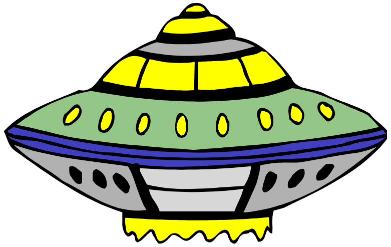
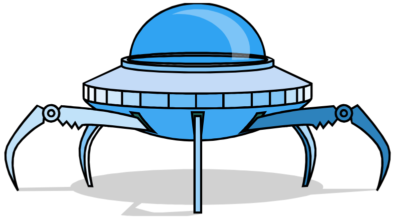
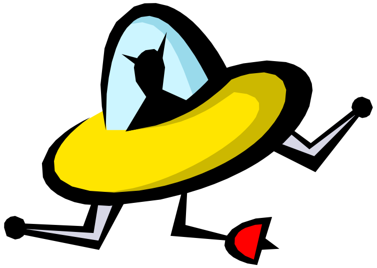
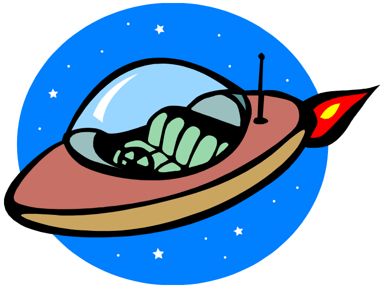
More on objects







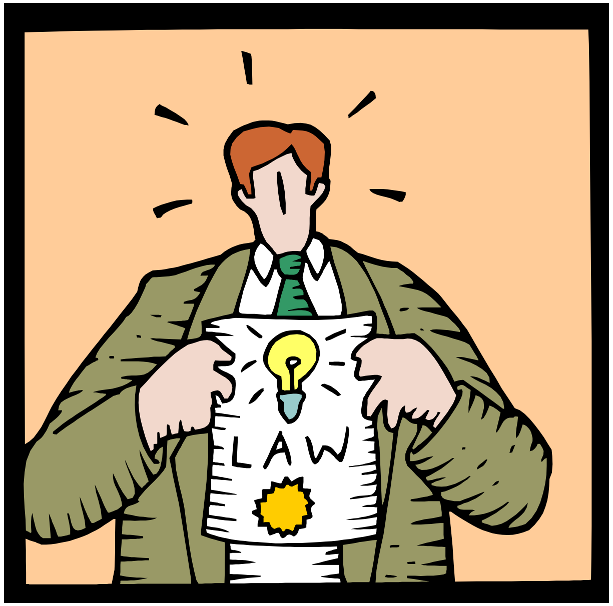


Overview

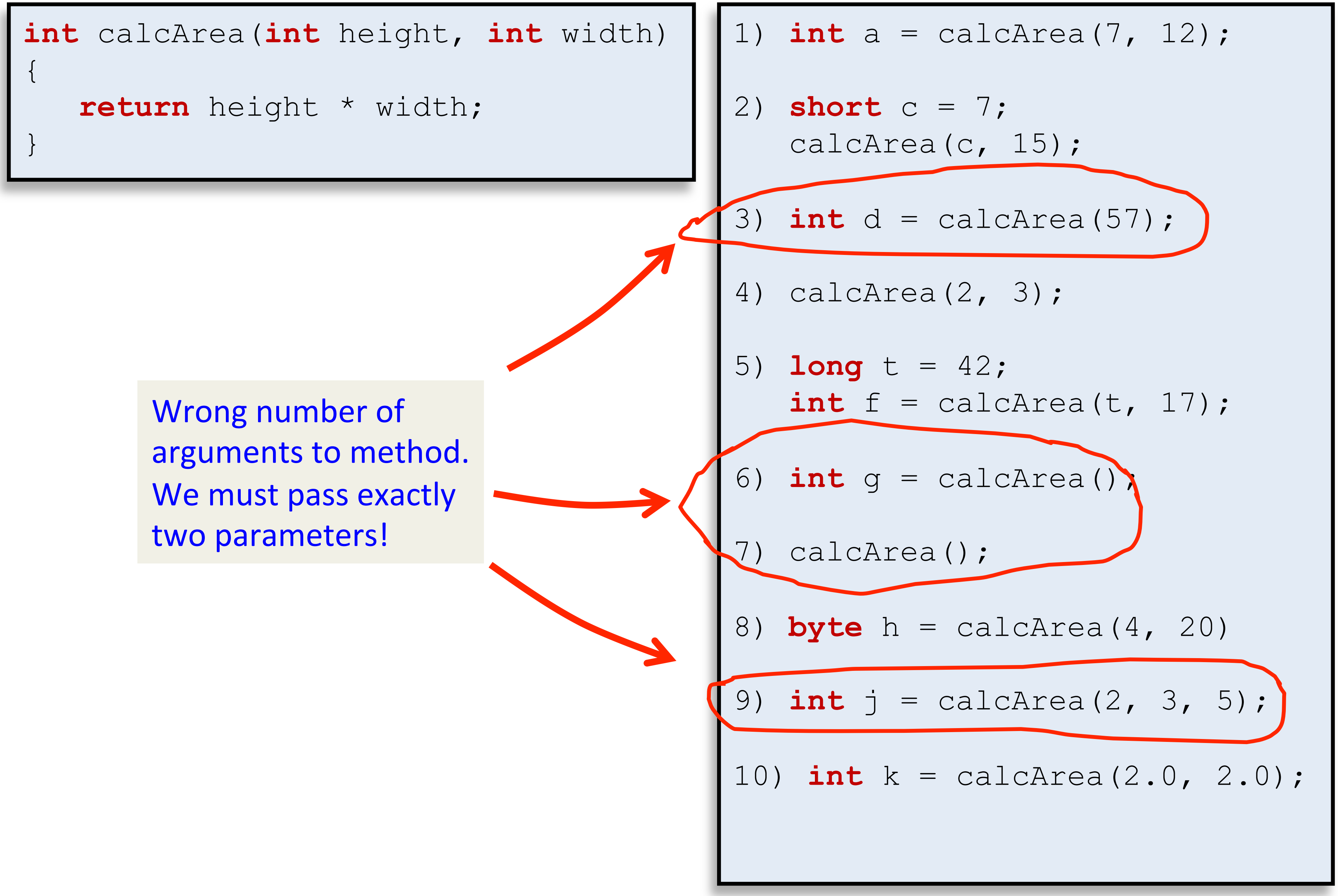
- Methods
  - Parameter and return type puzzler
- Increment/decrement
- Application Programming Interface (API)
  - ChargedParticle
  - ColorSeparation

```
int calcArea(int height, int width)
{
   return height * width;
}
```

Given the method above, which of the methods calls on the right are legal?



```
1)  int a = calcArea(7, 12);

2)  short c = 7;
    calcArea(c, 15);

3)  int d = calcArea(57);

4)  calcArea(2, 3);

5)  long t = 42;
    int f = calcArea(t, 17);

6)  int g = calcArea();

7)  calcArea();

8)  byte h = calcArea(4, 20)

9)  int j = calcArea(2, 3, 5);

10)  int k = calcArea(2.0, 2.0);
```



```
int calcArea(int height, int width)
{
   return height * width;
}
```

```
1)  int a = calcArea(7, 12);

2)  short c = 7;
    calcArea(c, 15);

3)  int d = calcArea(57);

4)  calcArea(2, 3);

5)  long t = 42;
    int f = calcArea(t, 17);

6)  int g = calcArea();

7)  calcArea();

8)  byte h = calcArea(4, 20)

9)  int j = calcArea(2, 3, 5);

10)  int k = calcArea(2.0, 2.0);
```

Wrong number of arguments to method. We must pass exactly two parameters!

```
int calcArea(int height, int width)
{
   return height * width;
}
```

```
1) int a = calcArea(7, 12);

2) short c = 7;
   calcArea(c, 15);

~~3) int d = calcArea(57);~~

4) calcArea(2, 3);

5) long t = 42;
   int f = calcArea(t, 17);

~~6) int g = calcArea();~~

~~7) calcArea();~~

8) byte h = calcArea(4, 20)

~~9) int j = calcArea(2, 3, 5);~~

10) int k = calcArea(2.0, 2.0);
```

Parameter type problem.
A `long` won't fit into an `int`
parameters without spilling.

Return type problem.
Method returns an `int` which won't
fint into a `byte` without spilling.

Parameter type problem.
The `double`'s won't demote to
lowly `int` paramters.

```
int calcArea(int height, int width)
{
   return height * width;
}
```

```
1) int a = calcArea(7, 12);

2) short c = 7;
   calcArea(c, 15);

~~3) int d = calcArea(57);~~

4) calcArea(2, 3);

~~5) long t = 42;~~
~~   int f = calcArea(t, 17);~~

~~6) int g = calcArea();~~

~~7) calcArea();~~

~~8) byte h = calcArea(4, 20)~~

~~9) int j = calcArea(2, 3, 5);~~

~~10) int k = calcArea(2.0, 2.0);~~
```

Lovely. Just how we'd expect somebody to do it!

First parameter is a `short` but it can fit in an `int` parameter since it is a bigger data type.

Sort of weird but it will compile. We get an `int` result back, but we just ignore it.

```
double calcArea(double height,
                double width)
{
   return height * width;
}
```

Which are legal if instead the method took two `double`'s and returned a `double`?

```
1)  int a = calcArea(7, 12);

2)  short c = 7;
    calcArea(c, 15);

3)  double d = calcArea(7.0, 2);

4)  double e = calcArea(7, 2.0);

5)  double f = calcArea(7.2, 2.0);

6)  int g = calcArea(7.2, 2.0);

7)  float  h = 1.99f;
    double i = calcArea(f, f);

8)  double j = calcArea("7.0",
                        "12.0");

9)  String k = "" + calcArea(1, 2);

10)  double m = calcArea(-1.0, -9.0);
```

```
double calcArea(double height,
                double width)
{
   return height * width;
}
```

Parameters 7 and 12 promote to `double`, but return value can't demote to an `int`.

Parameters are fine, but return value can't demote to an `int`.

```
1) int a = calcArea(7, 12);

2) short c = 7;
   calcArea(c, 15);

3) double d = calcArea(7.0, 2);

4) double e = calcArea(7, 2.0);

5) double f = calcArea(7.2, 2.0);

6) int g = calcArea(7.2, 2.0);

7) float  h = 1.99f;
   double i = calcArea(f, f);

8) double j = calcArea("7.0",
                       "12.0");

9) String k = "" + calcArea(1, 2);

10) double m = calcArea(-1.0, -9.0);
```

```
double calcArea(double height,
                double width)
{
   return height * width;
}
```

```
~~1) int a = calcArea(7, 12);~~

2) short c = 7;
   calcArea(c, 15);

3) double d = calcArea(7.0, 2);

4) double e = calcArea(7, 2.0);

5) double f = calcArea(7.2, 2.0);

~~6) int g = calcArea(7.2, 2.0);~~

7) float  h = 1.99f;
   double i = calcArea(f, f);

8) double j = calcArea("7.0",
                       "12.0");

9) String k = "" + calcArea(1, 2);

10) double m = calcArea(-1.0, -9.0);
```

Parameters are of type `String` and won't convert to `double` without a call to `Double.parseDouble()`

```
double calcArea(double height,
                double width)
{
    return height * width;
}
```

```
~~1) int a = calcArea(7, 12);~~

2) short c = 7;
   calcArea(c, 15);

3) double d = calcArea(7.0, 2);

4) double e = calcArea(7, 2.0);

5) double f = calcArea(7.2, 2.0);

~~6) int g = calcArea(7.2, 2.0);~~

7) float  h = 1.99f;
   double i = calcArea(h, h);

~~8) double j = calcArea("7.0",~~
                    ~~"12.0");~~

9) String k = "" + calcArea(1, 2);

10) double m = calcArea(-1.0, -9.0);
```

Types such as `short`, `int`, and `float` will all type promote to `double` if needed.

The `double` return result can be appended to a `String` using + (but we must have the blank string "" first).

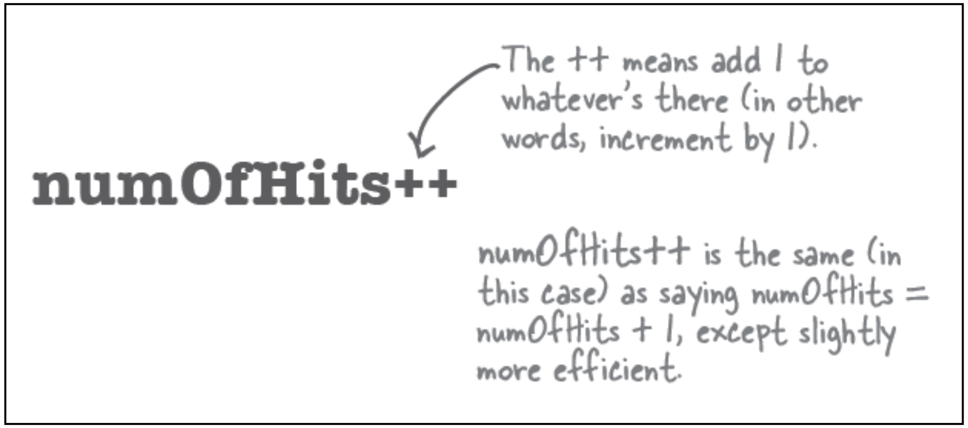
Increment and decrement

```
x = x + 1;

x += 1;

x++;

++x;
```

Each line increments `x` by one.

```
x = x - 1;

x -= 1;

x--;

--x;
```

Each line decrements `x` by one.



numOfHits++

The ++ means add 1 to whatever's there (in other words, increment by 1).

numOfHits++ is the same (in this case) as saying numOfHits = numOfHits + 1, except slightly more efficient.

Incrementing 1 trillion times

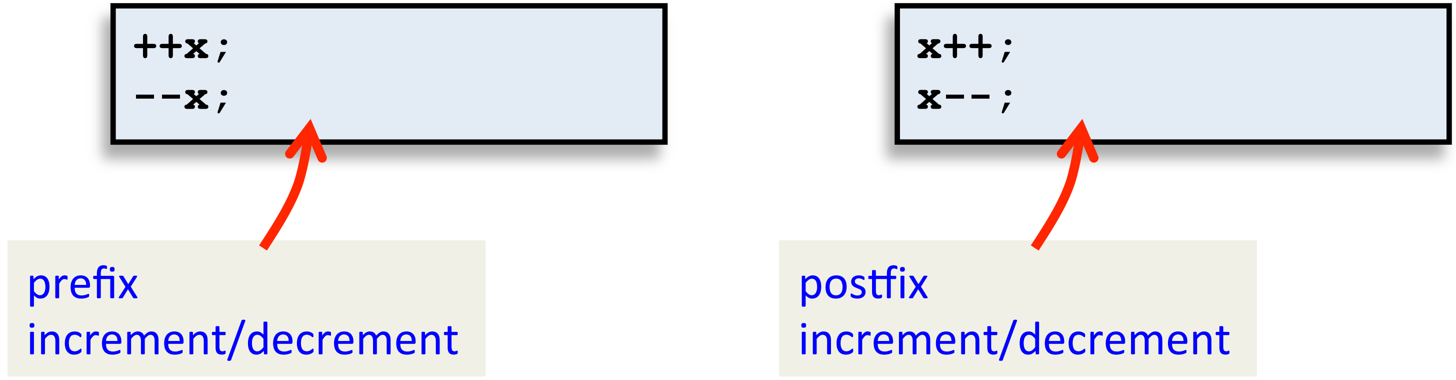
```
public class IncrementSpeed
{
   public static void main
   {
      long num   = Long.pa
      long val   = 0;
      long start = System.currentTimeMillis();
      for (long i = 0; i < num; i++)
         val = val + 1;
      long elapsed = System.currentTimeMillis() - start;
      System.out.println("Time = " + (elapsed / 1000.0));
   }
}
```

```
% java IncrementSpeed 1000000000000
Time = 592.153
```

```
public class IncrementSpeed2
{
   public static void main(String[] args)
   {
      long num   = Long.pa
      long val   = 0;
      long start = System.
      for (long i = 0; i < num; i++)
         val++;
      long elapsed = System.currentTimeMillis() - start;
      System.out.println("Time = " + (elapsed / 1000.0));
   }
}
```

```
% java IncrementSpeed2 1000000000000
Time = 594.194
```

Pre and post increment/decrement



```
++x;
--x;
```

prefix increment/decrement

```
x++;
x--;
```

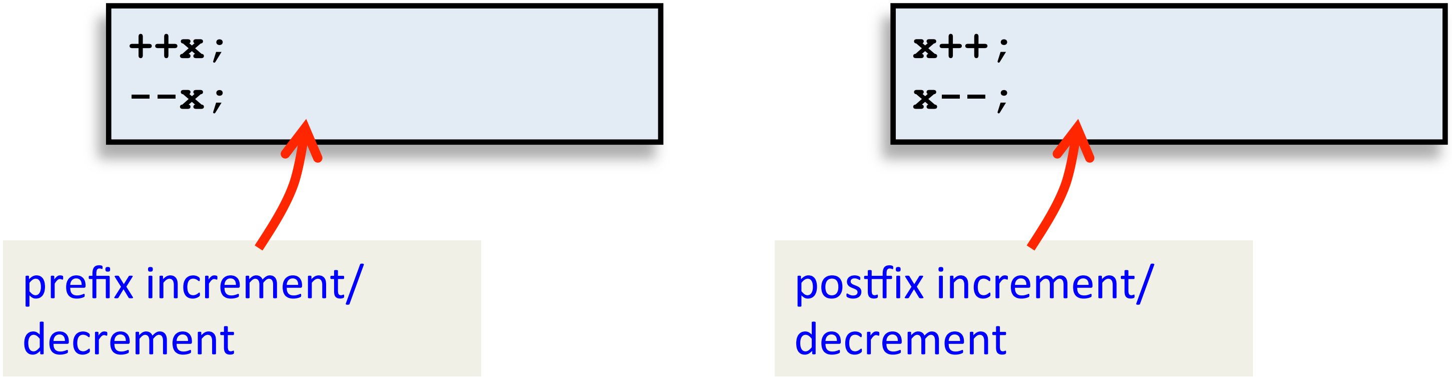
postfix increment/decrement

- If used on a line by itself, no difference
  - Use whichever one you fancy!
  - Otherwise, you better know what you are doing.

```
int x = 0;
int z = ++x;
System.out.println("x=" + x +
                   ", z=" + z);
```

```
int x = 0;
int z = x++;
System.out.println("x=" + x +
                   ", z=" + z);
```

Pre and post increment/decrement



```
++x;
--x;
```

prefix increment/ decrement

```
x++;
x--;
```

postfix increment/ decrement

- If used on a line by itself, no difference
  - Use whichever one you fancy!
  - Otherwise, you better know what you are doing.

```
int x = 0;
int z = ++x;
System.out.println("x=" + x +
                   ", z=" + z);
```

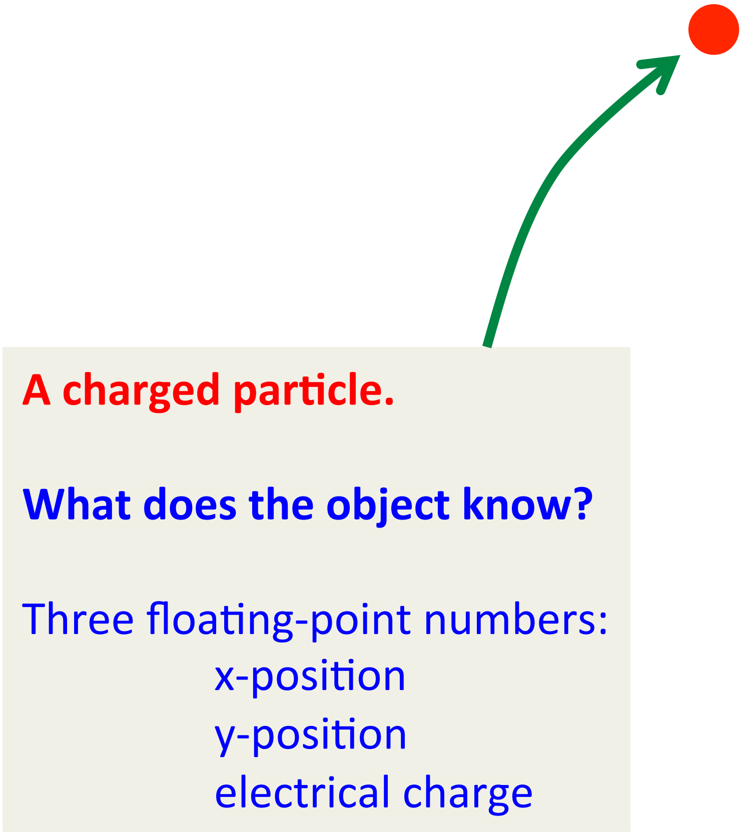
```
x=1, z=1
```

```
int x = 0;
int z = x++;
System.out.println("x=" + x +
                   ", z=" + z);
```

```
x=1, z=0
```

Using objects

- Assume we are given an object data type that represents a charged particle.



A charged particle.

What does the object know?

Three floating-point numbers:

x-position
y-position
electrical charge

Using objects

- Assume we are given an object data type that represents a charged particle.

A charged particle.

What does the object know?

Three floating-point numbers:
x-position
y-position
electrical charge

What can the object do?

Calculate the electrical potential at a point (x, y) given the particle's x-position, y-position, and charge.

Print itself out to the console.

Charged particle API

- API (Application Programming Interface)
  – Public specification for what a class does
  – All a client program needs to know
  – Signature, return type, and comments for all public methods
- API for charged particle class:

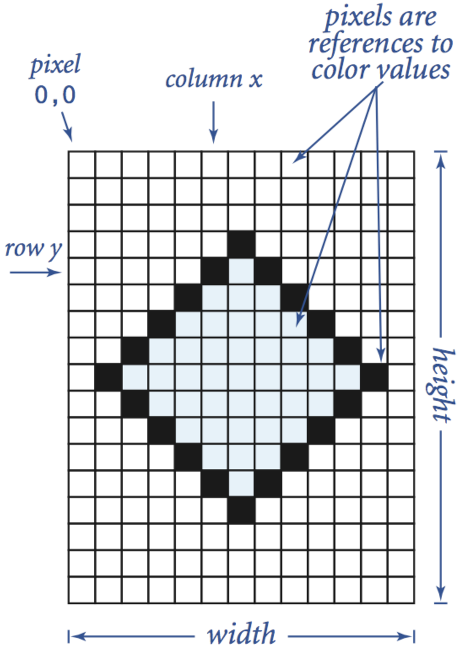
```
public class Charge
--------------------------------------------
       Charge(double x0, double y0, double q0) // location and charge
double potentialAt(double x, double y)         // potential @ (x,y) due to charge
String toString()                              // string representation
```

FourChargeClient solution

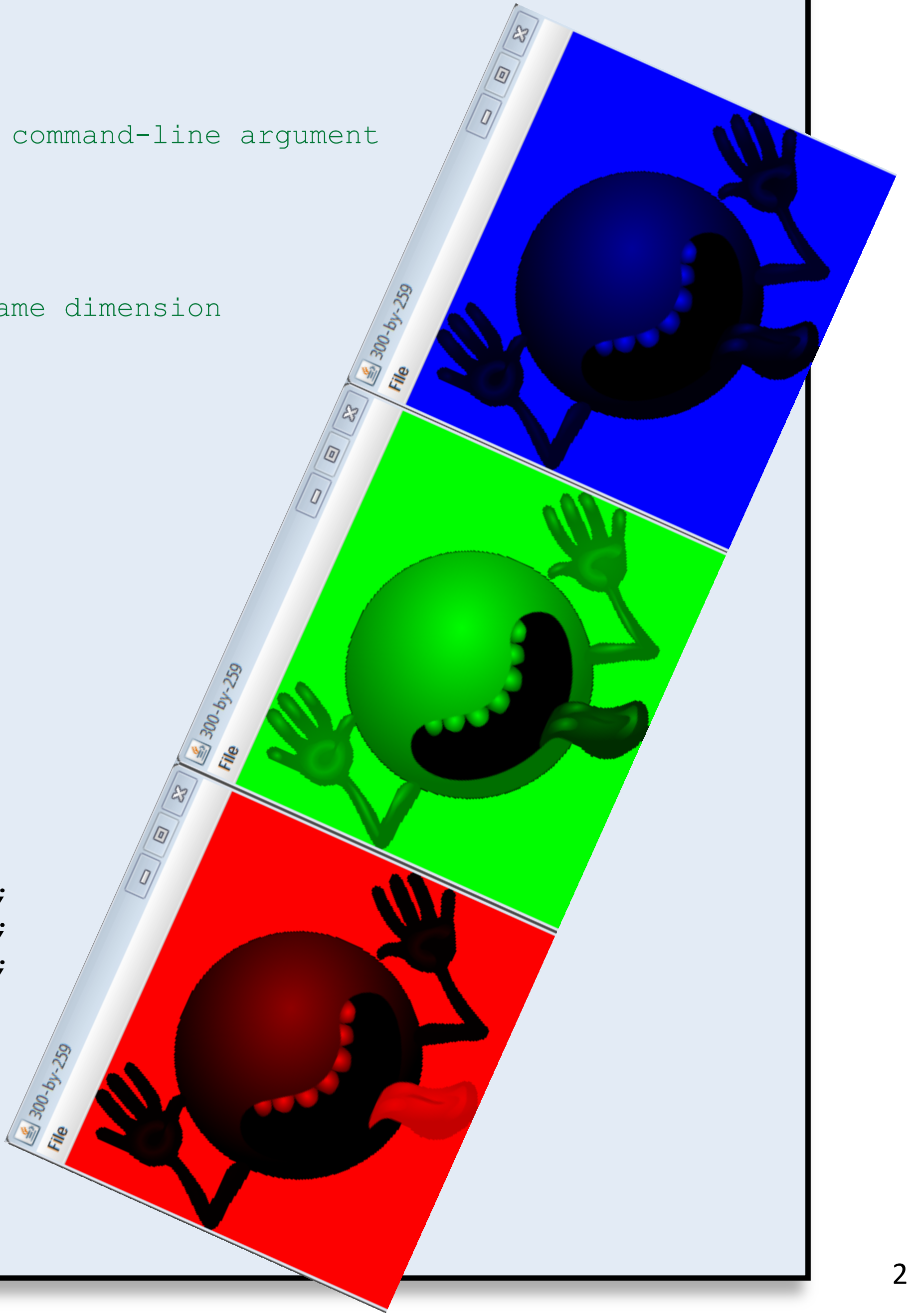
```java
public class FourChargeClient
{
   public static void main(String [] args)
   {
      // read in distance w from command line
      double w = Double.parseDouble(args[0]);
      // set up center of screen location
      double cx = 0.5;
      double cy = 0.5;

      // Construct four charges
      Charge c1 = new Charge(cx + w, cy, 1.0);    // East
      Charge c2 = new Charge(cx, cy - w, 1.0);    // South
      Charge c3 = new Charge(cx - w, cy, 1.0);    // West
      Charge c4 = new Charge(cx, cy + w, 1.0);    // North

      // Compute potentials at (.25, .5)
      double px = 0.25;
      double py = 0.5;
      double v1 = c1.potentialAt(px, py);
      double v2 = c2.potentialAt(px, py);
      double v3 = c3.potentialAt(px, py);
      double v4 = c4.potentialAt(px, py);

      // Output total potential
      double sum = v1 + v2 + v2 + v4;
      System.out.println("Potential = " + sum);
    }
}
```

API for object representing an image

```
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 i, int j)           // return the color of pixel (i,j)
  void set(int i, int j, Color c)  // set the color of pixel (i,j) to c
  void show()                      // display the image in a window
  void save(String filename)       // save the image to a file
```



```java
import java.awt.Color;
public class ColorSeparation
{
    public static void main(String [] args)
    {
        // read in the picture specified on the command-line argument
        Picture pic = new Picture(args[0]);
        int width = pic.width();
        int height = pic.height();

        // create three empty pictures of the same dimension
        Picture R = new Picture(width, height);
        Picture G = new Picture(width, height);
        Picture B = new Picture(width, height);

        // separate colors
        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                // color value of current pixel
                Color c = pic.get(x, y);

                int r = c.getRed();
                int g = c.getGreen();
                int b = c.getBlue();

                R.set(x, y, new Color(r, 0, 0));
                G.set(x, y, new Color(0, g, 0));
                B.set(x, y, new Color(0, 0, b));
            }
        }
        // display each one in its own window
        R.show();
        G.show();
        B.show();
    }
}
```



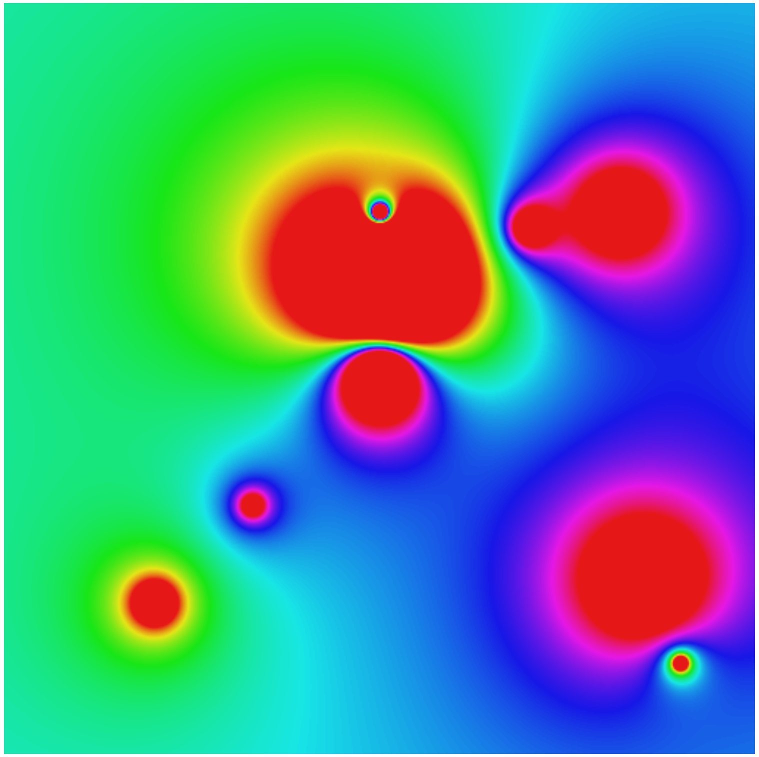
Using Charge and Picture

- Goal: read in point charges from a file, compute total potential in unit square

```
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   -5
```

charges.txt

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



Midterm

- Review on Monday, come with questions!
- Wednesday October 12th 3-5PM, Main 205 lab
- Note sheet:
  - One-sided
  - 8 ½ x 11
  - hand-written
- No other aids, electronic or otherwise
- Covered material:
  - Lecture
  - Head First Java, chapter 1 - 5
  - Lab