas to 512 pixels
`static void setCanvasSize(int w, int h)`	sets the width and height of the canvas to `w` and `h` pixels
`static void setXscale()`	sets the x-scale of the canvas to the default ($[0, 1]$)
`static void setYscale()`	sets the y-scale of the canvas to the default ($[0, 1]$)
`static void setScale()`	sets the x- and y-scale of the canvas to the default ($[0, 1]$)
`static void setXscale(double min, double max)`	sets the x-scale of the canvas to `[min, max]`
`static void setYscale(double min, double max)`	sets the y-scale of the canvas to `[min, max]`
`static void setScale(double min, double max)`	sets the x- and y-scale of the canvas to `[min, max]`
`static void clear()`	clears the canvas to the default (white) color
`static void clear(Color c)`	clears the canvas to the color `c`
`static double getPenRadius()`	returns the current pen radius of the canvas
`static void setPenRadius()`	sets the pen radius of the canvas to the default (0.002)
`static void setPenRadius(double r)`	sets the pen radius of the canvas to `r`
`static Color getPenColor()`	returns the current pen color of the canvas
`static void setPenColor()`	sets the pen color of the canvas to the default (black) color
`static void setPenColor(Color c)`	sets the pen color of the canvas to the color `c`
`static void setPenColor(int r, int g, int b)`	sets the pen color of the canvas to the `(r, g, b)` color
`static void getFont()`	returns the current font for the canvas
`static void setFont()`	sets the font for the canvas to the default (sans serif, 16 point) font
`static void setFont(Font f)`	sets the font for the canvas to the font `f`
`static void drawLine(double x0, double y0, double x1, double y1)`	draws a line on the canvas from `(x0, y0)` to `(x1, y1)`
`static void draw(double x, double y)`	draws one pixel on the canvas at `(x, y)`
`static void point(double x, double y)`	draws a point on the canvas at `(x, y)`
`static void circle(double x, double y, double r)`	draws a circle on the canvas of radius `r`, centered at `(x, y)`
`static void filledCircle(double x, double y, double r)`	draws a filled circle on the canvas of radius `r`, centered at `(x, y)`
`static void ellipse(double x, double y, double maj, double min)`	draws an ellipse on the canvas with semimajor and semiminor axes `maj` and `min`, centered at `(x, y)`
`static void filledEllipse(double x, double y, double maj, double min)`	draws a filled ellipse on the canvas with semimajor and semiminor axes `maj` and `min`, centered at `(x, y)`
`static void arc(double x, double y, double r, double a1, double a2)`	draws an arc on the canvas with radius `r`, centered at `(x, y)`, from `a1` to `a2` (in degrees)

StdDraw	
`static void square(double x, double y, double l)`	draws a square on the canvas of side length `2l`, centered at `(x, y)`
`static void filledSquare(double x, double y, double l)`	draws a filled square on the canvas of side length `2l`, centered at `(x, y)`
`static void rectangle(double x, double y, double w, double h)`	draws a rectangle on the canvas of width `2w` and height `2h`, centered at `(x, y)`
`static void filledRectangle(double x, double y, double w, double h)`	draws a filled rectangle on the canvas of width `2w` and height `2h`, centered at `(x, y)`
`static void polygon(double[] x, double[] y)`	draws a polygon on the canvas with vertices whose coordinates are given by `x` and `y`
`static void filledPolygon(double[] x, double[] y)`	draws a filled polygon on the canvas with vertices whose coordinates are given by `x` and `y`
`static void picture(double x, double y, String name)`	draws the image with the given name on the canvas, centered at `(x, y)`
`static void picture(double x, double y, String name, double a)`	draws the image with the given name on the canvas, centered at `(x, y)`, rotated by `a` (in degrees)
`static void picture(double x, double y, String name, double w, double h)`	draws the image with the given name on the canvas, centered at `(x, y)`, rescaled to a `w x h` bounding box
`static void picture(double x, double y, String name, double w, double h, double a)`	draws the image with the given name on the canvas, centered at `(x, y)`, rotated by `a` (in degrees), rescaled to a `w x h` bounding box
`static void text(double x, double y, String s)`	draws the text `s` on the canvas, centered at `(x, y)`
`static void text(double x, double y, String s, double a)`	draws the text `s` on the canvas, centered at `(x, y)`, rotated by `a` (in degrees)
`static void textLeft(double x, double y, String s)`	draws the text `s` on the canvas, left-aligned at `(x, y)`
`static void textRight(double x, double y, String s)`	draws the text `s` on the canvas, right-aligned at `(x, y)`
`static void pause(int t)`	pauses the canvas for `t` milliseconds
`static void show()`	copies offscreen buffer to onscreen buffer on the canvas
`static void enableDoubleBuffering()`	enables double buffering on the canvas
`static void disableDoubleBuffering()`	disables double buffering on the canvas
`static void save(String name)`	saves the drawing on the canvas to a file with the given name
`static boolean isMousePressed()`	returns `true` if the mouse is being pressed on the canvas, and `false` otherwise
`static double mouseX()`	returns the x-coordinate of the mouse pointer on the canvas
`static double mouseY()`	returns the y-coordinate of the mouse pointer on the canvas
`static boolean hasNextKeyTyped()`	returns `true` if the user has typed a key on the canvas, and `false` otherwise
`static char nextKeyTyped()`	returns the next key typed by the user on the canvas
`static boolean isKeyPressed(int c)`	returns `true` if the keycode `c` is being pressed on the canvas, and `false` otherwise

StdIn

`static boolean isEmpty()`	returns `true` if standard input is empty, and `false` otherwise
`static boolean hasNextLine()`	returns `true` if standard input has a next line, and `false` otherwise
`static boolean hasNextChar()`	returns `true` if standard input has a next char, and `false` otherwise
`static String readLine()`	reads and returns the next line from standard input
`char readChar()`	reads and returns the next char from this standard input
`static String readAll()`	reads and returns the remainder of standard input as a string
`static String readString()`	reads and returns the next token from standard input as a string
`static int readInt()`	reads and returns the next int from standard input
`static double readDouble()`	reads and returns the next double from standard input
`static float readFloat()`	reads and returns the next float from standard input
`static long readLong()`	reads and returns the next long from standard input
`static short readShort()`	reads and returns the next short from standard input
`static byte readByte()`	reads and returns the next byte from standard input
`static boolean readBoolean()`	reads and returns the next boolean from standard input
`static String[] readAllStrings()`	reads and returns all the remaining tokens from standard input as an array of strings
`static String[] readAllLines()`	reads and returns all the remaining lines from standard input as an array of strings
`static int[] readAllInts()`	reads and returns all the remaining tokens from standard input as an array of ints
`static long[] readAllLongs()`	reads and returns all the remaining tokens from standard input as an array of longs
`static double[] readAllDoubles()`	reads and returns all the remaining tokens from standard input as an array of doubles
`static void resync()`	reinitializes the scanner underlying standard input

StdOut

`static void println()`	prints a newline to standard output
`static void println(Object x)`	prints an object and a newline to standard output
`static void println(boolean x)`	prints a boolean and a newline to standard output
`static void println(char x)`	prints a char and a newline to standard output
`static void println(double x)`	prints a double and a newline to standard output
`static void println(float x)`	prints a float and a newline to standard output
`static void println(int x)`	prints an int and a newline to standard output
`static void println(long x)`	prints a long and a newline to standard output
`static void println(short x)`	prints a short and a newline to standard output
`static void println(byte x)`	prints a byte and a newline to standard output
`static void print()`	flushes standard output
`static void print(Object x)`	prints an object to standard output
`static void print(boolean x)`	prints a boolean to standard output
`static void print(char x)`	prints a char to standard output
`static void print(double x)`	prints a double to standard output
`static void print(float x)`	prints a float to standard output
`static void print(int x)`	prints an int to standard output
`static void print(long x)`	prints a long to standard output
`static void print(short x)`	prints a short to standard output
`static void print(byte x)`	prints a byte to standard output
`static void printf(String fmt, Object... args)`	prints `args` to standard output using format string `fmt`
`static void printf(Locale loc, String fmt, Object... args)`	prints `args` to standard output using locale `loc` and format string `fmt`
`static void resync()`	reinitializes the writer underlying standard output

StdRandom

`static seed(long s)`	sets the seed of the random number generator to `s`
`static long getSeed()`	returns the seed of the random number generator
`static double uniform()`	returns a double chosen uniformly at random from the interval `[0, 1)`
`static int uniform(int n)`	returns an integer chosen uniformly at random from the interval `[0, n)`
`static long uniform(long n)`	returns a long chosen uniformly at random from the interval `[0, n)`
`static int uniform(int a, int b)`	returns an integer chosen uniformly at random from the interval `[a, b)`
`static double uniform(double a, double b)`	returns a double chosen uniformly at random from the interval `[a, b)`
`static boolean bernoulli(double p)`	returns `true` with probability `p` and `false` with probability `1 - p`
`static boolean bernoulli()`	returns `true` with probability 0.5 and `false` with probability 0.5
`static double gaussian()`	returns a double from the standard Gaussian distribution
`static double gaussian(double mu, double sigma)`	returns a double from a Gaussian distribution with mean `mu` and standard deviation `sigma`
`static int geometric(double p)`	returns an integer from a geometric distribution with success probability `p`
`static int poisson(double lambda)`	returns an integer from a Poisson distribution with mean `lambda`
`static double pareto()`	returns a double from the standard Pareto distribution
`static double pareto(double alpha)`	returns a double from a Pareto distribution with shape parameter `alpha`
`static double cauchy()`	returns a double from the Cauchy distribution
`static int discrete(double[] probabilities)`	returns an integer `i` with probability `probabilities[i]`
`static int discrete(int[] frequencies)`	returns an integer `i` with probability `frequencies[i]`
`static double exp(double lambda)`	returns a double from an exponential distribution with rate `lambda`
`static void shuffle(Object[] a)`	shuffles the array `a`
`static void shuffle(double[] a)`	shuffles the array `a`
`static void shuffle(int[] a)`	shuffles the array `a`
`static void shuffle(Object[] a, int lo, int hi)`	shuffles the subarray `a[lo, ..., hi)`
`static void shuffle(double[] a, int lo, int hi)`	shuffles the subarray `a[lo, ..., hi)`
`static void shuffle(int[] a, int lo, int hi)`	shuffles the subarray `a[lo, ..., hi)`
`static int[] permutation(int n)`	returns a uniformly random permutation of `n` elements
`static int[] permutation(int n, int k)`	returns a uniformly random permutation of `k` (out of `n`) elements

StdStats

`static double max(double[] a)`	returns the maximum value in the array `a`
`static double max(double[] a, int lo, int hi)`	returns the maximum value in the subarray `a[lo, ..., hi)`
`static int max(int[] a)`	returns the maximum value in the array `a`
`static double min(double[] a)`	returns the minimum value in the array `a`
`static double min(double[] a, int lo, int hi)`	returns the minimum value in the subarray `a[lo, ..., hi)`
`static int min(int[] a)`	returns the minimum value in the array `a`
`static double mean(double[] a)`	returns the average value in the array `a`
`static double mean(double[] a, int lo, int hi)`	returns the average value in the subarray `a[lo, ..., hi)`
`static double mean(int[] a)`	returns the average value in the array `a`
`static double var(double[] a)`	returns the sample variance in the array `a`
`static double var(double[] a, int lo, int hi)`	returns the sample variance in the subarray `a[lo, ..., hi)`
`static double var(int[] a)`	returns the sample variance in the array `a`
`static double varp(double[] a)`	returns the population variance in the array `a`
`static double varp(double[] a, int lo, int hi)`	returns the population variance in the subarray `a[lo, ..., hi)`
`static double stddev(double[] a)`	returns the sample standard deviation in the array `a`
`static double stddev(double[] a, int lo, int hi)`	returns the sample standard deviation in the subarray `a[lo, ..., hi)`
`static double stddev(int[] a)`	returns the sample standard deviation in the array `a`
`static double stddevp(double[] a)`	returns the population standard deviation in the array `a`
`static double stddevp(double[] a, int lo, int hi)`	returns the population standard deviation in the subarray `a[lo, ..., hi)`
`static double sum(double[] a)`	returns the sum of all values in the array `a`
`static double sum(double[] a, int lo, int hi)`	returns the sum of all values in the subarray `a[lo, ..., hi)`
`static int sum(int[] a)`	returns the sum of all values in the array `a`
`static void plotPoints(double[] a)`	plots the values in the array `a` as points
`static void plotLines(double[] a)`	plots the values in the array `a` as line end-points
`static void plotBars(double[] a)`	plots the values in the array `a` as bars

Stopwatch

`Stopwatch()`	creates a new stopwatch
`double elapsedTime()`	returns the elapsed time (in seconds) since the stopwatch was created