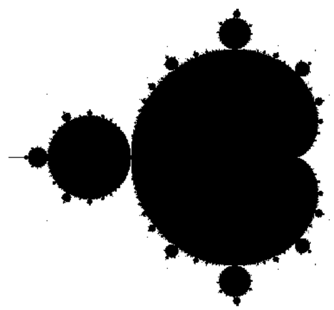
3.2 Creating Data Types



Data Types

Data type. Set of values and operations on those values.

Basic types.

Data Type	Set of Values	Some Operations
`boolean`	`true`, `false`	not, and, or, xor
`int`	-2^{31} to $2^{31} - 1$	add, subtract, multiply
`String`	sequence of Unicode characters	concatenate, compare

Last time. Write programs that use data types.
Today. Write programs to create our own data types.

Defining Data Types in Java

To define a data type, define:

- Set of values.
- Operations defined on them.

Java class. Allows us to define data types by specifying:

- Instance variables. (set of values)
- Methods. (operations defined on them)
- Constructors. (create and initialize new objects)

Point Charge Data Type

Goal. Create a data type to manipulate point charges.

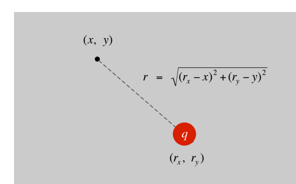
Set of values. Three real numbers. [position and electrical charge]

Operations.

- Create a new point charge at (r_x, r_y) with electric charge q.
- Determine electric potential V at (x, y) due to point charge.
- Convert to string.

$$V = k \frac{q}{r}$$

r = distance between (x, y) and (r_x, r_y)
k = electrostatic constant = 8.99×10^9 N $\cdot$m^2 / C^2



Charge Data Type: A Simple Client

Client program. Uses data type operations to calculate something.

```
public static void main(String[] args) {
   double x = Double.parseDouble(args[0]);
   double y = Double.parseDouble(args[1]);
   Charge c1 = new Charge(.51, .63, 21.3);
   Charge c2 = new Charge(.13, .94, 81.9);
   double v1 = c1.potentialAt(x, y);
   double v2 = c2.potentialAt(x, y);
   StdOut.println(c1);
   StdOut.println(c2);
   StdOut.println(v1 + v2);
}
```

automagically invokes the `toString()` method

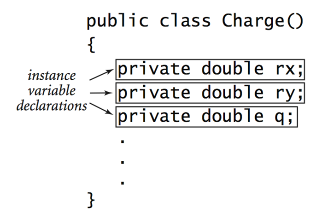
```
% java Charge .50 .50
21.3 at (0.51, 0.63)
81.9 at (0.13, 0.94)
2.74936907085912E12
```


Anatomy of Instance Variables

Instance variables. Specifies the set of values.

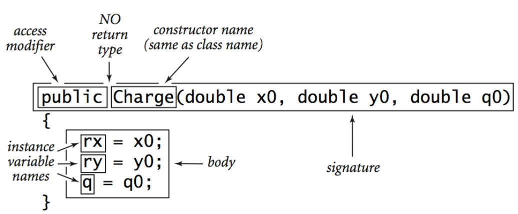
- Declare outside any method.
- Always use access modifier `private`.

stay tuned



Anatomy of a Constructor

Constructor. Invoke with `new` to create new objects.

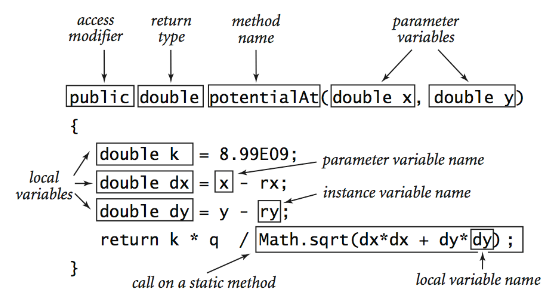



```
Charge c1 = new Charge(.51, .63, 21.3);
Charge c2 = new Charge(.13, .94, 81.9);
```

invoking a constructor

Anatomy of a Data Type Method

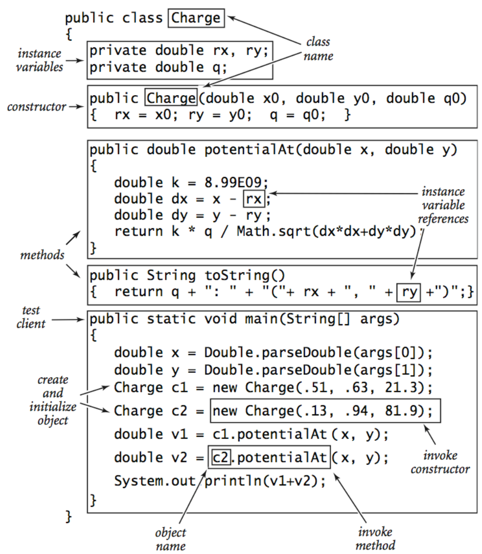
Method. Define operations on instance variables.




```
double v1 = c1.potentialAt(x, y);
double v2 = c2.potentialAt(x, y);
```

invoking a method

Anatomy of a Class

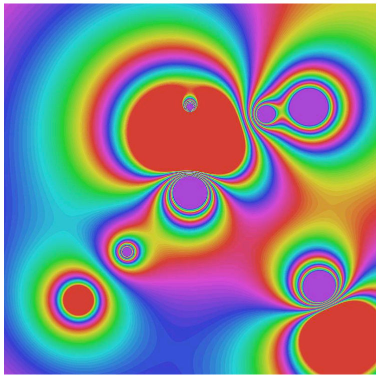


Potential Visualization

Potential visualization. Read in N point charges from a file; compute total potential at each point in unit square.

```
% more charges.txt
9
.51 .63 -100
.50 .50   40
.50 .72   10
.33 .33    5
.20 .20  -10
.70 .70   10
.82 .72   20
.85 .23   30
.90 .12  -50
```

```
% java Potential < charges.txt
```



Potential Visualization

Arrays of objects. Allocate memory for the array; then allocate memory for each individual object.

```
// read in the data
int N = StdIn.readInt();
Charge[] a = new Charge[N];
for (int i = 0; i < N; i++) {
   double x0 = StdIn.readDouble();
   double y0 = StdIn.readDouble();
   double q0 = StdIn.readDouble();
   a[i] = new Charge(x0, y0, q0);
}
```


Potential Visualization

```
// plot the data
int SIZE = 512;
Picture pic = new Picture(SIZE, SIZE);
for (int row = 0; row < SIZE; row++) {
   for (int col = 0; col < SIZE; col++) {
      double V = 0.0;
      for (int i = 0; i < N; i++) {
         double x = 1.0 * row / SIZE;
         double y = 1.0 * col / SIZE;
         V += a[i].potentialAt(x, y);
      }
      Color color = getColor(V);
      pic.set(row, SIZE-1-col, color);
   }
}
pic.show();
```

$$V = \sum_i (k\, q_i \,/\, r_i)$$

compute color as a function of potential V

(0, 0) is upper left

Complex Numbers

Complex Number Data Type

Goal. Create a data type to manipulate complex numbers.
Set of values. Two real numbers: real and imaginary parts.

API. `public class Complex` (PROGRAM 3.2.2)

	`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*
`String`	`toString()`	*string representation*

a = 3 + 4i, b = -2 + 3i
a + b = 1 + 7i
a × b = -18 + i
|a| = 5

Applications of Complex Numbers

Relevance. A quintessential mathematical abstraction.

Applications.
- Fractals.
- Impedance in RLC circuits.
- Signal processing and Fourier analysis.
- Control theory and Laplace transforms.
- Quantum mechanics and Hilbert spaces.
- …

Complex Number Data Type: A Simple Client

Client program. Uses data type operations to calculate something.

```
public static void main(String[] args) {
   Complex a = new Complex( 3.0, 4.0);
   Complex b = new Complex(-2.0, 3.0);
   Complex c = a.times(b);
   StdOut.println("a = " + a);
   StdOut.println("b = " + b);
   StdOut.println("c = " + c);
}
```

result of `c.toString()`

```
% java TestClient
a = 3.0 + 4.0i
b = -2.0 + 3.0i
c = -18.0 + 1.0i
```

Remark. Can't write `a = b*c` since no operator overloading in Java.

Complex Number Data Type: Implementation

```
public class Complex {

   private double re;
   private double im;

   public Complex(double real, double imag) {
      re = real;
      im = imag;
   }

   public String toString() { return re + " + " + im + "i"; }

   public double abs() { return Math.sqrt(re*re + im*im); }

   public Complex plus(Complex b) {
      double real = re + b.re;
      double imag = im + b.im;
      return new Complex(real, imag);
   }

   public Complex times(Complex b) {
      double real = re * b.re - im * b.im;
      double imag = re * b.im + im * b.re;
      return new Complex(real, imag);
   }
}
```

instance variables

constructor

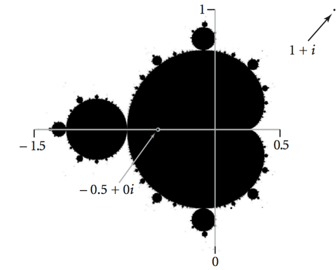
creates a Complex object, and returns a reference to it

refers to b's instance variable

methods

Mandelbrot Set

Mandelbrot set. A set of complex numbers.
Plot. Plot (x, y) black if z = x + iy is in the set, and white otherwise.

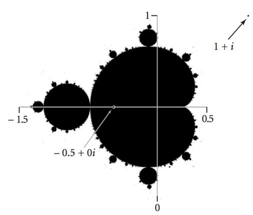


- No simple formula describes which complex numbers are in set.
- Instead, describe using an algorithm.

Mandelbrot Set

Mandelbrot set. Is complex number z_0 is in set?

- Iterate $z_{t+1} = (z_t)^2 + z_0$.
- If $|z_t|$ diverges to infinity, then z_0 not in set; otherwise z_0 is in set.



t	z_t
0	-1/2 + 0i
1	-1/4 + 0i
2	-7/16 + 0i
3	-79/256 + 0i
4	-26527/65536 + 0i
5	-1443801919/4294967296 + 0i

z = -1/2 is in Mandelbrot set

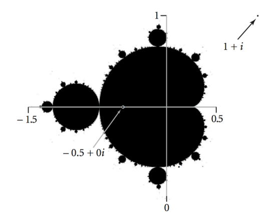
t	z_t
0	1 + i
1	1 + 3i
2	-7 + 7i
3	1 - 97i
4	-9407 - 193i
5	88454401 + 3631103i

z = 1 + i not in Mandelbrot set

Plotting the Mandelbrot Set

Practical issues.

- Cannot plot infinitely many points.
- Cannot iterate infinitely many times.



Approximate solution.

- Sample from an N-by-N grid of points in the plane.
- Fact: if $|z_t| > 2$ for any t, then z not in Mandelbrot set.
- Pseudo-fact: if $|z_{255}| \le 2$ then z "likely" in Mandelbrot set.

Complex Number Data Type: Another Client

Mandelbrot function with complex numbers.

- Is z in the Mandelbrot set?
- Returns white (definitely no) or black (probably yes).

```
public static Color mand(Complex z0) {
   Complex z = z0;
   for (int t = 0; t < 255; t++) {
      if (z.abs() > 2.0) return Color.WHITE;
      z = z.times(z);
      z = z.plus(z0);        // z = z^2 + z0
   }
   return Color.BLACK;
}
```

$z = z^2 + z_0$

More dramatic picture: replace `Color.WHITE` with grayscale or color.

`new Color(255-t, 255-t, 255-t)`

Complex Number Data Type: Another Client

Plot the Mandelbrot set in gray scale.

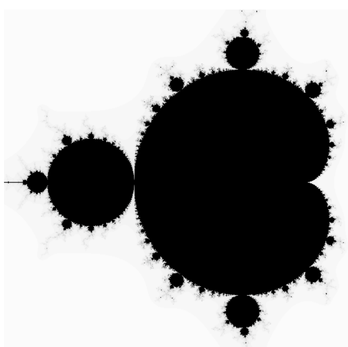
```
public static void main(String[] args) {
   double xc   = Double.parseDouble(args[0]);
   double yc   = Double.parseDouble(args[1]);
   double size = Double.parseDouble(args[2]);
   int N = 512;
   Picture pic = new Picture(N, N);

   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
         double x0 = xc - size/2 + size*i/N;
         double y0 = yc - size/2 + size*j/N;
         Complex z0 = new Complex(x0, y0);
         Color color = mand(z0);
         pic.set(i, N-1-j, color);
      }
   }
   pic.show();
}
```

scale to screen coordinates

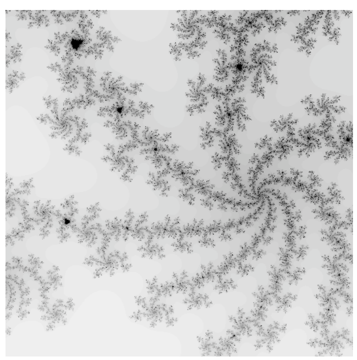
(0, 0) is upper left

Mandelbrot Set

`% java Mandelbrot -.5 0 2`



`% java Mandelbrot .1045 -.637 .01`



Vector Data Type

Set of values. Sequence of real numbers. [Cartesian coordinates]

API.

public class Vector (PROGRAM 3.2.4)

	Vector(double[] a)	
Vector	plus(Vector b)	*sum 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*

$x = (0, 3, 4, 0)$, $y = (0, -3, 1, -4)$

$x + y = (0, 0, 5, -4)$

$3x = (0, 9, 12, 0)$

$x \cdot y = (0 \cdot 0) + (3 \cdot -3) + (4 \cdot 1) + (0 \cdot -4) = -5$

$|x| = (0^2 + 3^2 + 4^2 + 0^2)^{1/2} = 5$

$\vec{x} = x \,/\, |x| = (0, 0.6, 0.8, 0)$

Vector Data Type Applications

Relevance. A quintessential mathematical abstraction.

Applications.

- Statistics.
- Linear algebra.
- Clustering and similarity search.
- Force, velocity, acceleration, momentum, torque.
- ...

Vector Data Type: Implementation

```
public class Vector {
   private int N;
   private double[] coords;                    instance variables

   public Vector(double[] a) {
      N = a.length;
      coords = new double[N];
      for (int i = 0; i < N; i++)
         coords[i] = a[i];
   }                                           constructor

   public double dot(Vector b) {
      double sum = 0.0;
      for (int i = 0; i < N; i++)
         sum += (coords[i] * b.coords[i]);
      return sum;
   }

   public Vector plus(Vector b) {
      double[] c = new double[N];
      for (int i = 0; i < N; i++)
         c[i] = coords[i] + b.coords[i];
      return new Vector(c);                    methods
   }
```


Vector Data Type: Implementation

```
public Vector times(double t) {
   double[] c = new double[N];
   for (int i = 0; i < N; i++)
      c[i] = t * coords[i];
   return new Vector(c);
}

public double magnitude() {
   return Math.sqrt(this.dot(this));
}

public Vector direction() {
   return this.times(1.0 / this.magnitude());
}
...
```

This. The keyword `this` is a reference to the invoking object.
Ex. When you invoke `a.magnitude()`, `this` is an alias for `a`.

Applications of Data Types

Data type. Set of values and collection of operations on those values.

Simulating the physical world.

- Java objects model real-world objects.
- Not always easy to make model reflect reality.
- Ex: charged particle, molecule, COS 126 student, ...

Extending the Java language.

- Java doesn't have a data type for every possible application.
- Data types enable us to add our own abstractions.
- Ex: complex, vector, polynomial, matrix, ...