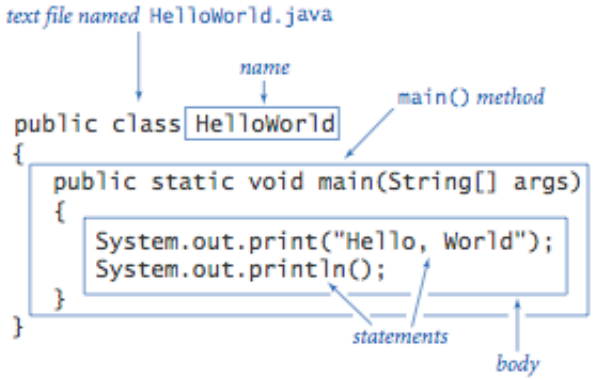
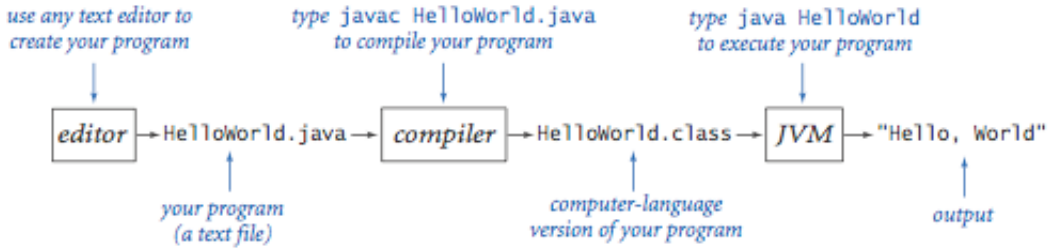
Appendix D: Java Programming Cheatsheet

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

Hello, World.



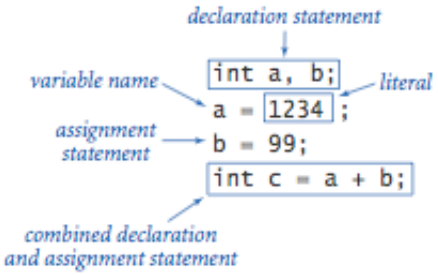
Editing, compiling, and executing.



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.

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

expression	*value*	*comment*
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.

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

expression	*value*
`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.

values	true or false		
literals	`true` `false`		
operations	and	or	not
operators	&&	||	!

a	!a
`true`	`false`
`false`	`true`

a	b	a && b	a || b
`false`	`false`	`false`	`false`
`false`	`true`	`false`	`true`
`true`	`false`	`false`	`true`
`true`	`true`	`true`	`true`

Comparison operators.

op	*meaning*	`true`	`false`
`==`	*equal*	`2 == 2`	`2 == 3`
`!=`	*not equal*	`3 != 2`	`2 != 2`
`<`	*less than*	`2 < 13`	`2 < 2`
`<=`	*less than or equal*	`2 <= 2`	`3 <= 2`
`>`	*greater than*	`13 > 2`	`2 > 13`
`>=`	*greater than or equal*	`3 >= 2`	`2 >= 3`

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* b*th 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)*

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

The full java.lang.Math API.

Type conversion.

expression	*expression type*	*expression value*
`"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.

absolute value

```
if (x < 0) x = -x;
```

put x and y into sorted order

```
if (x > y)
{
   int t = x;
   y = x;
   x = t;
}
```

maximum of x and y

```
if (x > y) max = x;
else       max = y;
```

error check for division operation

```
if (den == 0) System.out.println("Division by zero");
else          System.out.println("Quotient = " + num/den);
```

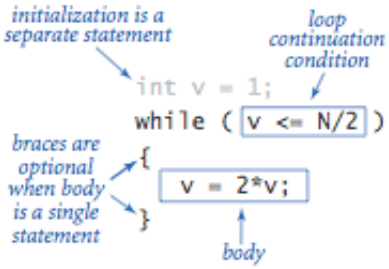
error check for quadratic formula

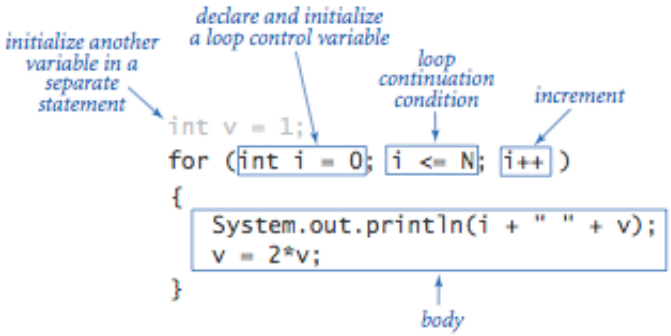
```
double discriminant = b*b - 4.0*c;
if (discriminant < 0.0)
{
   System.out.println("No real roots");
}
else
{
   System.out.println((-b + Math.sqrt(discriminant))/2.0);
   System.out.println((-b - Math.sqrt(discriminant))/2.0);
}
```

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.





print largest power of two less than or equal to N	`int v = 1;`
	`while (v <= N/2)`
	`  v = 2*v;`
	`System.out.println(v);`

compute a finite sum (1 + 2 + . . . + N)	`int sum = 0;`
	`for (int i = 1; i <= N; i++)`
	`  sum += i;`
	`System.out.println(sum);`

compute a finite product (N! = 1 × 2 × . . . × N)	`int product = 1;`
	`for (int i = 1; i <= N; i++)`
	`  product *= i;`
	`System.out.println(product);`

print a table of function values	`for (int i = 0; i <= N; i++)` `  System.out.println(i + " " + 2*Math.PI*i/N);`

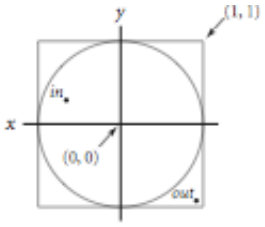
print the ruler function (see Program 1.2.1)	`String ruler = " ";`
	`for (int i = 1; i <= N; i++)`
	`  ruler = ruler + i + ruler;`
	`System.out.println(ruler);`

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.

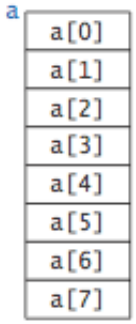
```
do
{
   x = 2.0*Math.random() - 1.0;
   y = 2.0*Math.random() - 1.0;
} while (Math.sqrt(x*x + y*y) > 1.0);
```



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.

create an array with random values	`double[] a = new double[N];`
	`for (int i = 0; i < N; i++)`
	`  a[i] = Math.random();`

print the array values, one per line	`for (int i = 0; i < N; i++)` `  System.out.println(a[i]);`

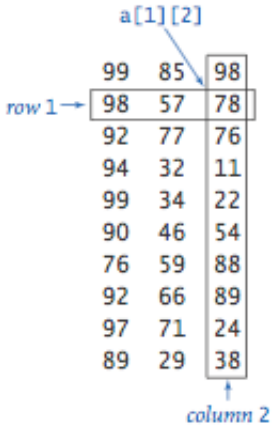
find the maximum of the array values	`double max = Double.NEGATIVE_INFINITY;`
	`for (int i = 0; i < N; i++)`
	`  if (a[i] > max) max = a[i];`

compute the average of the array values	`double sum = 0.0;`
	`for (int i = 0; i < N; i++)`
	`  sum += a[i];`
	`double average = sum / N;`

copy to another array	`double[] b = new double[N];`
	`for (int i = 0; i < N; i++)`
	`  b[i] = a[i];`

reverse the elements within an array	`for (int i = 0; i < N/2; i++)`
	`{`
	`  double temp = b[i];`
	`  b[i] = b[N-1-i];`
	`  b[N-i-1] = temp;`
	`}`

Two-dimensional arrays.



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.

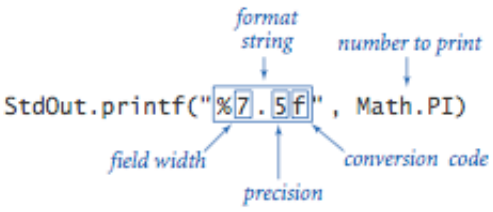
```
for (int i = 0; i < a.length; i++)
{
   for (int j = 0; j < a[i].length; j++)
      System.out.print(a[i][j] + " ");
   System.out.println();
}
```

Our standard output library.

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

API for our library of static methods for standard output

The full StdOut API.



Anatomy of a formatted print statement

type	*code*	*typical literal*	*sample format strings*	*converted string values for output*
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.

public class StdIn		
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*

API for our library of static methods for standard input

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 (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* 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.

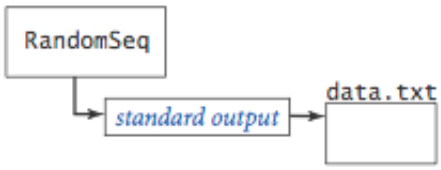
public class StdAudio		
void	play(String file)	*play the given* .wav *file*
void	play(double[] a)	*play the given sound wave*
void	play(double x)	*play sample for 1/44100 second*
void	save(String file, double[] a)	*save to a* .wav *file*
double[]	read(String file)	*read from a* .wav *file*

API for our library of static methods for standard audio

The full StdAudio API.

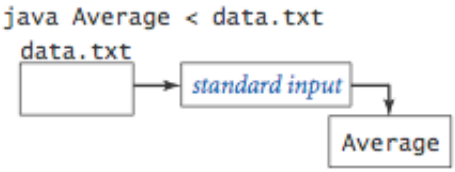
Redirection and piping.

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



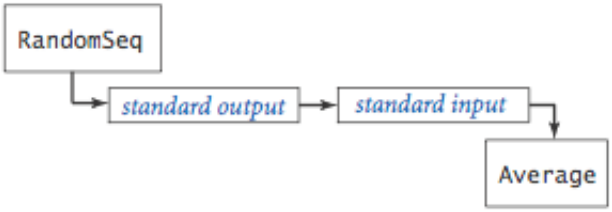
Redirecting standard output to a file

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



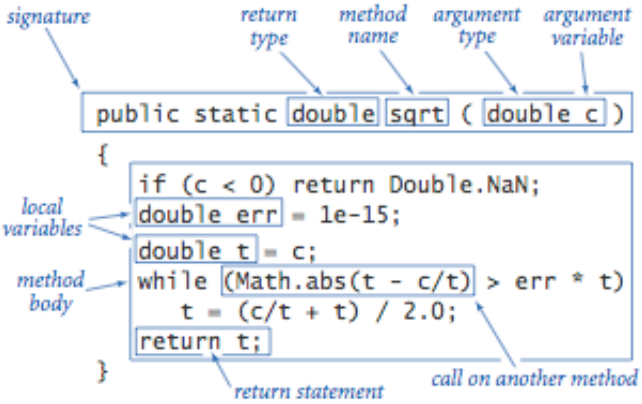
Redirecting from a file to standard input

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



Piping the output of one program to the input of another

Functions.



absolute value of an `int` *value*

```
public static int abs(int x)
{
   if (x < 0) return -x;
   else       return  x;
}
```

absolute value of a `double` *value*

```
public static double abs(double x)
{
   if (x < 0.0) return -x;
   else         return  x;
}
```

primality test

```
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;
}
```

hypotenuse of a right triangle

```
public static double hypotenuse(double a, double b)
{  return Math.sqrt(a*a + b*b);  }
```

Harmonic number

```
public static double H(int N)
{
   double sum = 0.0;
   for (int i = 1; i <= N; i++)
      sum += 1.0 / i;
   return sum;
}
```

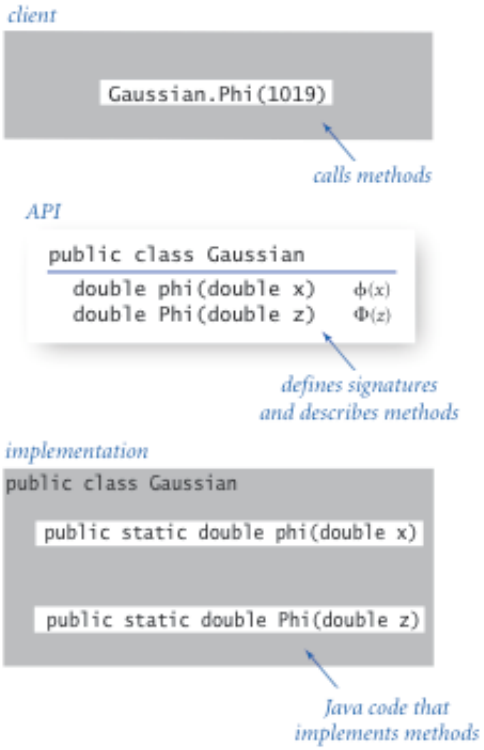
uniform random integer in $[0, N)$

```
public static int uniform(int N)
{  return (int) (Math.random() * N);  }
```

draw a triangle

```
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);
}
```

Libraries of functions.



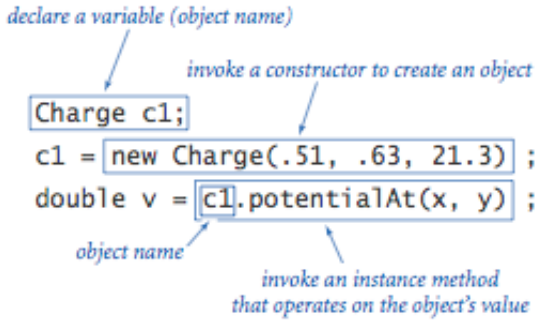
Our standard random library.

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[]

Our standard statistics library.

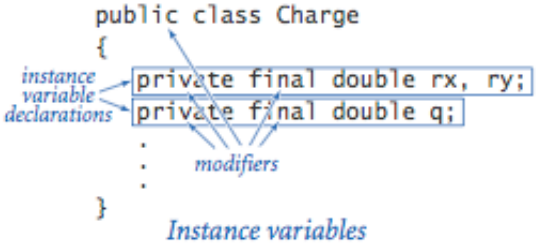
public class StdStats	
double max(double[] a)	*largest value*
double min(double[] a)	*smallest value*
double mean(double[] a)	*average*
double var(double[] a)	*sample variance*
double stddev(double[] a)	*sample standard deviation*
double median(double[] a)	*median*
void plotPoints(double[] a)	*plot points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

Using an object.

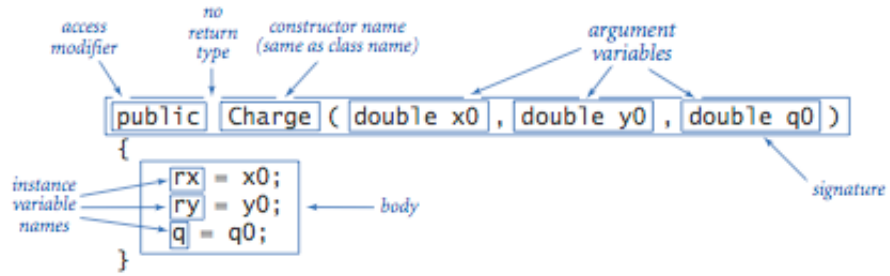


Creating an object.

Instance variables.

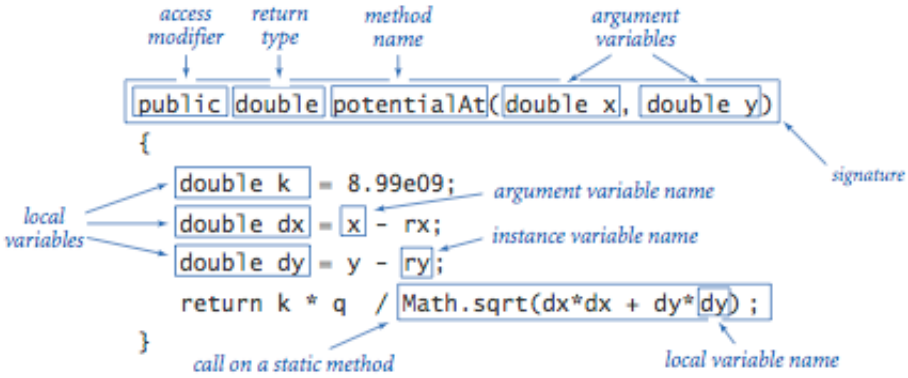


Constructors.



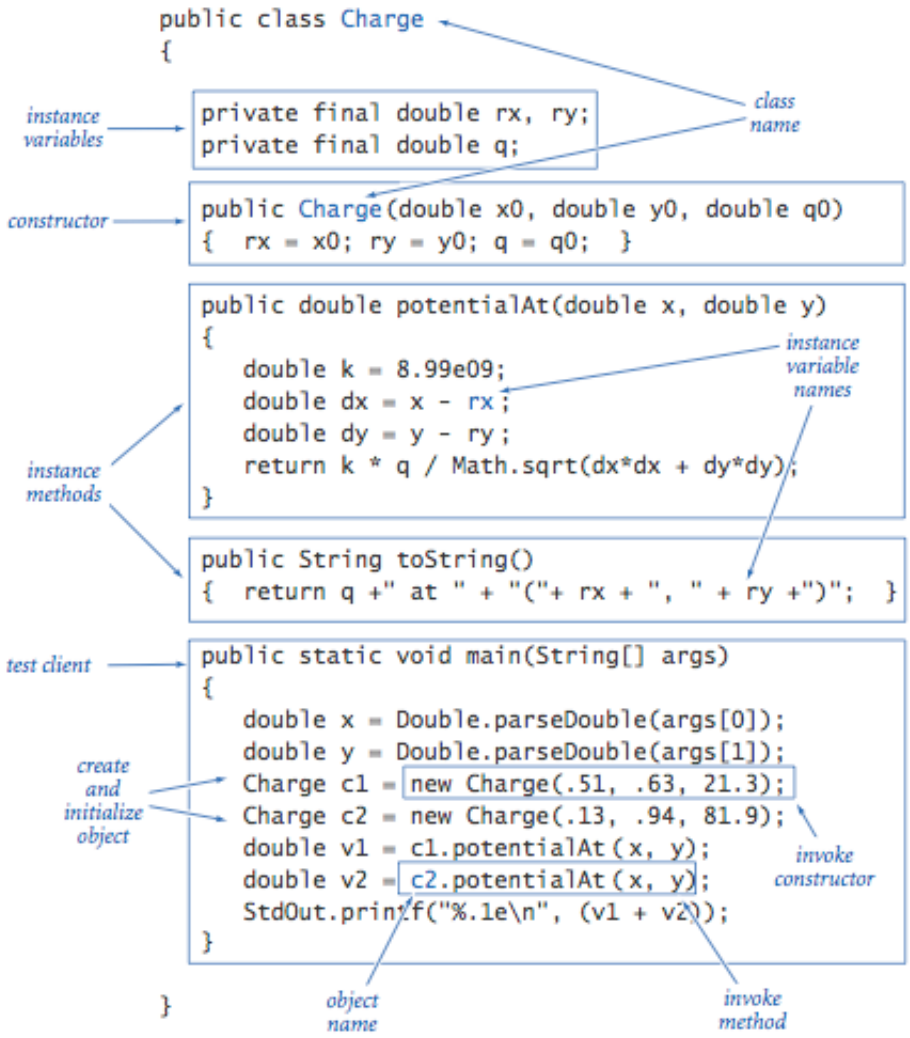
Anatomy of a constructor

Instance methods.

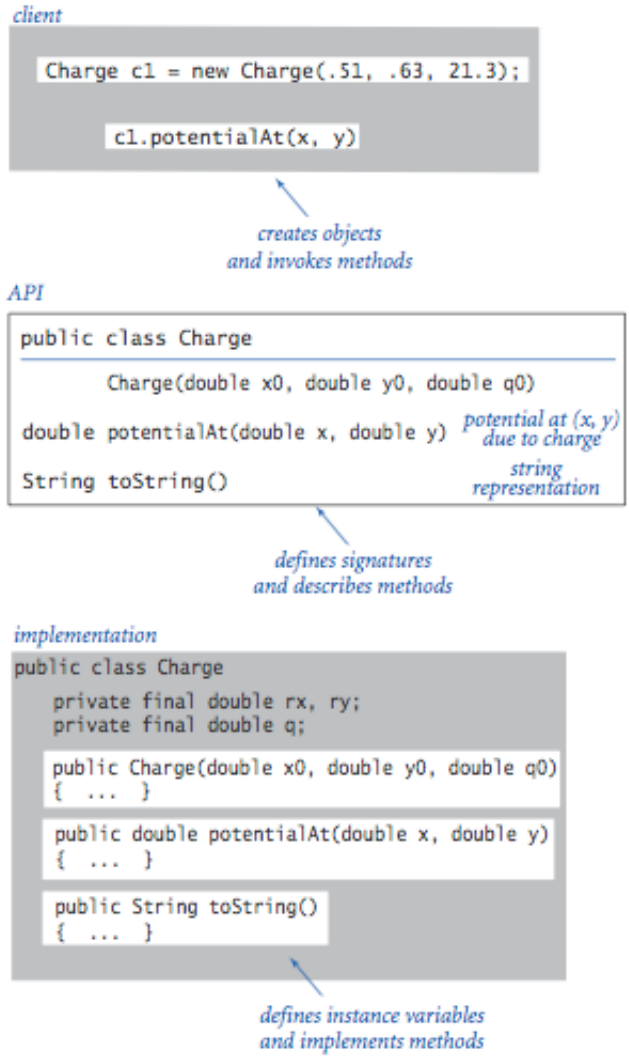


Anatomy of an instance method

Classes.



Object-oriented libraries.



Java's String data type.

public class String (Java string data type)

	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* as *to* bs
String[]	split(String delim)	*strings between occurrences of* delim
boolean	equals(String t)	*is this string's value the same as* t*'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()	*red intensity*
int	getGreen()	*green intensity*
int	getBlue()	*blue intensity*
Color	brighter()	*brighter version of this color*
Color	darker()	*darker version of this color*
String	toString()	*string representation of this color*
boolean	equals(Color c)	*is this color's value the same as* `c`*'s?*

The full java.awt.Color API.

Our input library.

`public class In`

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

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

The full In API.

Our output library.

public class Out	
Out()	*create an output stream to standard output*
Out(String name)	*create an output stream to a file*
void print(String s)	*print* s *to the output stream*
void println(String s)	*print* s *and a newline to the output stream*
void println()	*print a newline to the output stream*
void printf(String f, ...)	*formatted print to the output steam*

The full Out API.

Our picture library.

public class Picture	
Picture(String filename)	*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 x, int y)	*return the color of pixel* (x, y)
void set(int x, int y, Color c)	*set the color of pixel* (x, y) *to* c
void show()	*display the image in a window*
void save(String filename)	*save the image to a file*

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.