


INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

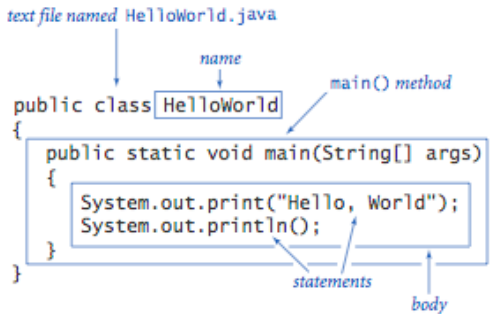
Robert Sedgewick • Kevin Wayne

INTRO TO PROGRAMMING
1. Elements of Programming
2. Functions
3. OOP
4. Data Structures
INTRO TO CS
0. Prologue
5. A Computing Machine
6. Building a Computer
7. Theory of Computation
8. Systems
9. Scientific Computation
ALGORITHMS, 4TH EDITION

WEB RESOURCES
FAQ
Data
Code
Errata
Appendices
Lecture Slides
Programming Assignments

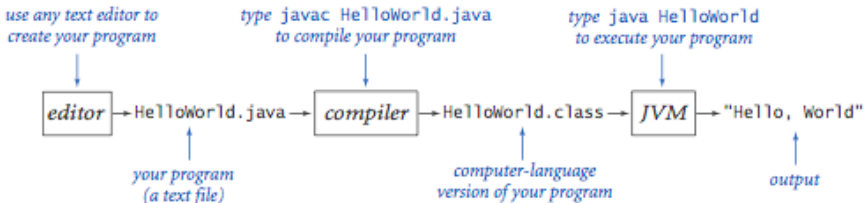
APPENDIX D: JAVA PROGRAMMING CHEATSHEET

This appendix summarizes the most commonly-used Java language features in the textbook. Here are the [APIs](#) of the most common libraries.

Hello, World.



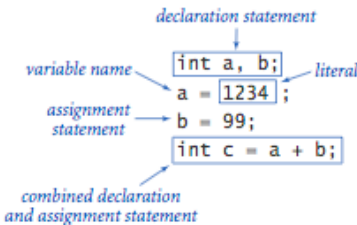
Editing, compiling, and executing.



Built-in data types.

type	set of values	common operators	sample literal values
int	integers	+ - * / %	99 -12 2147483647
double	floating-point numbers	+ - * /	3.14 -2.5 6.022e23
boolean	boolean values	&& !	true false
char	characters		'A' '1' '%' '\n'
String	sequences of characters	+	"AB" "Hello" "2.5"

Declaration and assignment statements.



Integers.

<i>values</i>	integers between -2^{31} and $+2^{31}-1$				
<i>typical literals</i>	1234 99 -99 0 1000000				
<i>operations</i>	add	subtract	multiply	divide	remainder
<i>operators</i>	+	-	*	/	%

<i>expression</i>	<i>value</i>	<i>comment</i>
5 + 3	8	
5 - 3	2	
5 * 3	15	
5 / 3	1	no fractional part
5 % 3	2	remainder
1 / 0		run-time error
3 * 5 - 2	13	* has precedence
3 + 5 / 2	5	/ has precedence
3 - 5 - 2	-4	left associative
(3 - 5) - 2	-4	better style
3 - (5 - 2)	0	unambiguous

Floating-point numbers.

<i>values</i>	real numbers (specified by IEEE 754 standard)				
<i>typical literals</i>	3.14159	6.022e23	-3.0	2.0	1.4142135623730951
<i>operations</i>	add	subtract	multiply	divide	
<i>operators</i>	+	-	*	/	

<i>expression</i>	<i>value</i>
3.141 + .03	3.171
3.141 - .03	3.111
6.02e23 / 2.0	3.01e23
5.0 / 3.0	1.6666666666666667
10.0 % 3.141	0.577
1.0 / 0.0	Infinity
Math.sqrt(2.0)	1.4142135623730951
Math.sqrt(-1.0)	NaN

Booleans.

<i>values</i>	true or false		
<i>literals</i>	true false		
<i>operations</i>	and	or	not
<i>operators</i>	&&		!

a	!a	a	b	a && b	a b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

Comparison operators.

<i>op</i>	<i>meaning</i>	<i>true</i>	<i>false</i>
<code>==</code>	<i>equal</i>	<code>2 == 2</code>	<code>2 == 3</code>
<code>!=</code>	<i>not equal</i>	<code>3 != 2</code>	<code>2 != 2</code>
<code><</code>	<i>less than</i>	<code>2 < 13</code>	<code>2 < 2</code>
<code><=</code>	<i>less than or equal</i>	<code>2 <= 2</code>	<code>3 <= 2</code>
<code>></code>	<i>greater than</i>	<code>13 > 2</code>	<code>2 > 13</code>
<code>>=</code>	<i>greater than or equal</i>	<code>3 >= 2</code>	<code>2 >= 3</code>

non-negative discriminant? `(b*b - 4.0*a*c) >= 0.0`
beginning of a century? `(year % 100) == 0`
legal month? `(month >= 1) && (month <= 12)`

Parsing command-line arguments.

`int Integer.parseInt(String s)` *convert s to an int value*
`double Double.parseDouble(String s)` *convert s to a double value*
`long Long.parseLong(String s)` *convert s to a long value*

Math library.

```
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)*

<i>expression</i>	<i>library</i>	<i>type</i>	<i>value</i>
<code>Integer.parseInt("123")</code>	Integer	int	123
<code>Math.sqrt(5.0*5.0 - 4.0*4.0)</code>	Math	double	3.0
<code>Math.random()</code>	Math	double	<i>random in [0, 1)</i>
<code>Math.round(3.14159)</code>	Math	long	3

The full java.lang.Math API.

Type conversion.

<i>expression</i>	<i>expression type</i>	<i>expression value</i>
"1234" + 99	String	"123499"
Integer.parseInt("123")	int	123
(int) 2.71828	int	2
Math.round(2.71828)	long	3
(int) Math.round(2.71828)	int	3
(int) Math.round(3.14159)	int	3
11 * 0.3	double	3.3
(int) 11 * 0.3	double	3.3
11 * (int) 0.3	int	0
(int) (11 * 0.3)	int	3

If and if-else statements.

<i>absolute value</i>	if (x < 0) x = -x;
<i>put x and y into sorted order</i>	<pre> if (x > y) { int t = x; y = x; x = t; } </pre>
<i>maximum of x and y</i>	<pre> if (x > y) max = x; else max = y; </pre>
<i>error check for division operation</i>	<pre> if (den == 0) System.out.println("Division by zero"); else System.out.println("Quotient = " + num/den); </pre>
<i>error check for quadratic formula</i>	<pre> double discriminant = b*b - 4.0*c; if (discriminant < 0.0) { System.out.println("No real roots"); } else { System.out.println((-b + Math.sqrt(discriminant))/2.0); System.out.println((-b - Math.sqrt(discriminant))/2.0); } </pre>

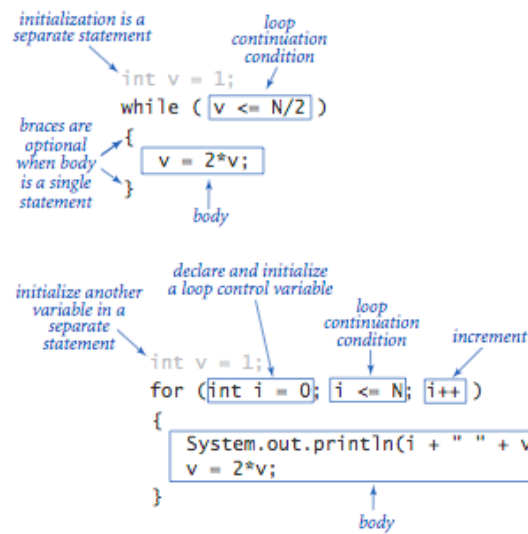
Nested if-else statement.

```

if      (income < 0) rate = 0.0;
else if (income < 47450) rate = .22;
else if (income < 114650) rate = .25;
else if (income < 174700) rate = .28;
else if (income < 311950) rate = .33;
else          rate = .35;

```

While and for loops.



<i>print largest power of two less than or equal to N</i>	<pre> int v = 1; while (v <= N/2) v = 2*v; System.out.println(v); </pre>
<i>compute a finite sum (1 + 2 + ... + N)</i>	<pre> int sum = 0; for (int i = 1; i <= N; i++) sum += i; System.out.println(sum); </pre>
<i>compute a finite product (N! = 1 × 2 × ... × N)</i>	<pre> int product = 1; for (int i = 1; i <= N; i++) product *= i; System.out.println(product); </pre>
<i>print a table of function values</i>	<pre> for (int i = 0; i <= N; i++) System.out.println(i + " " + 2*Math.PI*i/N); </pre>
<i>print the ruler function (see Program 1.2.1)</i>	<pre> String ruler = " "; for (int i = 1; i <= N; i++) ruler = ruler + i + ruler; System.out.println(ruler); </pre>

Break statement.

```

int i;
for (i = 2; i <= N/i; i++)
    if (N % i == 0) break;
if (i > N/i) System.out.println(N + " is prime");

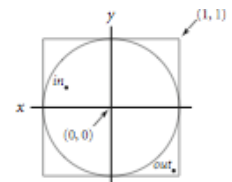
```

Do-while loop.

```

do
{
    x = 2.0*Math.random() - 1.0;
    y = 2.0*Math.random() - 1.0;
} while (Math.sqrt(x*x + y*y) > 1.0);

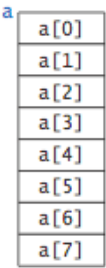
```



Switch statement.

```
switch (day)
{
    case 0: System.out.println("Sun"); break;
    case 1: System.out.println("Mon"); break;
    case 2: System.out.println("Tue"); break;
    case 3: System.out.println("Wed"); break;
    case 4: System.out.println("Thu"); break;
    case 5: System.out.println("Fri"); break;
    case 6: System.out.println("Sat"); break;
}
```

Arrays.



Compile-time initialization.

```
String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };

String[] rank =
{
    "2", "3", "4", "5", "6", "7", "8", "9", "10",
    "Jack", "Queen", "King", "Ace"
};
```

Typical array-processing code.

<i>create an array with random values</i>	<pre>double[] a = new double[N]; for (int i = 0; i < N; i++) a[i] = Math.random();</pre>
<i>print the array values, one per line</i>	<pre>for (int i = 0; i < N; i++) System.out.println(a[i]);</pre>
<i>find the maximum of the array values</i>	<pre>double max = Double.NEGATIVE_INFINITY; for (int i = 0; i < N; i++) if (a[i] > max) max = a[i];</pre>
<i>compute the average of the array values</i>	<pre>double sum = 0.0; for (int i = 0; i < N; i++) sum += a[i]; double average = sum / N;</pre>
<i>copy to another array</i>	<pre>double[] b = new double[N]; for (int i = 0; i < N; i++) b[i] = a[i];</pre>
<i>reverse the elements within an array</i>	<pre>for (int i = 0; i < N/2; i++) { double temp = b[i]; b[i] = b[N-1-i]; b[N-i-1] = temp; }</pre>

Two-dimensional arrays.

		a[1][2]	
	99	85	98
row 1 →	98	57	78
	92	77	76
	94	32	11
	99	34	22
	90	46	54
	76	59	88
	92	66	89
	97	71	24
	89	29	38
		column 2	

Compile-time initialization.

```
int[][] a =
{
    { 99, 85, 98, 0 },
    { 98, 57, 78, 0 },
    { 92, 77, 76, 0 },
    { 94, 32, 11, 0 },
    { 99, 34, 22, 0 },
    { 90, 46, 54, 0 },
    { 76, 59, 88, 0 },
    { 92, 66, 89, 0 },
    { 97, 71, 24, 0 },
    { 89, 29, 38, 0 },
    { 0, 0, 0, 0 }
};
```

Ragged arrays.

```
for (int i = 0; i < a.length; i++)
{
    for (int j = 0; j < a[i].length; j++)
        System.out.print(a[i][j] + " ");
    System.out.println();
}
```

Our standard output library.

public class StdOut	
void print(String s)	<i>print s</i>
void println(String s)	<i>print s, followed by newline</i>
void println()	<i>print a new line</i>
void printf(String f, ...)	<i>formatted print</i>

API for our library of static methods for standard output

The full [StdOut API](#).

format string number to print

StdOut.printf("%7.5f", Math.PI)

field width precision conversion code

Anatomy of a formatted print statement

<i>type</i>	<i>code</i>	<i>typical literal</i>	<i>sample format strings</i>	<i>converted string values for output</i>
int	d	512	"%14d" "%-14d"	" 512" "512"
double	f e	1595.1680010754388	"%14.2f" "%7f" "%14.4e"	" 1595.17" "1595.1680011" " 1.5952e+03"
String	s	"Hello, World"	"%14s" "%-14s" "%-14.5s"	" Hello, World" "Hello, World " "Hello "

Our standard input library.

<code>public class StdIn</code>		
boolean	isEmpty()	true if no more values, false otherwise
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
<i>API for our library of static methods for standard input</i>		

The full [StdIn API](#).

Our standard drawing library.


```

public class StdDraw
{
    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 (x0, x1)
    void setYscale(double y0, double y1)    reset y range to (y0, y1)
    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 w.png file
}

```

Note: Methods with the same names but no arguments reset to default values.

API for our library of static methods for standard drawing

The full [StdDraw API](#).

Our standard audio library.

```

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
}

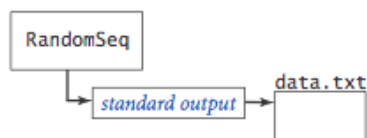
```

API for our library of static methods for standard audio

The full [StdAudio API](#).

Redirection and piping.

```
java RandomSeq 1000 > data.txt
```



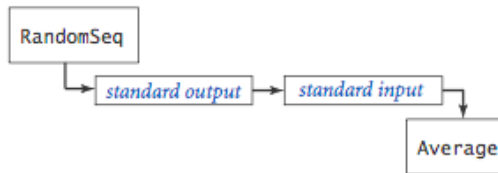
Redirecting standard output to a file

```
java Average < data.txt
```



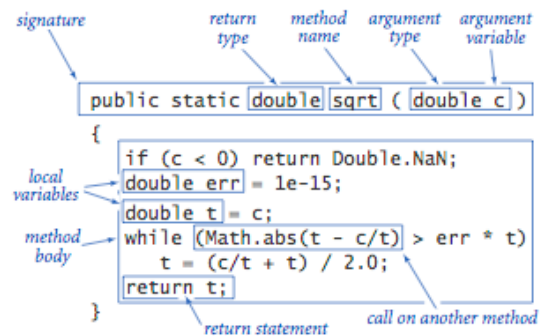
Redirecting from a file to standard input

```
java RandomSeq 1000 | java Average
```



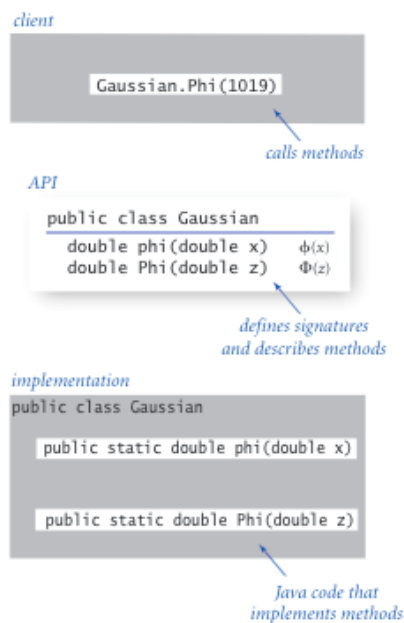
Piping the output of one program to the input of another

Functions.



<i>absolute value of an int value</i>	<pre>public static int abs(int x) { if (x < 0) return -x; else return x; }</pre>
<i>absolute value of a double value</i>	<pre>public static double abs(double x) { if (x < 0.0) return -x; else return x; }</pre>
<i>primality test</i>	<pre>public static boolean isPrime(int N) { if (N < 2) return false; for (int i = 2; i <= N/i; i++) if (N % i == 0) return false; return true; }</pre>
<i>hypotenuse of a right triangle</i>	<pre>public static double hypotenuse(double a, double b) { return Math.sqrt(a*a + b*b); }</pre>
<i>Harmonic number</i>	<pre>public static double H(int N) { double sum = 0.0; for (int i = 1; i <= N; i++) sum += 1.0 / i; return sum; }</pre>
<i>uniform random integer in [0, N)</i>	<pre>public static int uniform(int N) { return (int) (Math.random() * N); }</pre>
<i>draw a triangle</i>	<pre>public static void drawTriangle(double x0, double y0, double x1, double y1, double x2, double y2) { StdDraw.line(x0, y0, x1, y1); StdDraw.line(x1, y1, x2, y2); StdDraw.line(x2, y2, x0, y0); }</pre>

Libraries of functions.



Our standard random library.

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

Our standard statistics library.

<code>public class StdStats</code>		
<code>double max(double[] a)</code>		<i>largest value</i>
<code>double min(double[] a)</code>		<i>smallest value</i>
<code>double mean(double[] a)</code>		<i>average</i>
<code>double var(double[] a)</code>		<i>sample variance</i>
<code>double stddev(double[] a)</code>		<i>sample standard deviation</i>
<code>double median(double[] a)</code>		<i>median</i>
<code>void plotPoints(double[] a)</code>		<i>plot points at (i, a[i])</i>
<code>void plotLines(double[] a)</code>		<i>plot lines connecting points at (i, a[i])</i>
<code>void plotBars(double[] a)</code>		<i>plot bars to points at (i, a[i])</i>

Using an object.

declare a variable (object name)
`Charge c1;`
invoke a constructor to create an object
`c1 = new Charge(.51, .63, 21.3);`
`double v = c1.potentialAt(x, y);`
object name
invoke an instance method that operates on the object's value

Creating an object.

Instance variables.

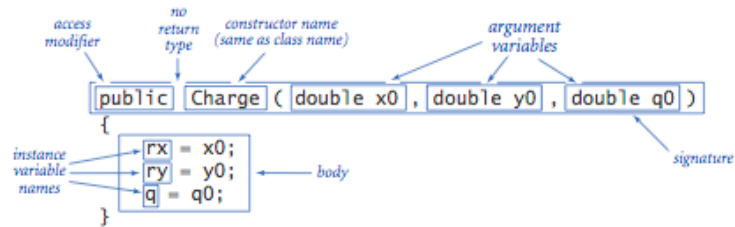
```

public class Charge
{
    private final double rx, ry;
    private final double q;
    .
    .
    .
}

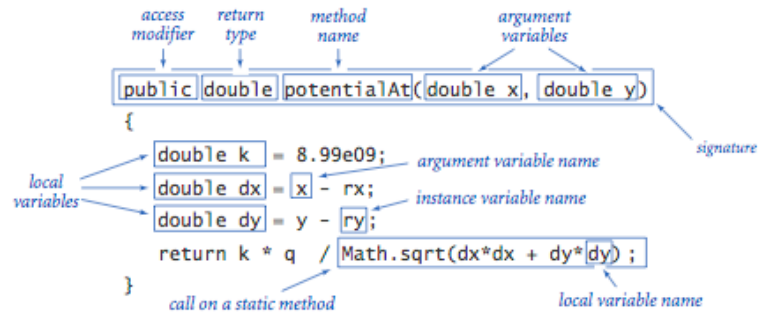
```

instance variable declarations
modifiers
Instance variables

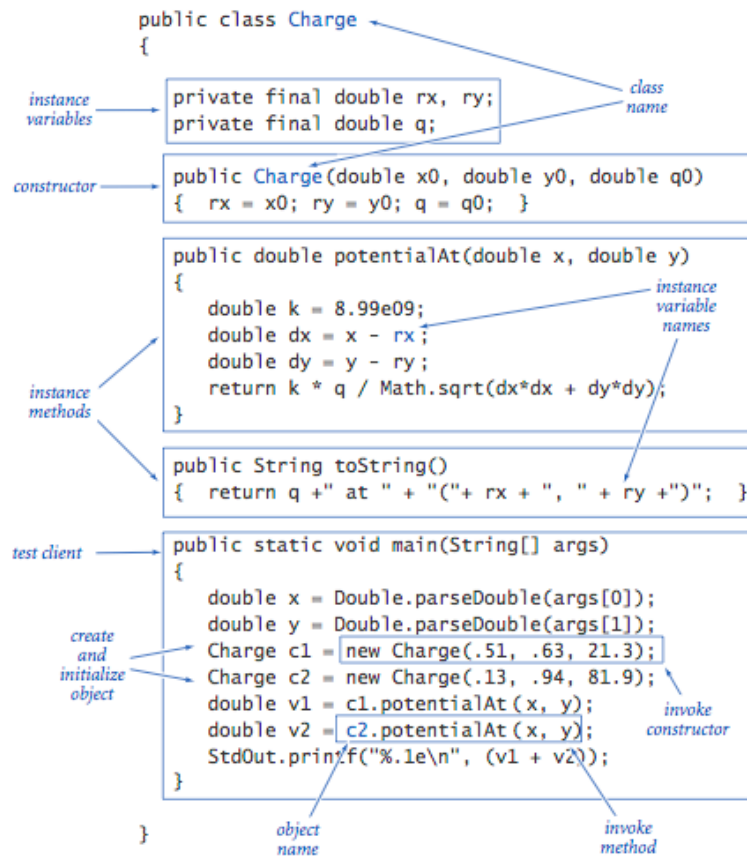
Constructors.

*Anatomy of a constructor*

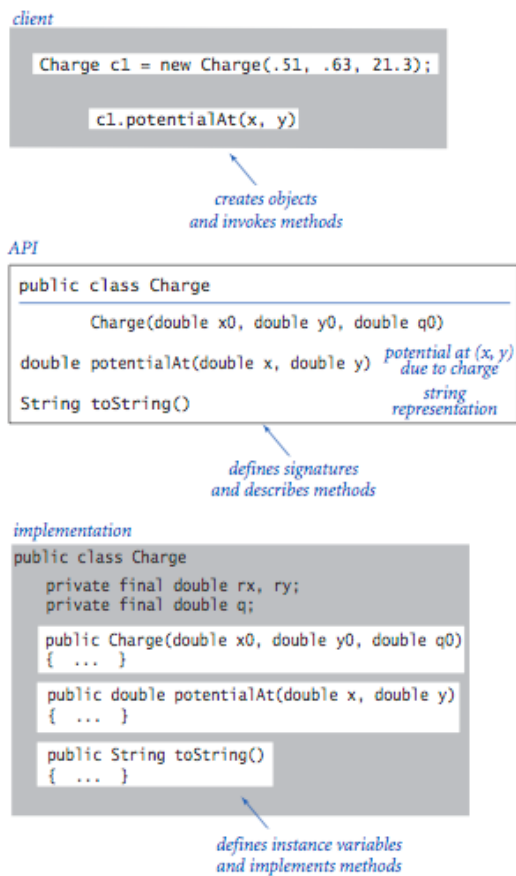
Instance methods.

*Anatomy of an instance method*

Classes.



Object-oriented libraries.



Java's String data type.

public class String (Java string data type)

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

The full `java.lang.String` API.

```
String a = "now is ";
String b = "the time ";
String c = "to"
```

call	value
a.length()	7
a.charAt(4)	i
a.substring(2, 5)	"w i"
b.startsWith("the")	true
a.indexOf("is")	4
a.concat(c)	"now is to"
b.replace('t','T')	"The Time "
a.split(" ")[0]	"now"
a.split(" ")[1]	"is"
b.equals(c)	false

Note: the [java.lang.StringBuilder](#) API is similar, but `StringBuilder` supports some operations more efficiently than `String` (notably, string concatenation) and some operations less efficiently (notably, substring extraction).

Java's Color data type.

```
public class java.awt.Color
```

Color(int r, int g, int b)		
int	getRed()	<i>red intensity</i>
int	getGreen()	<i>green intensity</i>
int	getBlue()	<i>blue intensity</i>
Color	brighter()	<i>brighter version of this color</i>
Color	darker()	<i>darker version of this color</i>
String	toString()	<i>string representation of this color</i>
boolean	equals(Color c)	<i>is this color's value the same as c's?</i>

The full [java.awt.Color](#) API.

Our input library.

```
public class In
```

In()		<i>create an input stream from standard input</i>
In(String name)		<i>create an input stream from a file or website</i>
boolean	isEmpty()	<i>true if no more input, false otherwise</i>
int	readInt()	<i>read a value of type int</i>
double	readDouble()	<i>read a value of type double</i>
...		

Note: All operations supported by `StdIn` are also supported for `In` objects.

The full [In](#) API.

Our output library.

public class Out

<code>Out()</code>	<i>create an output stream to standard output</i>
<code>Out(String name)</code>	<i>create an output stream to a file</i>
<code>void print(String s)</code>	<i>print s to the output stream</i>
<code>void println(String s)</code>	<i>print s and a newline to the output stream</i>
<code>void println()</code>	<i>print a newline to the output stream</i>
<code>void printf(String f, ...)</code>	<i>formatted print to the output steam</i>

The full [Out API](#).

Our picture library.

public class Picture

<code>Picture(String filename)</code>	<i>create a picture from a file</i>
<code>Picture(int w, int h)</code>	<i>create a blank w-by-h picture</i>
<code>int width()</code>	<i>return the width of the picture</i>
<code>int height()</code>	<i>return the height of the picture</i>
<code>Color get(int x, int y)</code>	<i>return the color of pixel (x, y)</i>
<code>void set(int x, int y, Color c)</code>	<i>set the color of pixel (x, y) to c</i>
<code>void show()</code>	<i>display the image in a window</i>
<code>void save(String filename)</code>	<i>save the image to a file</i>

The full [Picture API](#).

Compile-time and run-time errors. Here's a [list of errors](#) compiled by Mordechai Ben-Ari. It includes a list of common error message and typical mistakes that give rise to them.

Last modified on February 17, 2013.

Copyright © 2002–2012 [Robert Sedgewick](#) and [Kevin Wayne](#). All rights reserved.