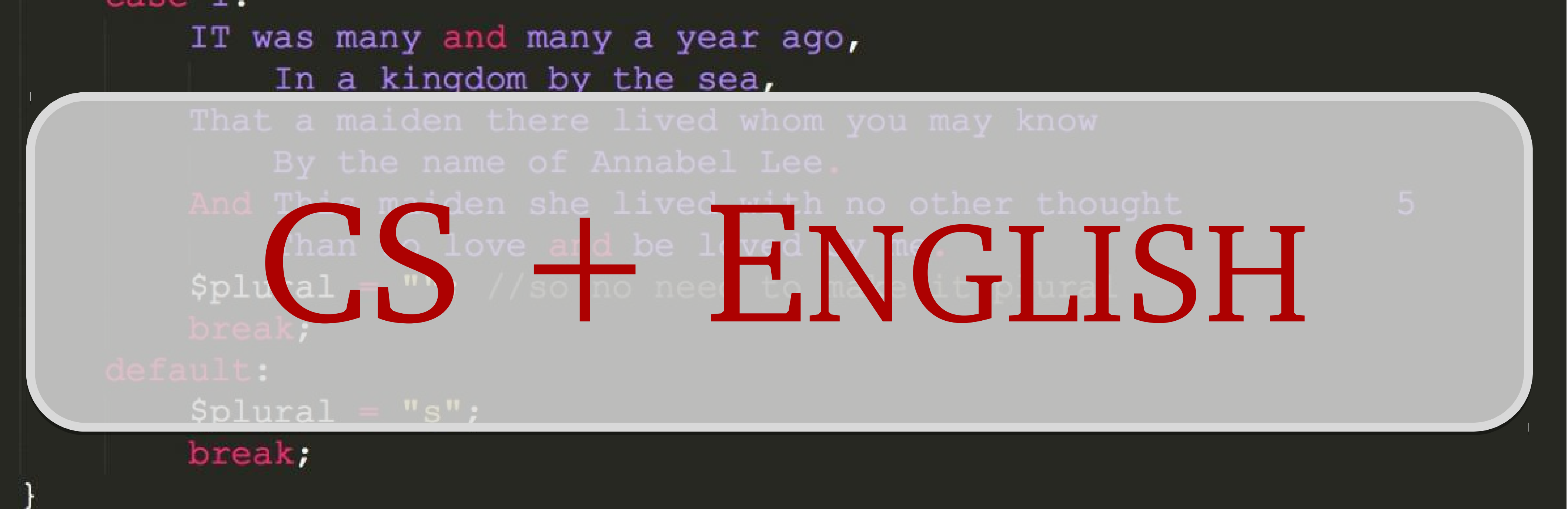
Parameters and Objects



```
        IT was many and many a year ago,
            In a kingdom by the sea,
        That a maiden there lived whom you may know
            By the name of Annabel Lee.
        And This maiden she lived with no other thought
            Than to love and be loved by me.
        $plural = ""; //so no need to make it plural
        break;
    default:
        $plural = "s";
        break;
}
```

CS + English

Enrich your computer science skills with the understanding of human experiences, critical thinking, and creativity taught in English.

More info: english.stanford.edu/csenglish

Questions? Email kdooling@stanford.edu.

Announcements

- Assignment 3 due at 3:15PM today.
  - Due on Wednesday at 3:15PM with one late period and Friday at 3:15 with two.

Assignment 4 Demo

Breakout!

- Due next Monday, February 9.
- **Start Early!**
  - There is a nice breakdown of the required tasks suggested in the handout.
  - This program is not as hard to write as it may seem.
- **Have Fun!**
  - There are a *lot* of fun extensions you can add onto the basic functionality.
  - We love giving extra credit on this one. ^_^

Midterm Logistics

- First midterm is **Tuesday, February 10** from 7PM – 10PM.
  - Room assignments TBA.
- Closed-book, closed-computer, limited notes.
  - You can have a double-sided 8.5” × 11” sheet of notes with you.
  - We'll provide a reference of the important methods we've seen so far.
- Covers material up through and including Wednesday's lecture on string processing.

Practice Exam

- We will be holding a practice midterm this Wednesday evening from 7PM – 10PM in Cemex Auditorium.
- Completely optional, but an excellent way to review the material and get practice writing code on paper.
- Can't make it? We'll post the exam and solutions up on the course website about 15 minutes after the practice exam starts.

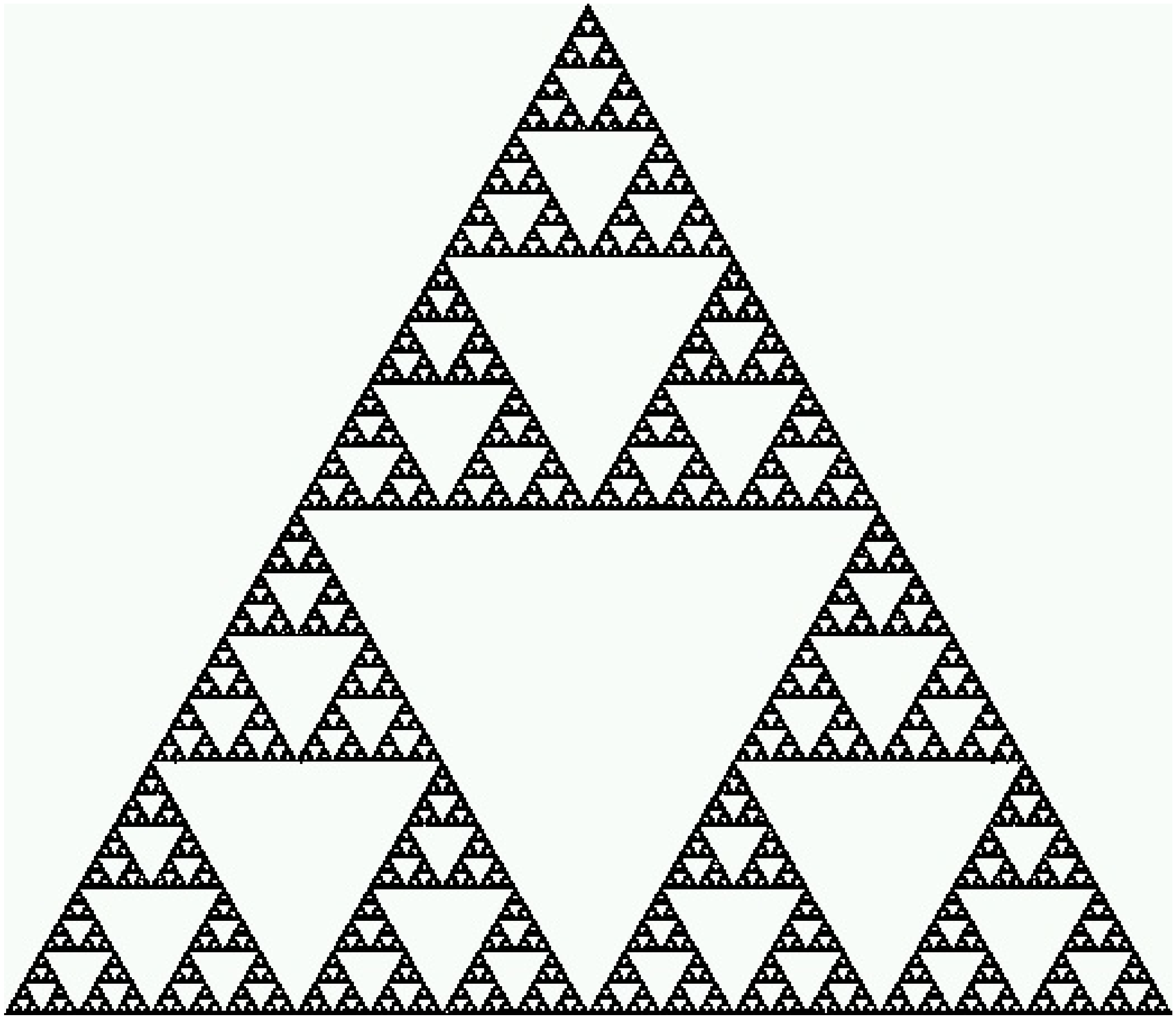
Let's Get Started!

The Chaos Game

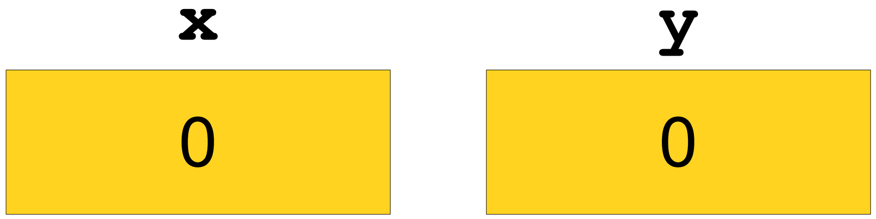
The Chaos Game

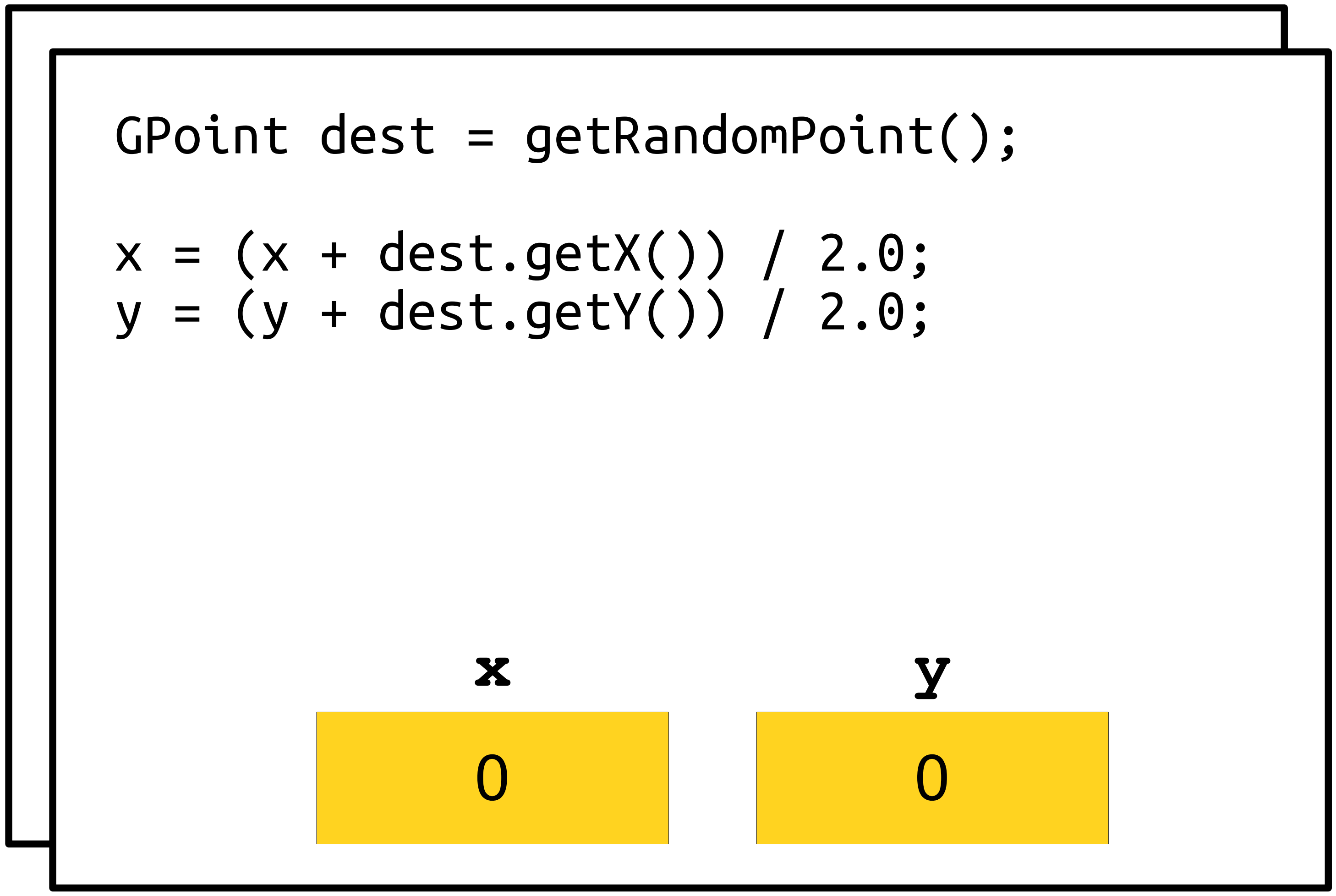
- Pick any three points.
- Starting at any of the points:
  - Choose one of the three points randomly.
  - Move halfway from your current location to the chosen point.
  - Draw a dot at your current location.
  - Repeat.

Sierpinski Triangle

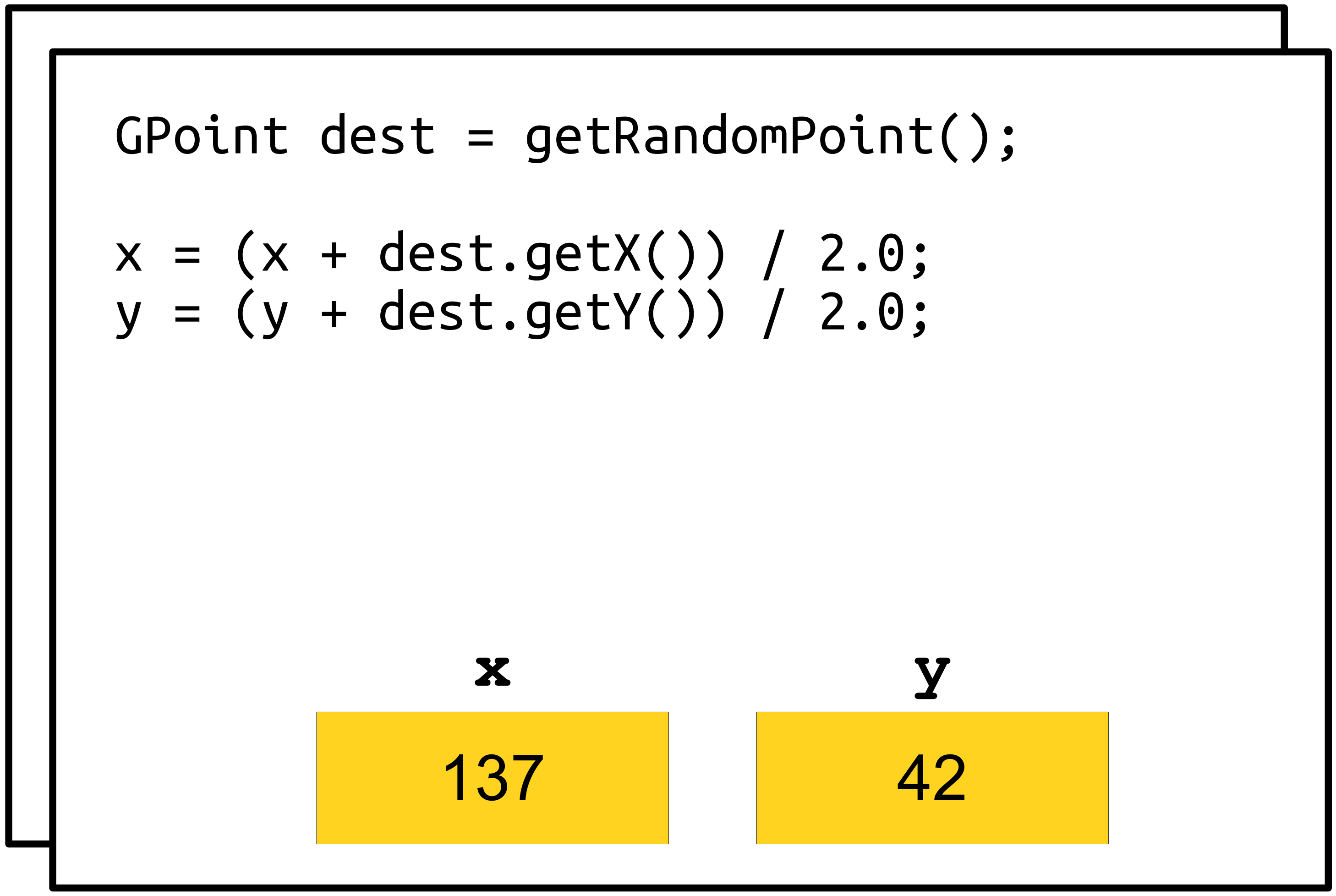


```
double x = 0;
double y = 0;

while (true) {
    moveRandomly(x, y);
    plotPixel(x, y);
}
```



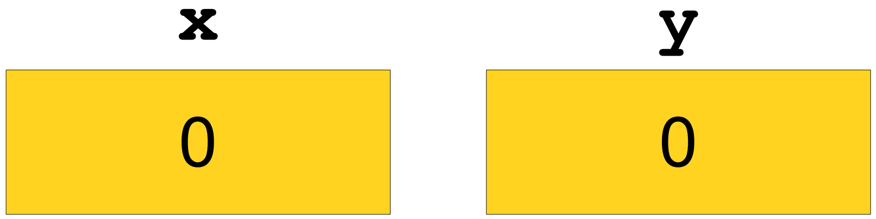


```
GPoint dest = getRandomPoint();

x = (x + dest.getX()) / 2.0;
y = (y + dest.getY()) / 2.0;
```

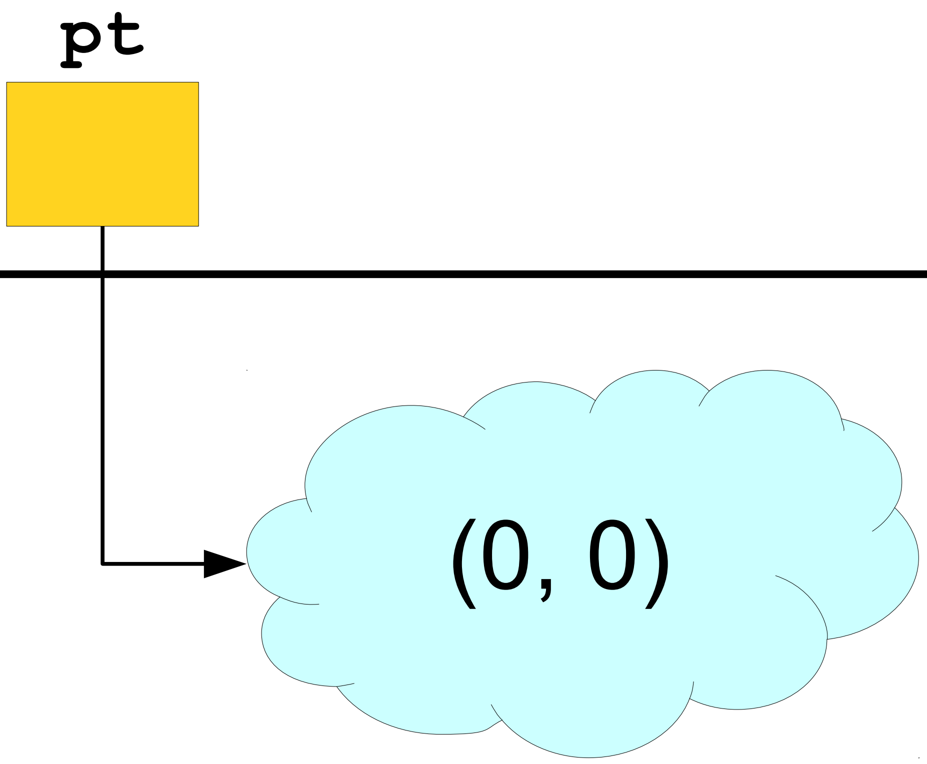


```
GPoint dest = getRandomPoint();

x = (x + dest.getX()) / 2.0;
y = (y + dest.getY()) / 2.0;
```

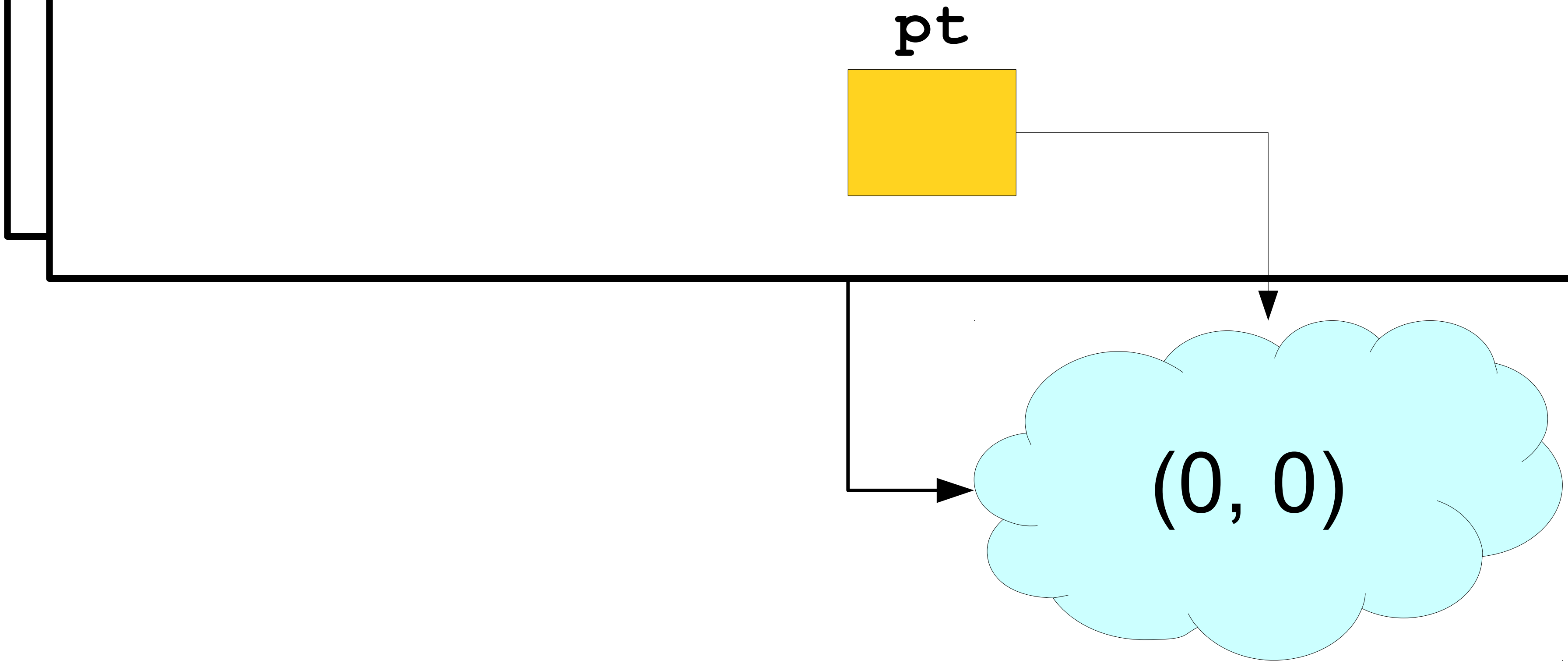
```
double x = 0;
double y = 0;

while (true) {
    moveRandomly(x, y);
    plotPixel(x, y);
}
```



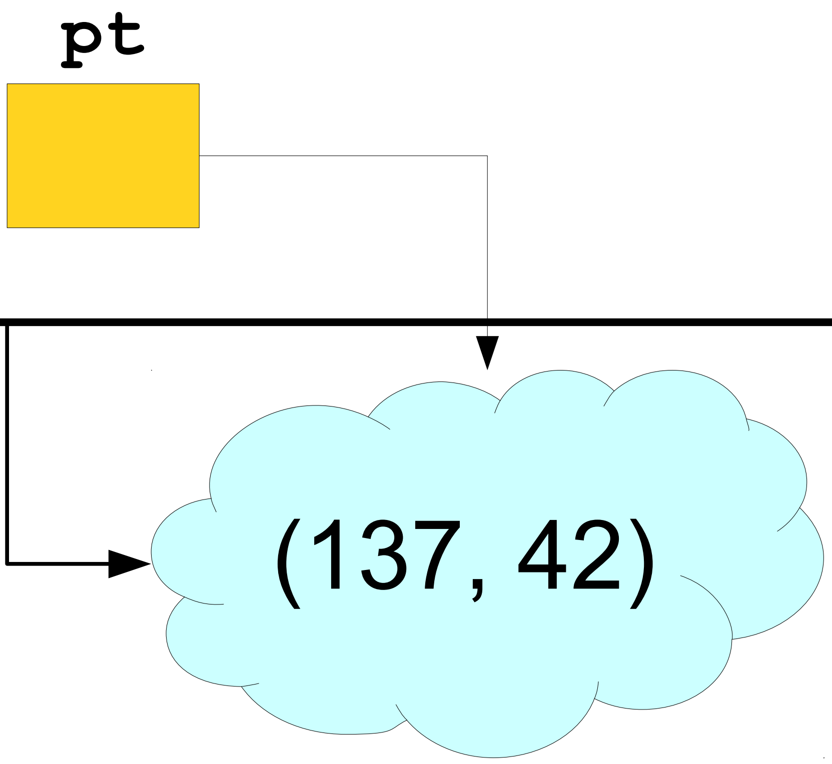
```
GPoint pt = new GPoint(0, 0);

while (true) {
    moveRandomly(pt);
    plotPixel(pt.getX(), pt.getY());
}
```



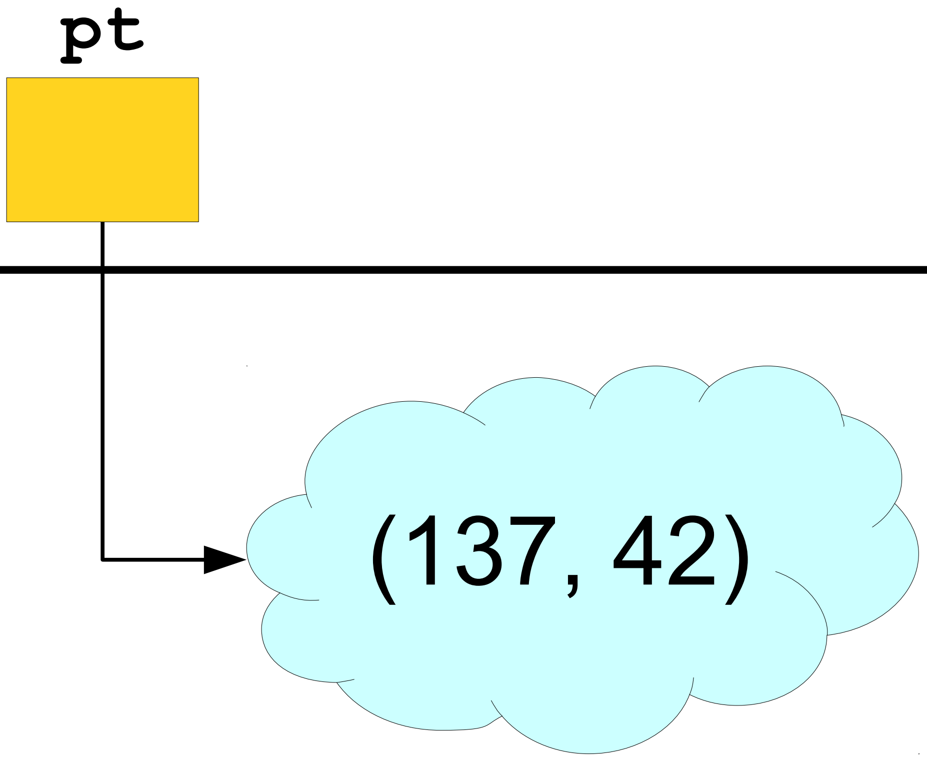
```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0;

pt.setLocation(newX, newY);
```



```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0;

pt.setLocation(newX, newY);
```



```
GPoint pt = new GPoint(0, 0);

while (true) {
    moveRandomly(pt);
    plotPixel(pt.getX(), pt.getY());
}
```

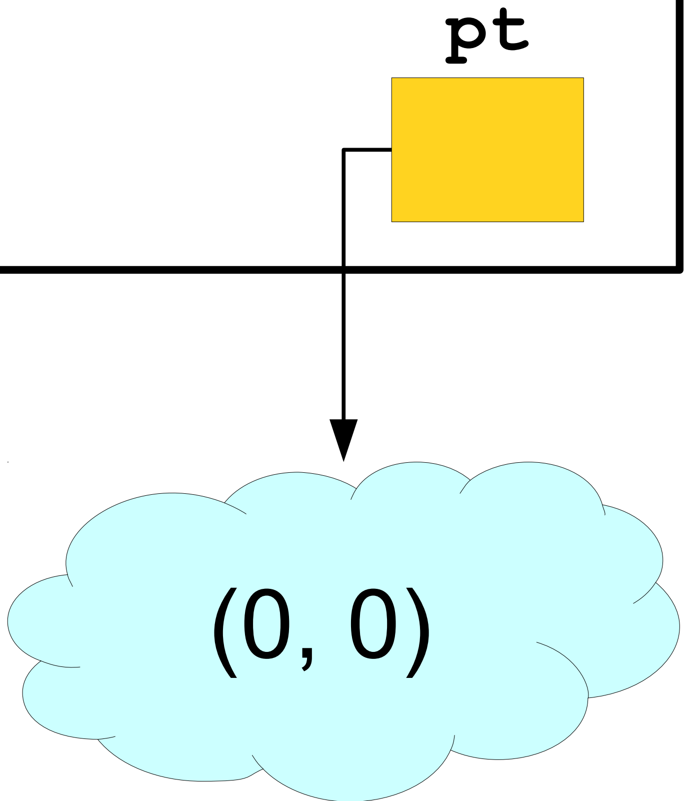


Parameter Passing

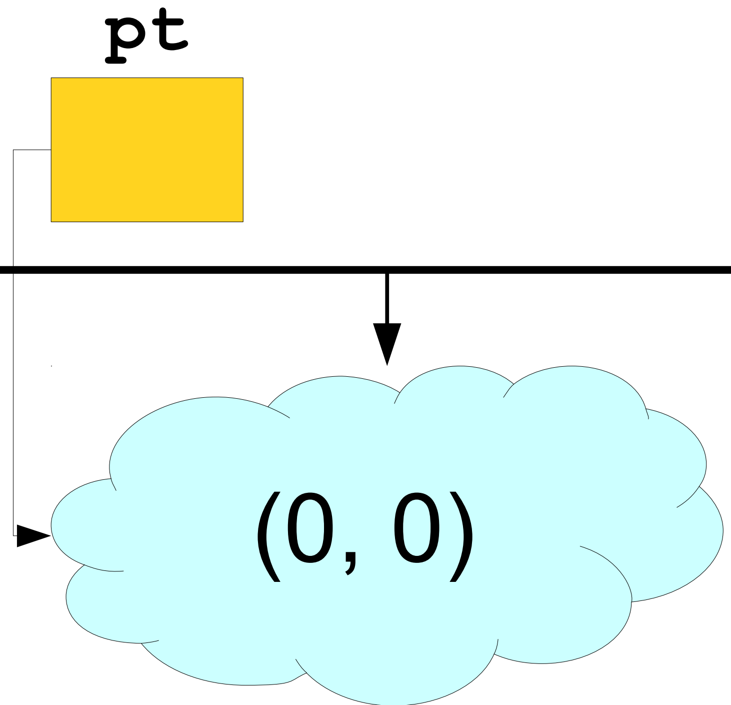
- All parameters in Java are passed by value.
- In Java, variables of primitive type (**`int`**, **`double`**, etc.) store actual values.
- In Java, variables of *object* type (`GOval`, `GRect`, etc.) don't actually store those objects. They store *references* to those objects.
  - They “point” to where the object really is.

Another Variation

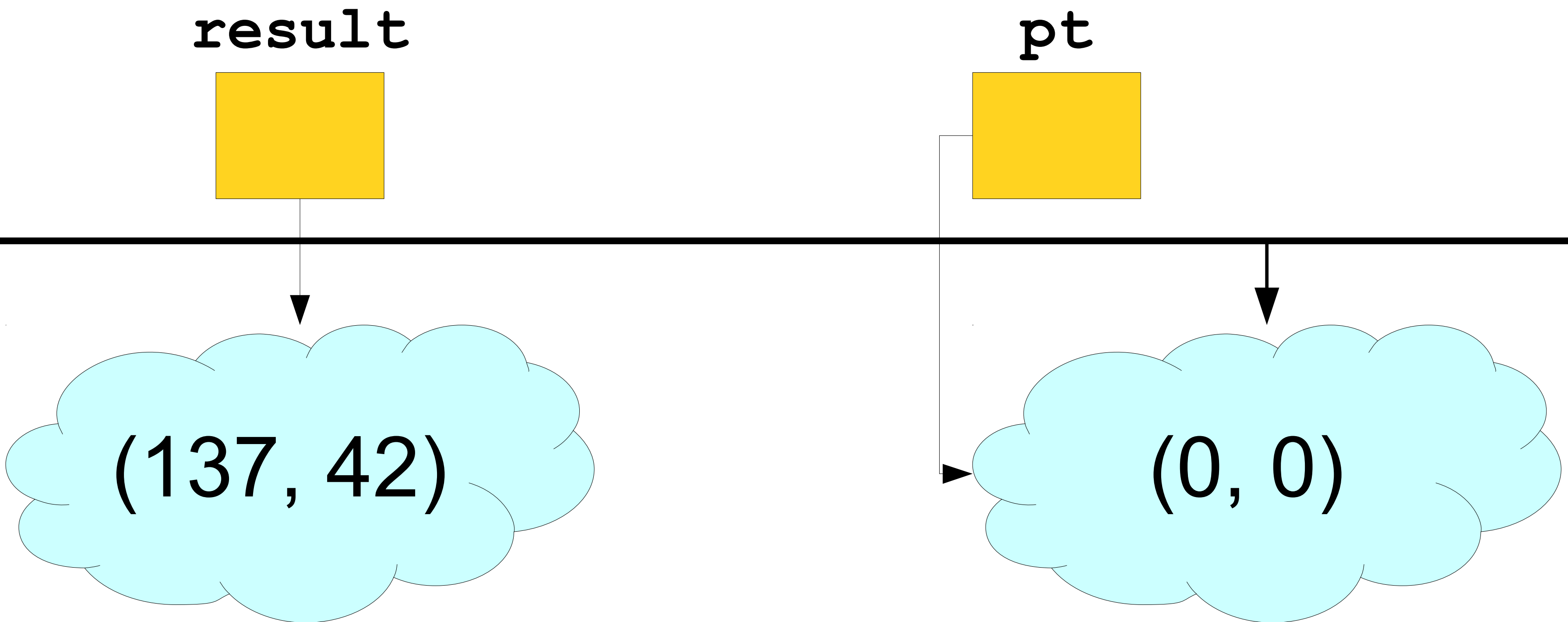
```
GPoint pt = new GPoint(0, 0);

while (true) {
    moveRandomly(pt);
    plotPixel(pt);
}
```



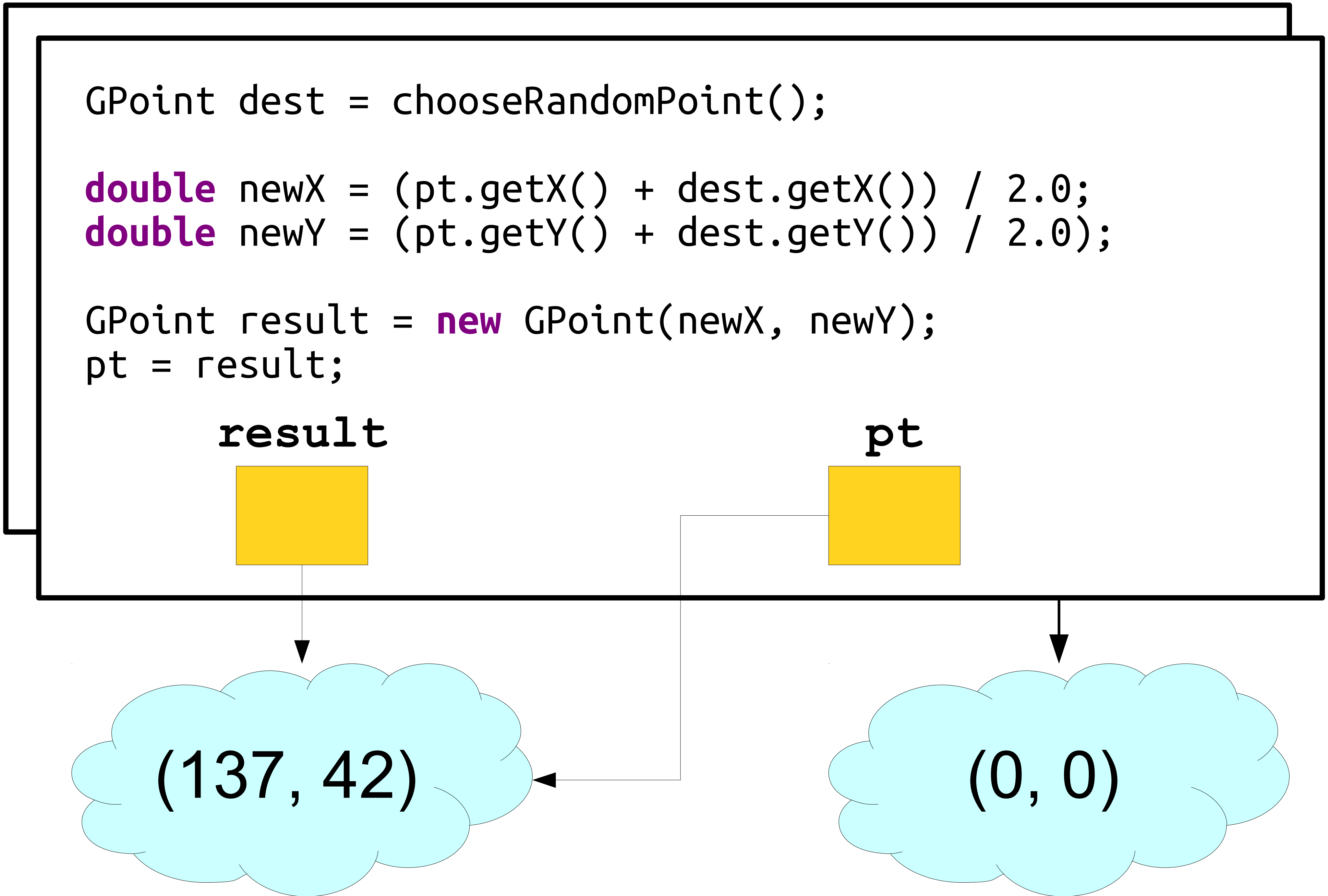
```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0);

GPoint result = new GPoint(newX, newY);
pt = result;
```



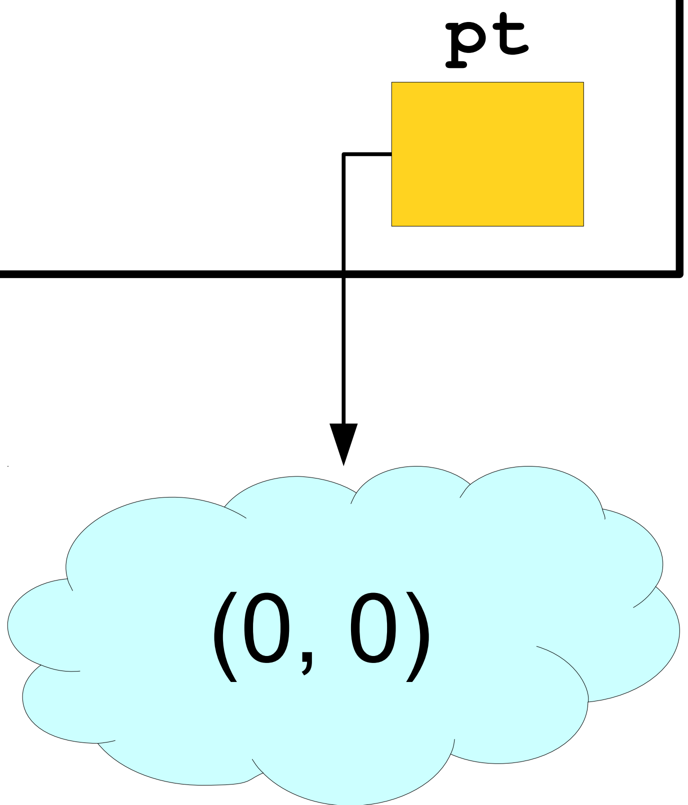
```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0);

GPoint result = new GPoint(newX, newY);
pt = result;
```



```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0);

GPoint result = new GPoint(newX, newY);
pt = result;
```



```
GPoint pt = new GPoint(0, 0);

while (true) {
    moveRandomly(pt);
    plotPixel(pt);
}
```

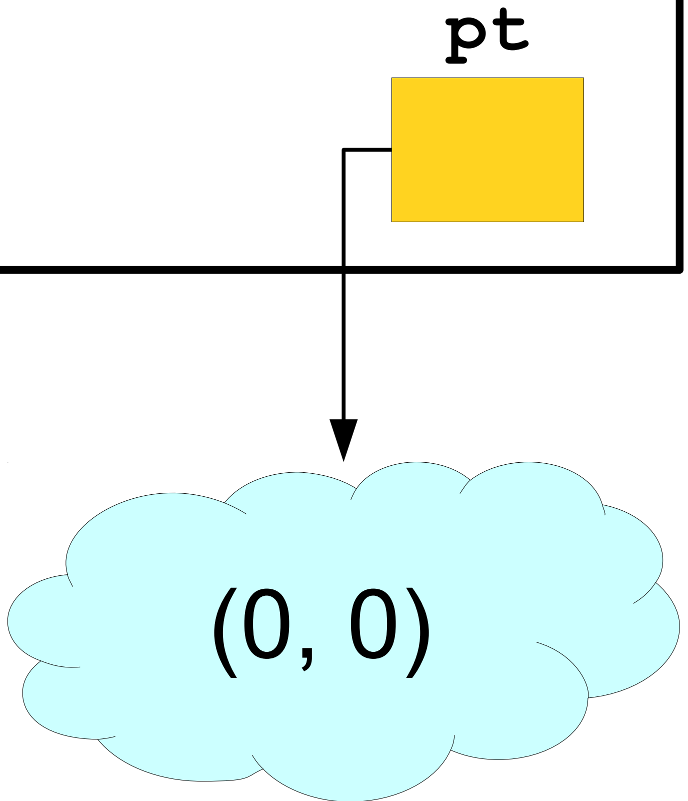


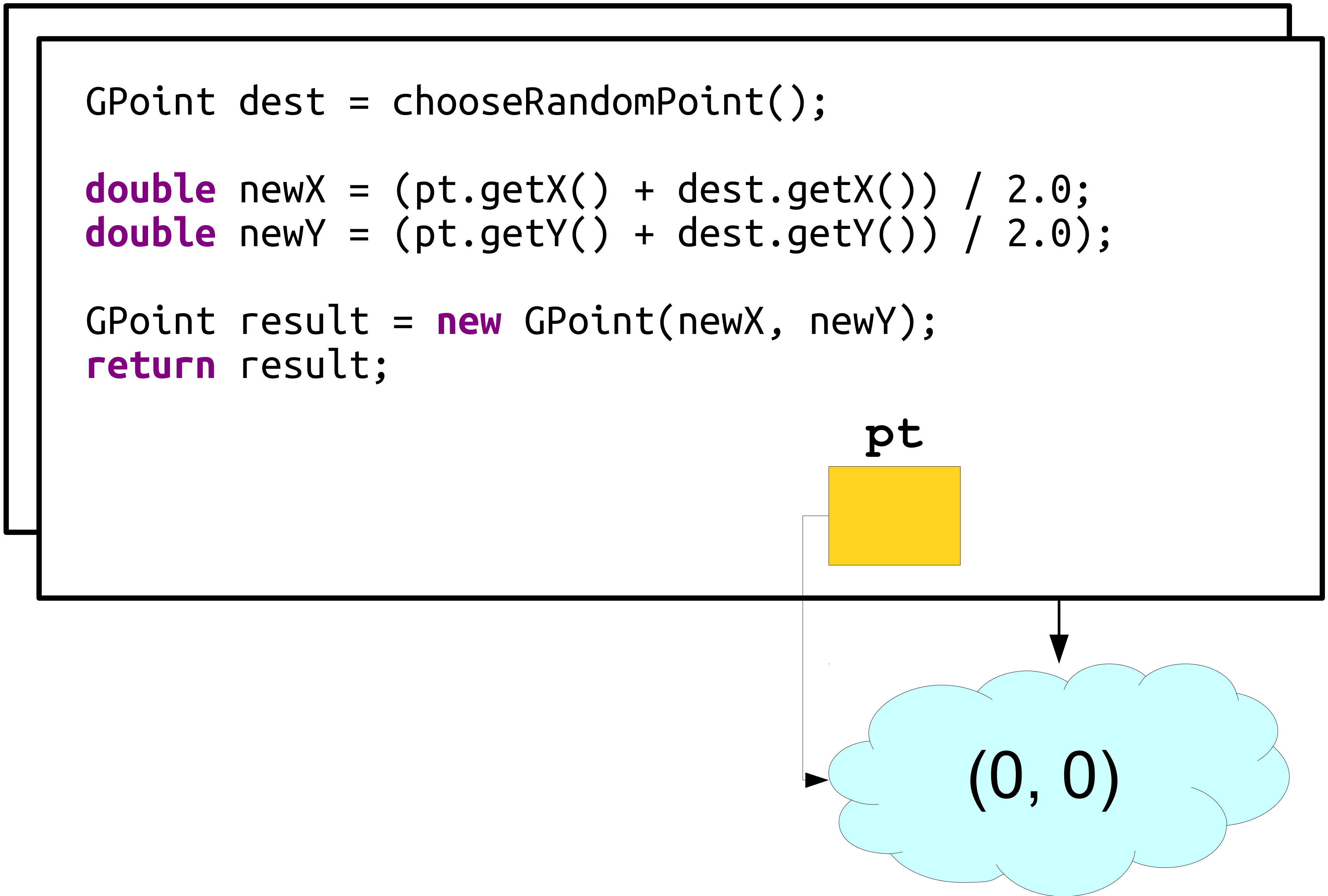
A Nuance

- If you pass an object into a method, that method can change properties of the object passed in.
  - The caller can then see these changes.
- If you pass an object into a method, that method cannot change *which object* is being referred to.
  - The caller will always end up referring to the same object, though the properties of that object might have changed.

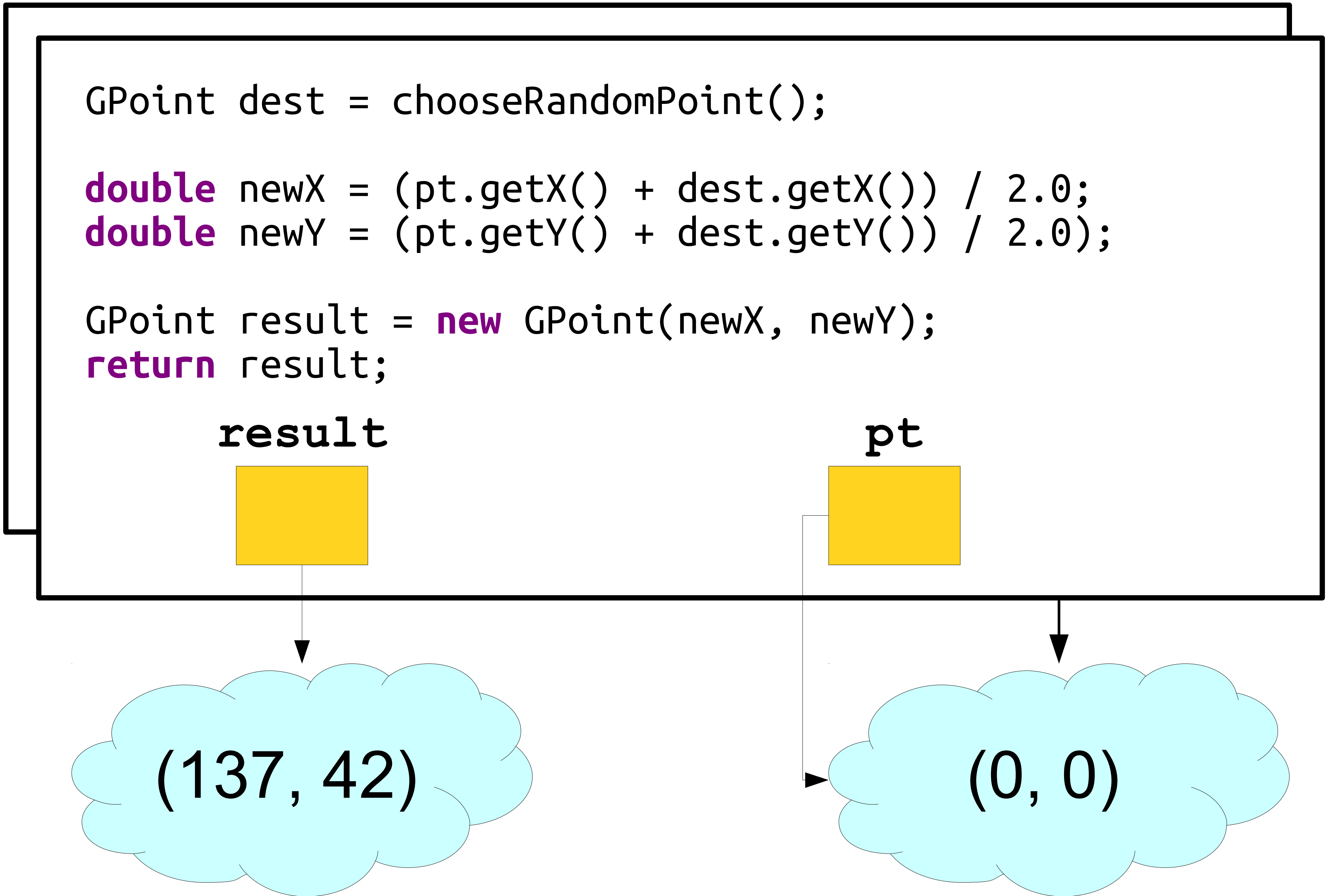
One Final Approach…

```
GPoint pt = new GPoint(0, 0);

while (true) {
    pt = moveRandomly(pt);
    plotPixel(pt);
}
```

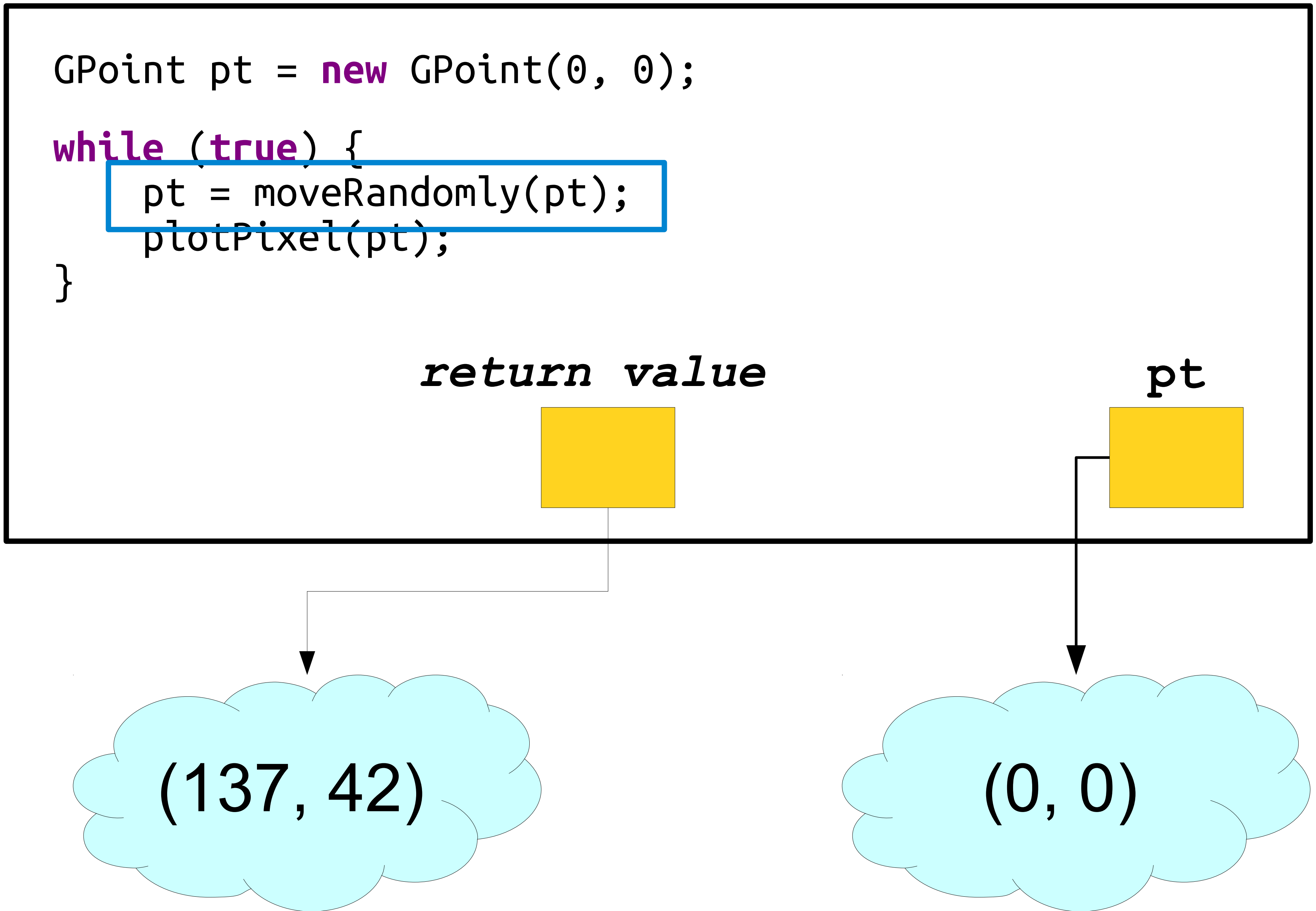




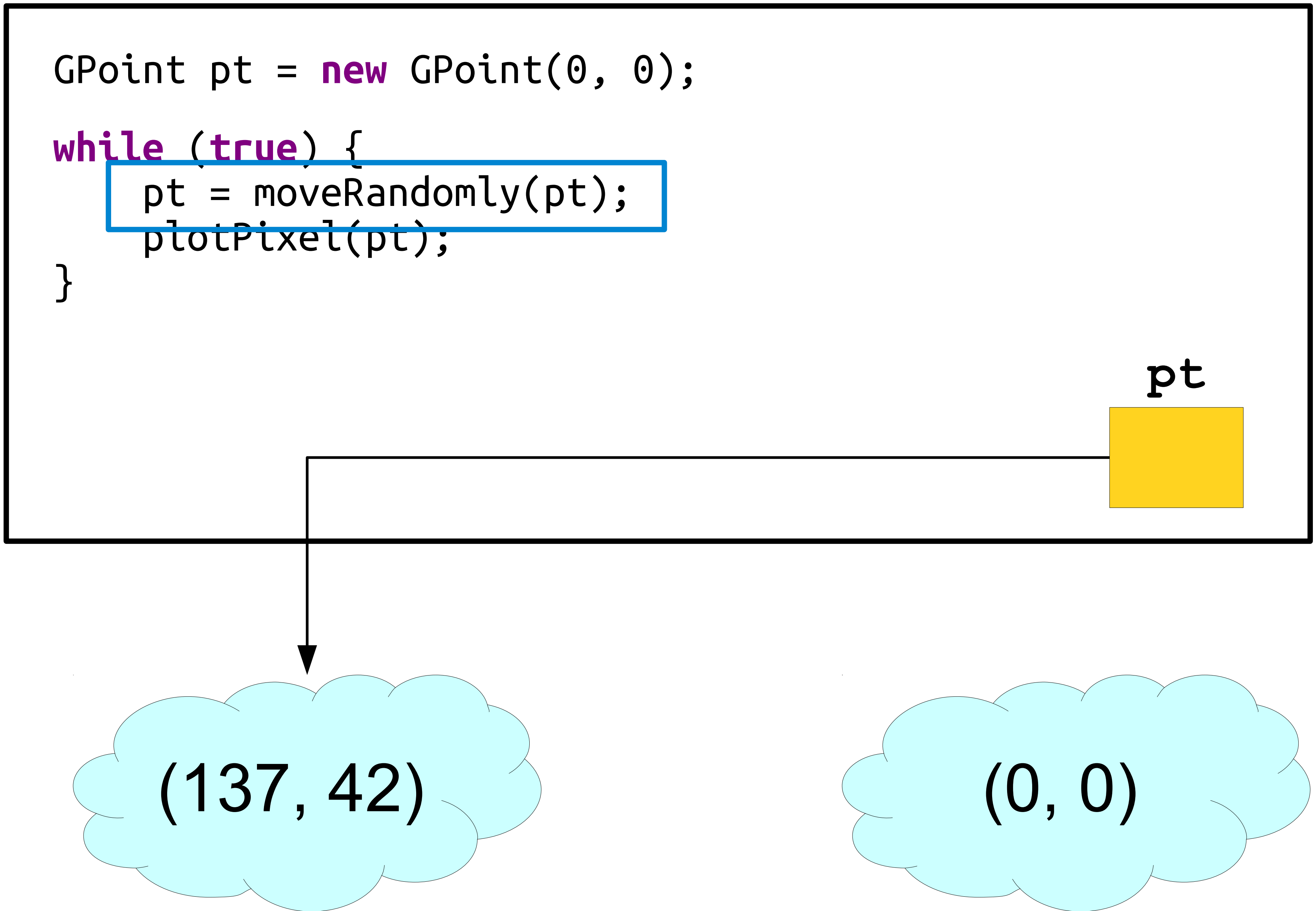
```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0);

GPoint result = new GPoint(newX, newY);
return result;
```



```
GPoint dest = chooseRandomPoint();

double newX = (pt.getX() + dest.getX()) / 2.0;
double newY = (pt.getY() + dest.getY()) / 2.0);

GPoint result = new GPoint(newX, newY);
return result;
```

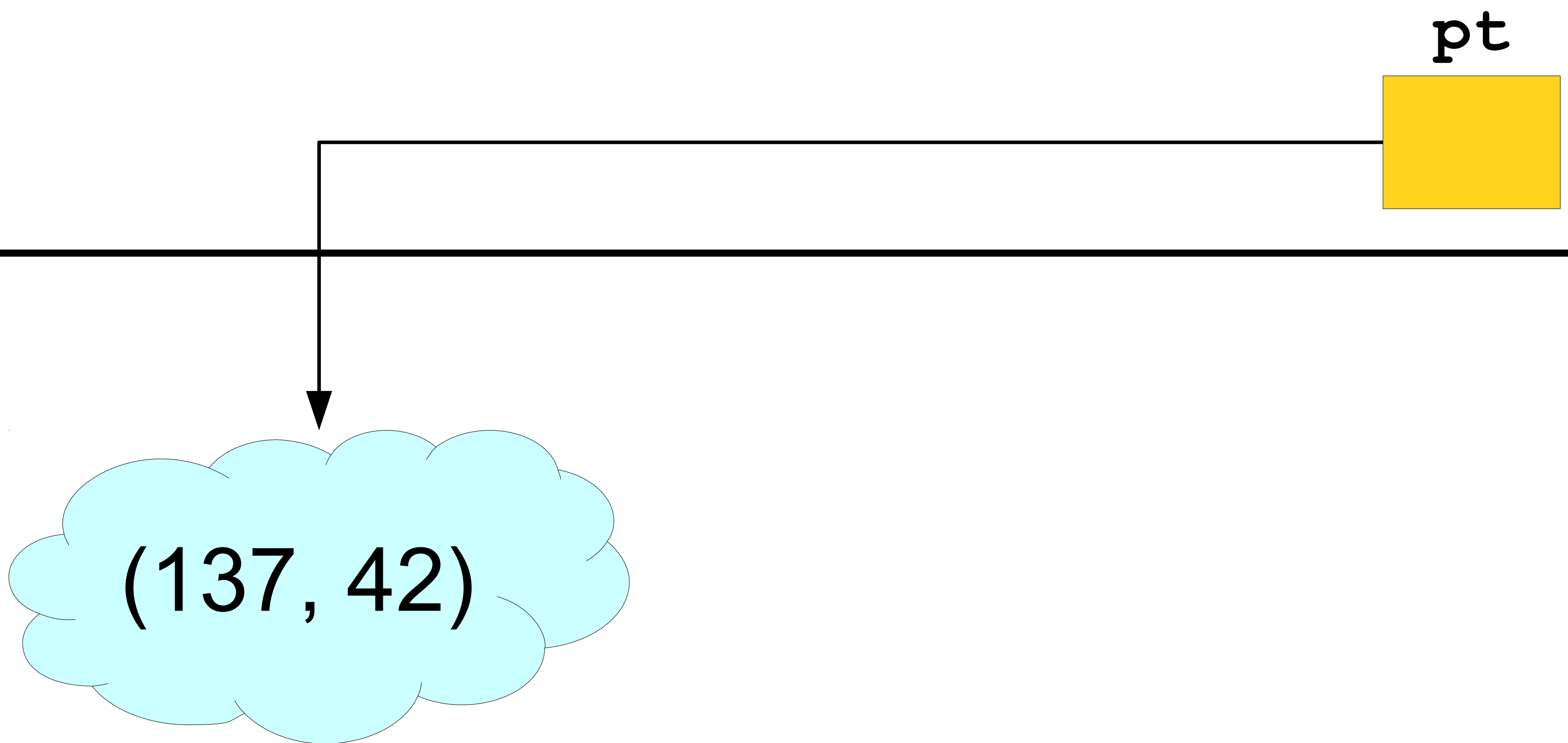


```
GPoint pt = new GPoint(0, 0);

while (true) {
    pt = moveRandomly(pt);
    plotPixel(pt);
}
```



```
GPoint pt = new GPoint(0, 0);

while (true) {
    pt = moveRandomly(pt);
    plotPixel(pt);
}
```

```
GPoint pt = new GPoint(0, 0);

while (true) {
    pt = moveRandomly(pt);
    plotPixel(pt);
}
```



Summary

- Primitive types are passed by value.
  - The callee gets a *copy* of the value.
  - The callee can change that *copy*, but cannot change the original.
- Object references are passed by value.
  - The callee gets a copy of the *reference*, not a copy of the *object*.
  - The callee can change the object, but cannot change *which* object is referred to.

Text Processing

“How Revolutionary Tools Cracked a 1700s Code”

http://www.nytimes.com/2011/10/25/science/25code.html

|oghnʌmórʌuvziĝqiymż|érzueengÿra=rzp+hmhmżÿ|ûʌx
zufsprzinhʌurhż|nômÿxlδʌ+b+zi|ûs▮ʌhx+wpipen|npppr
mππ|nxîδuʌhgÿczπhjδδʌhpnu|òphrʌmòsʌuh|ĩôtnno=rzx
hé|zûn:scq▯ʌfπĩ3t∠Λnzuvnmzÿπpʌurzh|w∠Λ|ppδcpp
ċpδuozp|ni|g|gÿgôpδêtnδcpnδδîiʌbÿ+pcp|ufxnzgt3nppuh
=r|ijâsbtn|:sê|lsûÿxuezig:zcûxʌgmizzpr=+pπôsxnhgπÿ
πpʌfznâgrêsnʌâshhrzlpi=|szut∠Λajnnʌizgzn|djnshʌoe

„ pûoogziô|û+cnzxixkmic:xÿʌδ+ippphrzzʌsʌc
„ xnzfs=hpmnδudsâiʌub+znrpzgsxδ=rʌûjfiûzôgub+hhcu
„ xnmxîcʌpʌuv◇Φδ|cishʌzûsnn:ôjme.

zprvπisÿδisîrzâ∠Λbcδ|xʌfnh+δ|ôvzhn:wπurxp|ic:u
c=gπbcîhjcirx|nf+=δfppo=rzbghpzû+lπêrnpjû+ʌniu∠Λ
zpih ner tzêgî+tpi=|m[illegible]izn|xrzzxâb∠Λôjnnʌcxmzfπug∠Λh=
ʌhpδipÿshôʌat+pδêʌszâsfπÿjnδisuzô∠Λlzupnhgzπp
ʌuvzn+nʌf|ác:ir:iʌôszh=|u+iʌxzubmôrʌuf:

„ πpszûjcî=pʌʌûnprzzzsb|ác:nznê|ôjmspÿpudgnô
„ +hc|ghgzîs|mpc|cs=gêhsuhπôj|cδpcipuw◇Φo=mzbÿml
znr

„ znrâspπ=|rÿpʌnδuc◇Δf|nprxzpsh=pueohûjôpδubδôr|n
„ pisʌh|iomirpi▯rôógl|=mzlhguihrtzpzâsxjôshʌfnl:pgêstpr=
„ zjnδôrgûiguphhrud∠Λxπâsgcδpênpuc◇Φzxrδûr:u
„ b+p|:ẽc.

Vzâsxtpi:mijhpûcâhsêʌgihmδnc|zôn:pÿrgzzgufδpm:
sâipufzîrw∠Λc=mπvm∠Λaxnzpʌhnh+rzh|fmphsên▯i
mâcpû|bÿ|rπnînhÿ|ôsiûmδaghzû+g=s|drxrδszâjg◇Φo
prδcgjêʌbzh|x|ppgh+|âlnz|gxg|xrudxâhûÿ+upsʌn|iʌ=ʌÿ
òruamiôrh=psπugmnhsubδâj|pigzfπzpn|nêxûc|a=nzpδnz
pʌhnh+rznâxprcpxpnÿ▯uêimg=gπvnrzîs|gs=b|ônnprgnhp
sipʌlênfpi:sûfδ|ipln=|xsÿpguimppmfzʌpnh+δnâp|ghc
ċudnrmû|uzucxcpδixrπzzôifzisnδÿêjuzâ∠Λnpxfcÿp
▯ʌnzp+nʌgzizcΔm.

Pimâooλîsmʌnʌ=cb

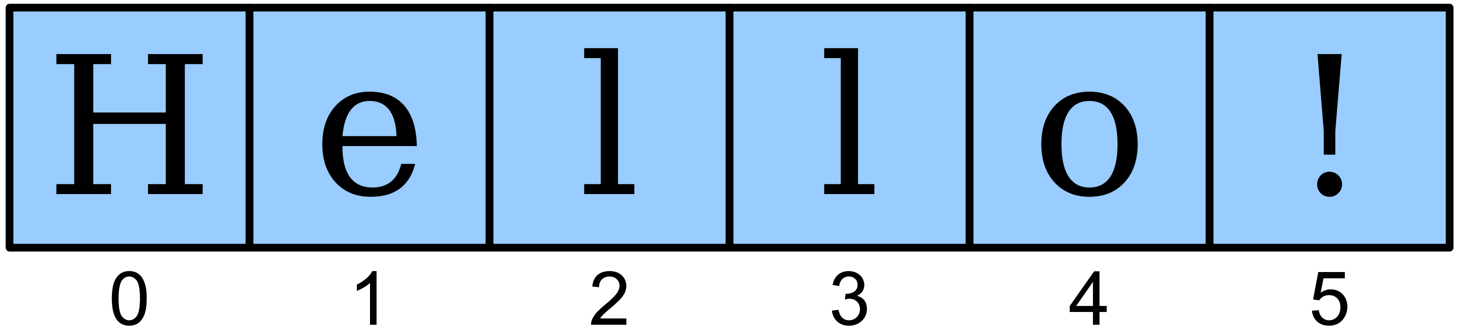
δêr:sÿpudrûinpp|v∠Λ.

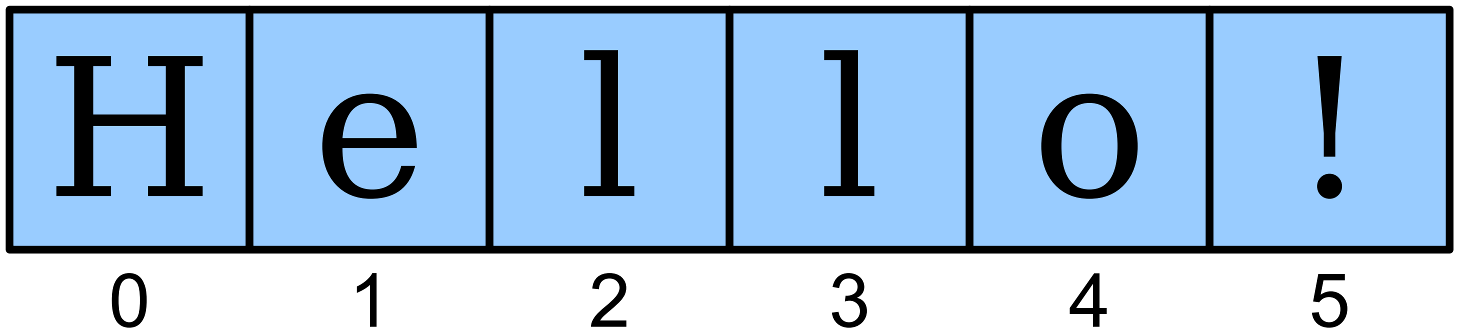
Hôigg∠Λbmiszfmnpʌc|ʌdmôhcpπhz=jpcûrδpr:ʌo
zh▯pxrg=n|pzêδfÿ|:jâ|b|hzʌiδmcp|ugxrzxfrênpucδan:âi|ôg

A ***string*** is a sequence of characters.

http://upload.wikimedia.org/wikipedia/commons/e/ea/Capra_ibex_nubiana_near_Mitzpe_Ramon_in_summer_2011_%284%29.JPG

H e l l o !


H
e
l
l
o
!
0
1
2
3
4
5



string`.charAt(`***index***`)`

The Data Type **char**

- The primitive type **char** represents a single character or glyph.
- Some examples:

```
char letterA = 'A';
char plus    = '+'
char zero    = '0';
char space   = ' ';
char newLine = '\n'; // An escape sequence
```