Objects and Strings

Friday Four Square!
Outside Gates, 4:15PM

An Interesting Listen

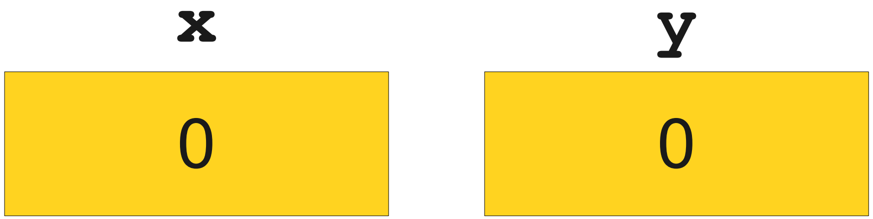
RadioLab: “Talking to Machines”
http://www.radiolab.org/2011/may/31/

The Chaos Game Revisited

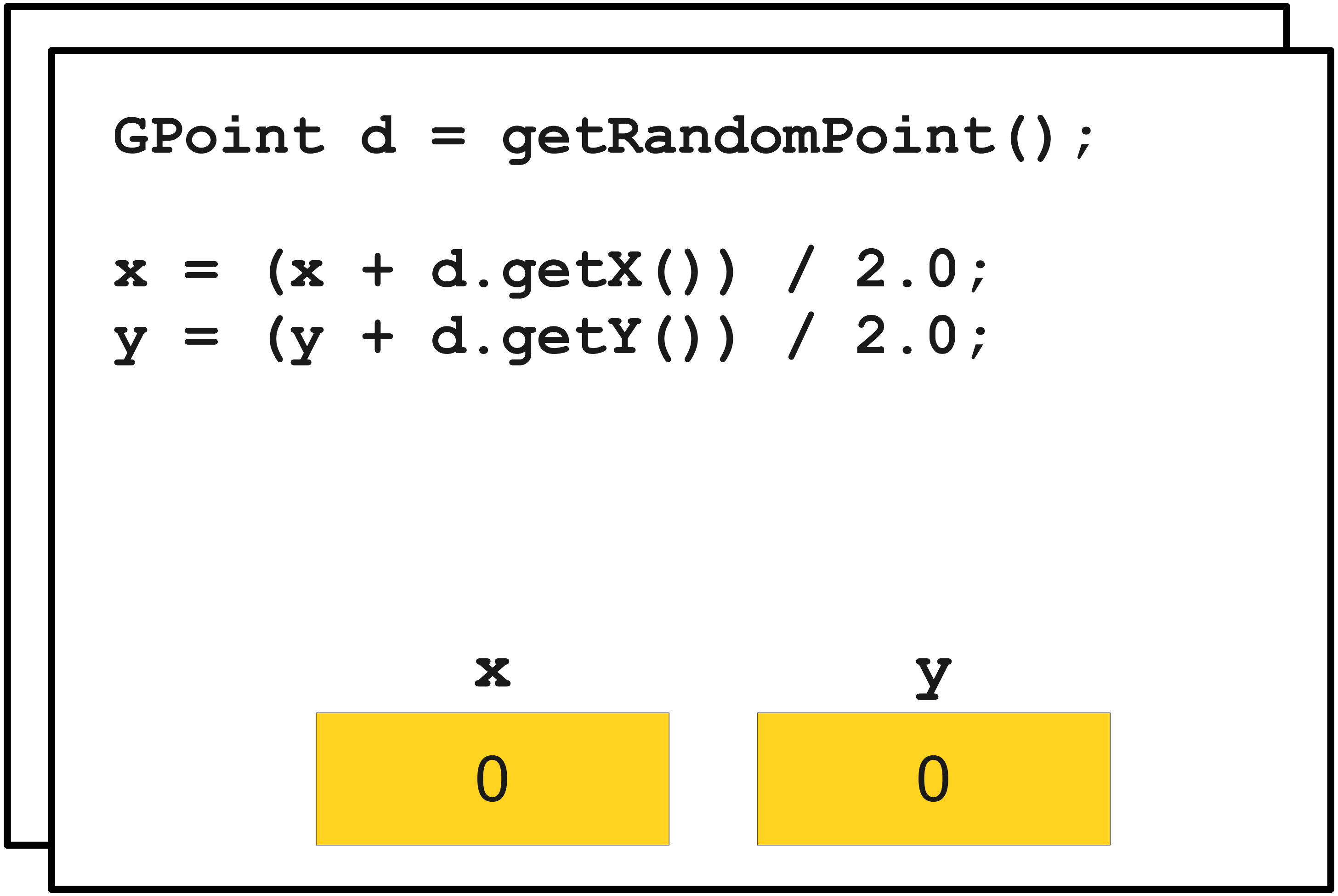
What Just Happened?

Pass by Value

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

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

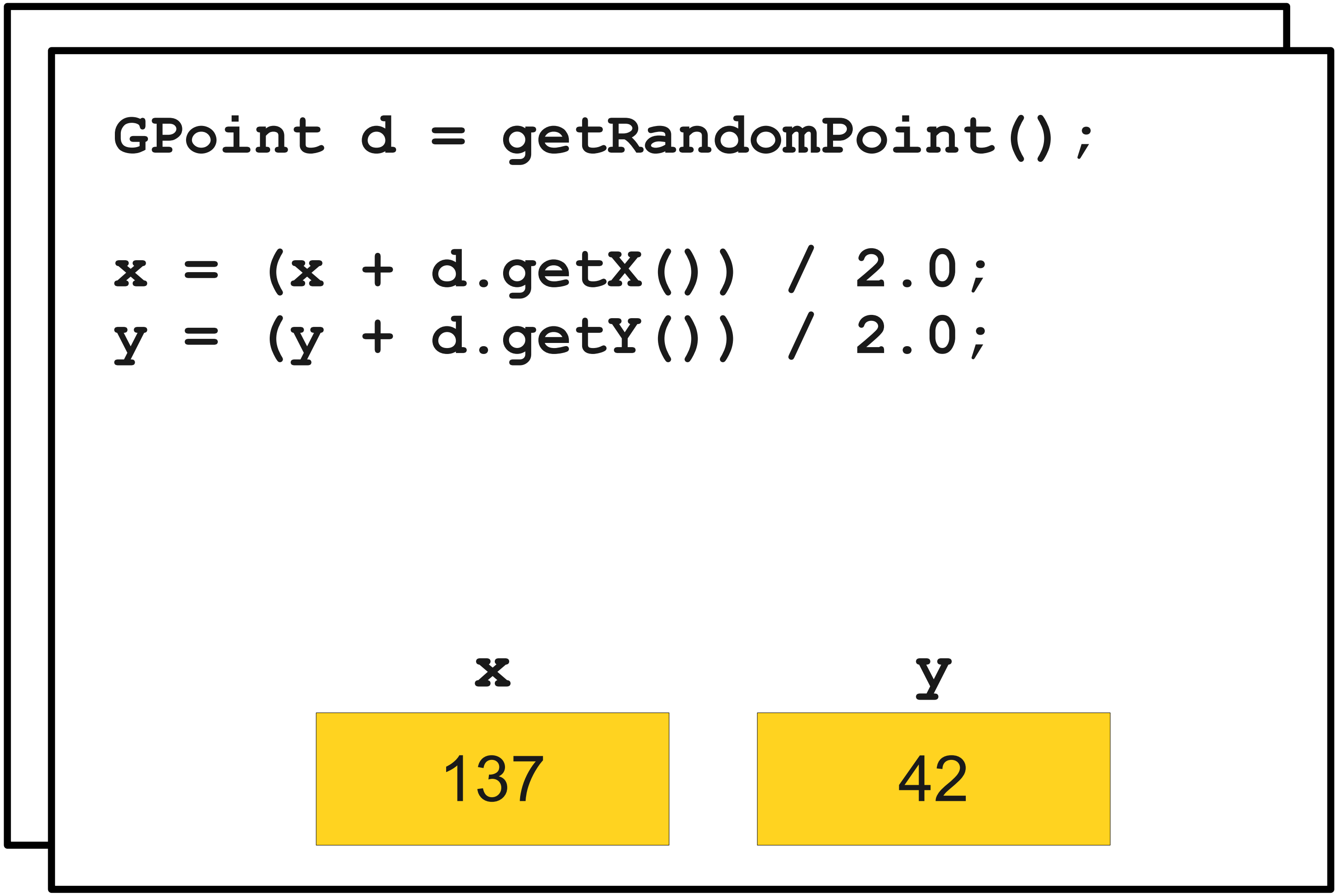


Pass by Value



```
GPoint d = getRandomPoint();

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

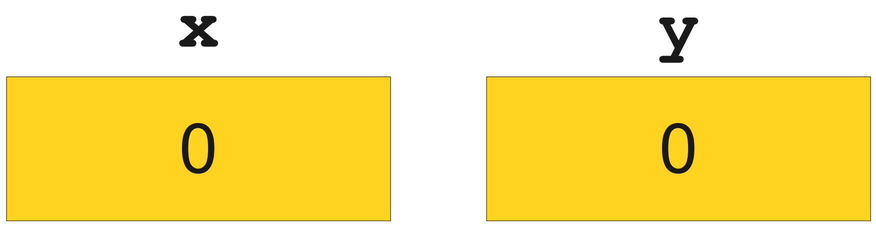
Pass by Value



```
GPoint d = getRandomPoint();

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

Pass by Value

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

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



x

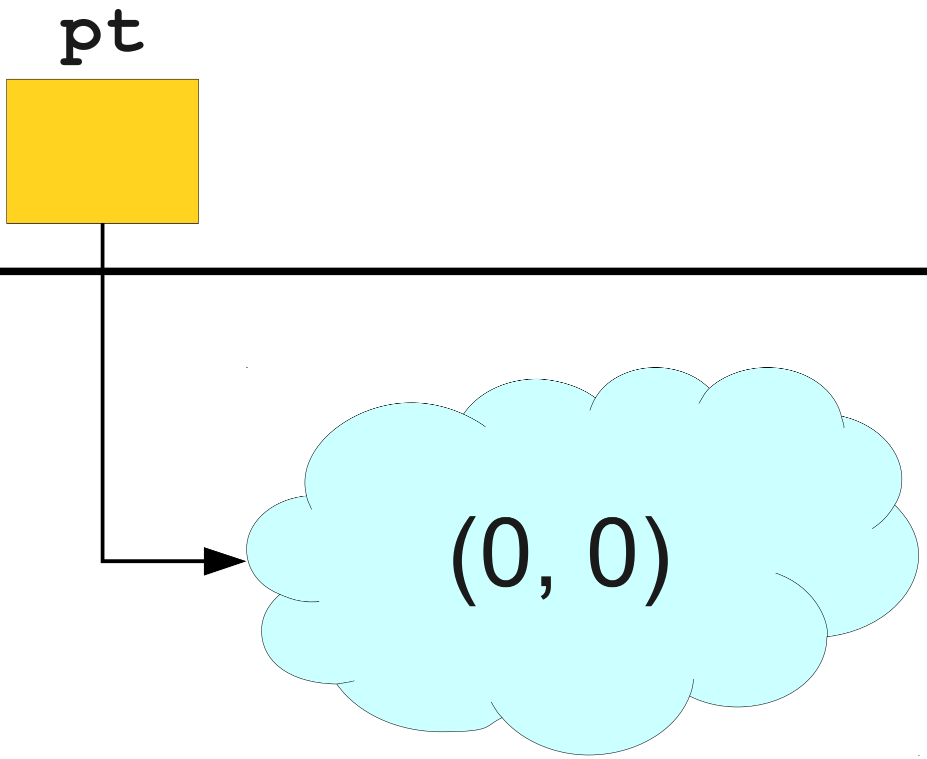
0

y

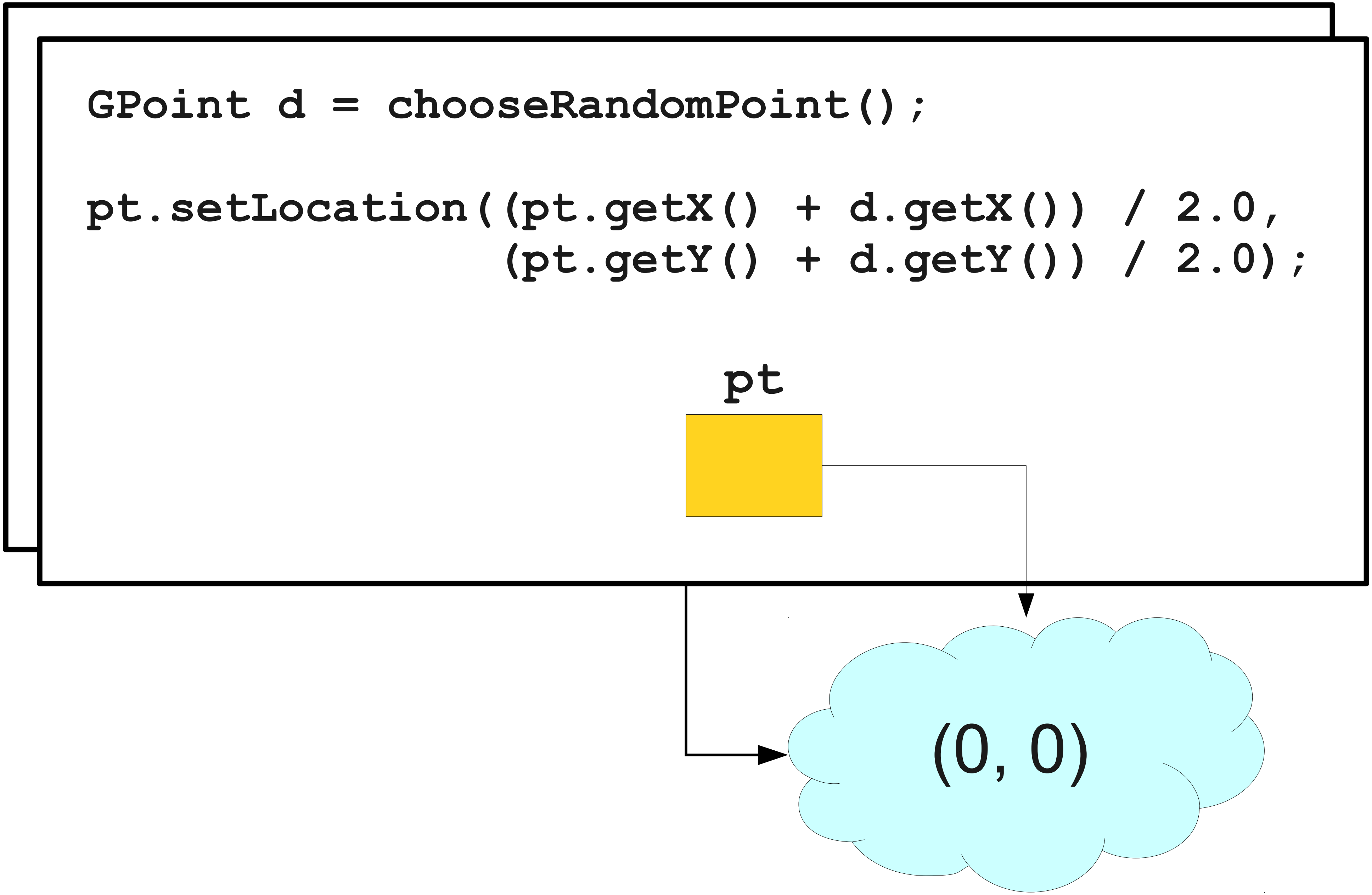
0

Pass by Reference

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

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



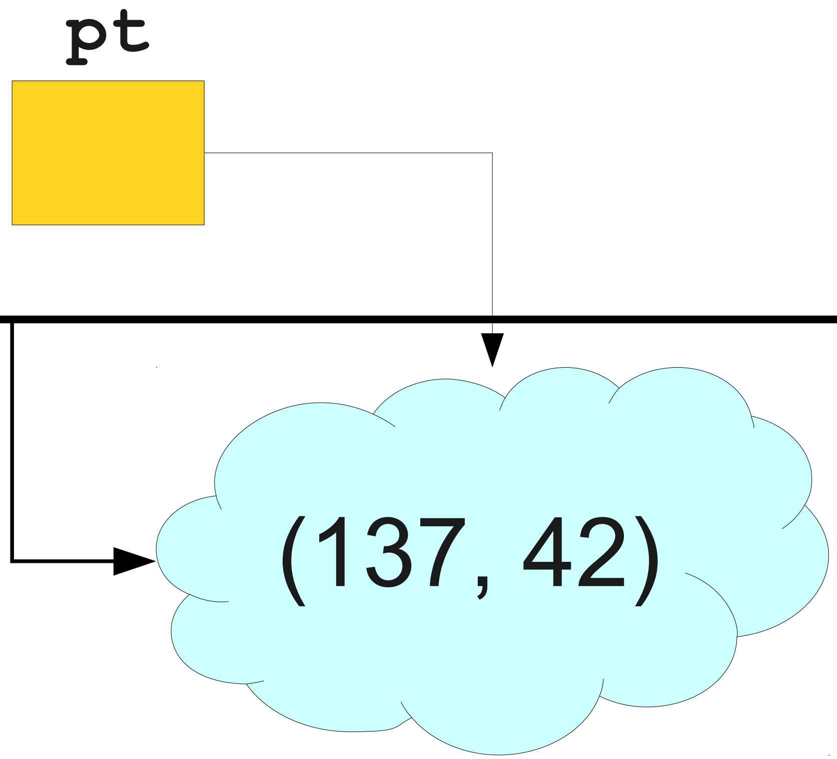
Pass by Reference



```
GPoint d = chooseRandomPoint();

pt.setLocation((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);
```

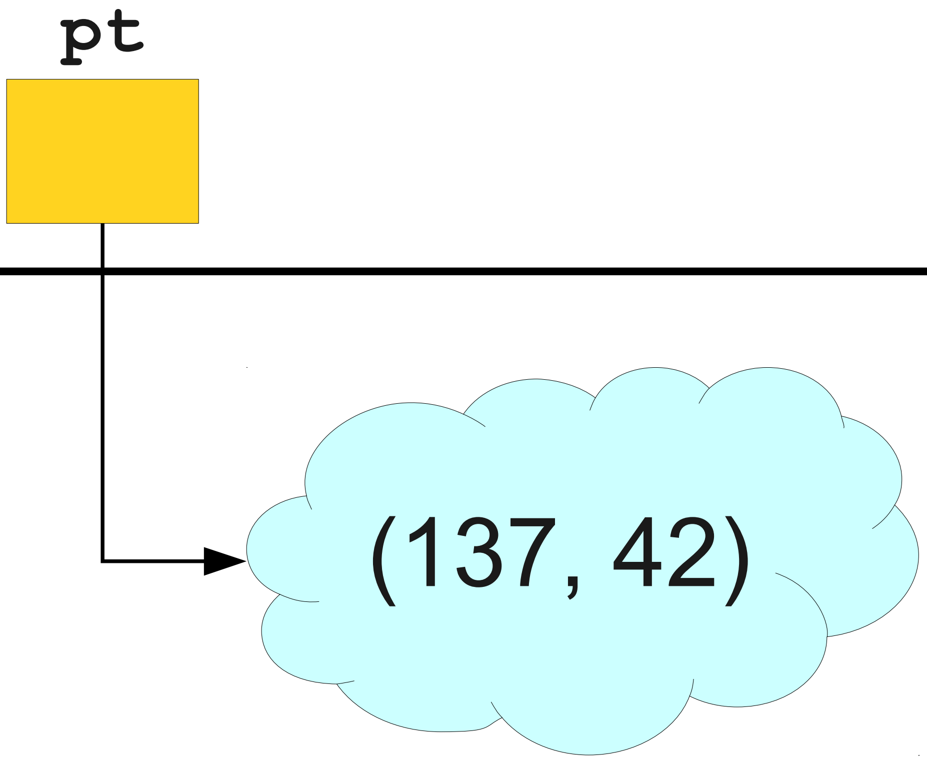
Pass by Reference

```
GPoint d = chooseRandomPoint();

pt.setLocation((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);
```



Pass by Reference

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

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



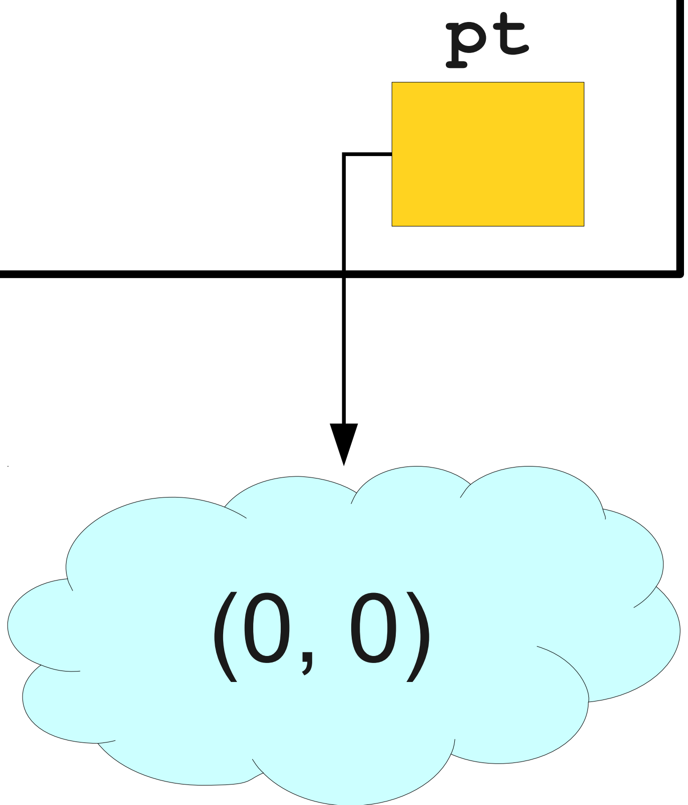
Parameter Passing

- Primitive types (**`int`**, **`double`**, **`boolean`**, etc.) are passed by **value**.
  - Changes made to them do not reflect in the caller.
- Objects (**`GRect`**, **`GOval`**, **`GPoint`**, etc.) are passed by **reference**.
  - Changes made to the referenced objects do reflect in the caller.

One More Change…

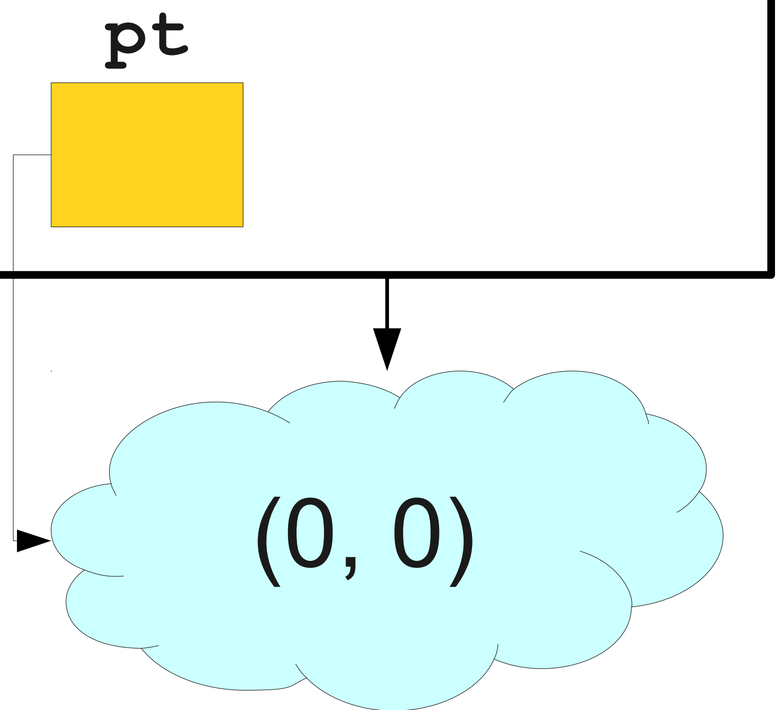
Pass by Reference, Take II

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

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



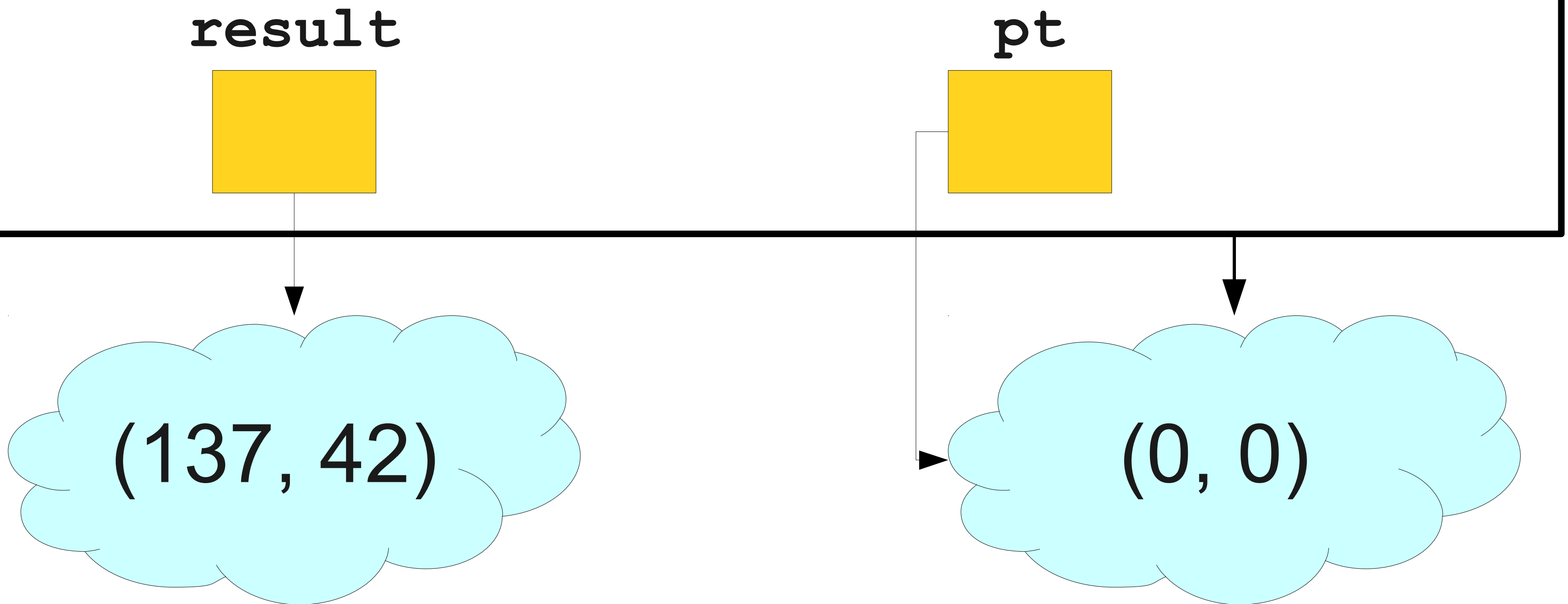
Pass by Reference, Take II

```
GPoint d = chooseRandomPoint();
GPoint result =
    new GPoint((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);

pt = result;
```



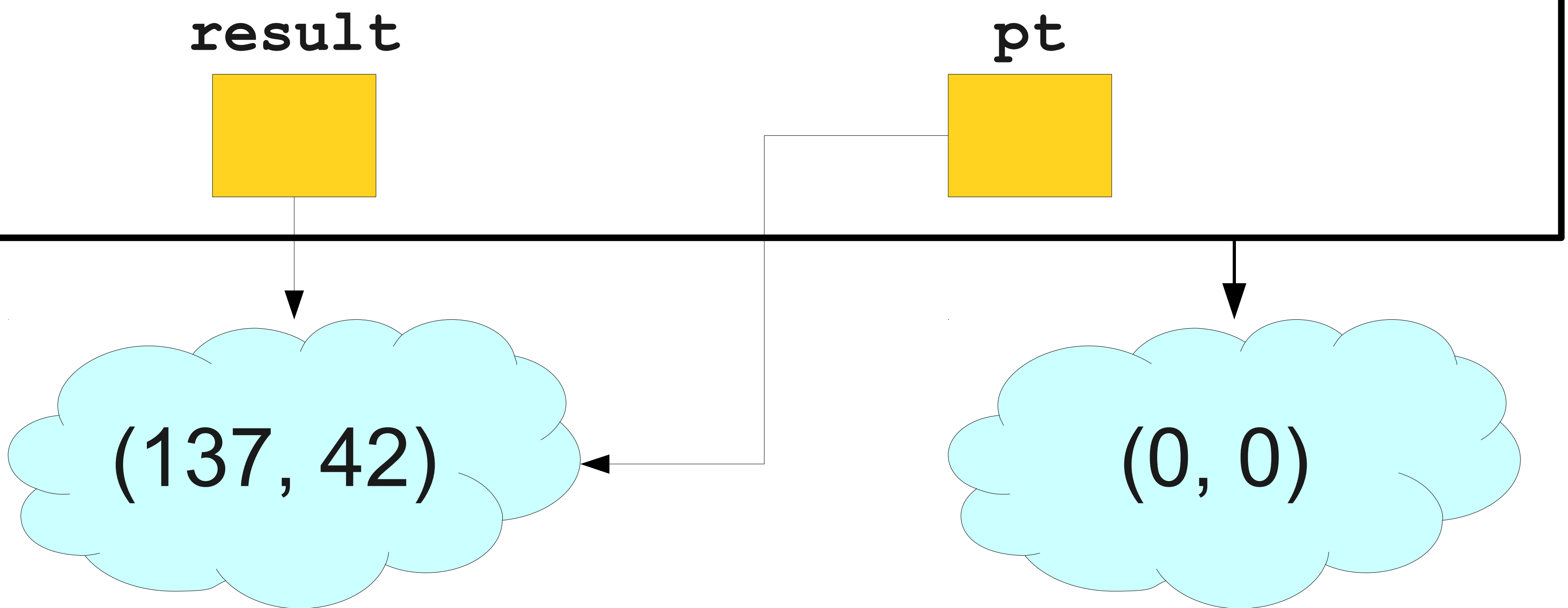
Pass by Reference, Take II

```
GPoint d = chooseRandomPoint();
GPoint result =
    new GPoint((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);

pt = result;
```



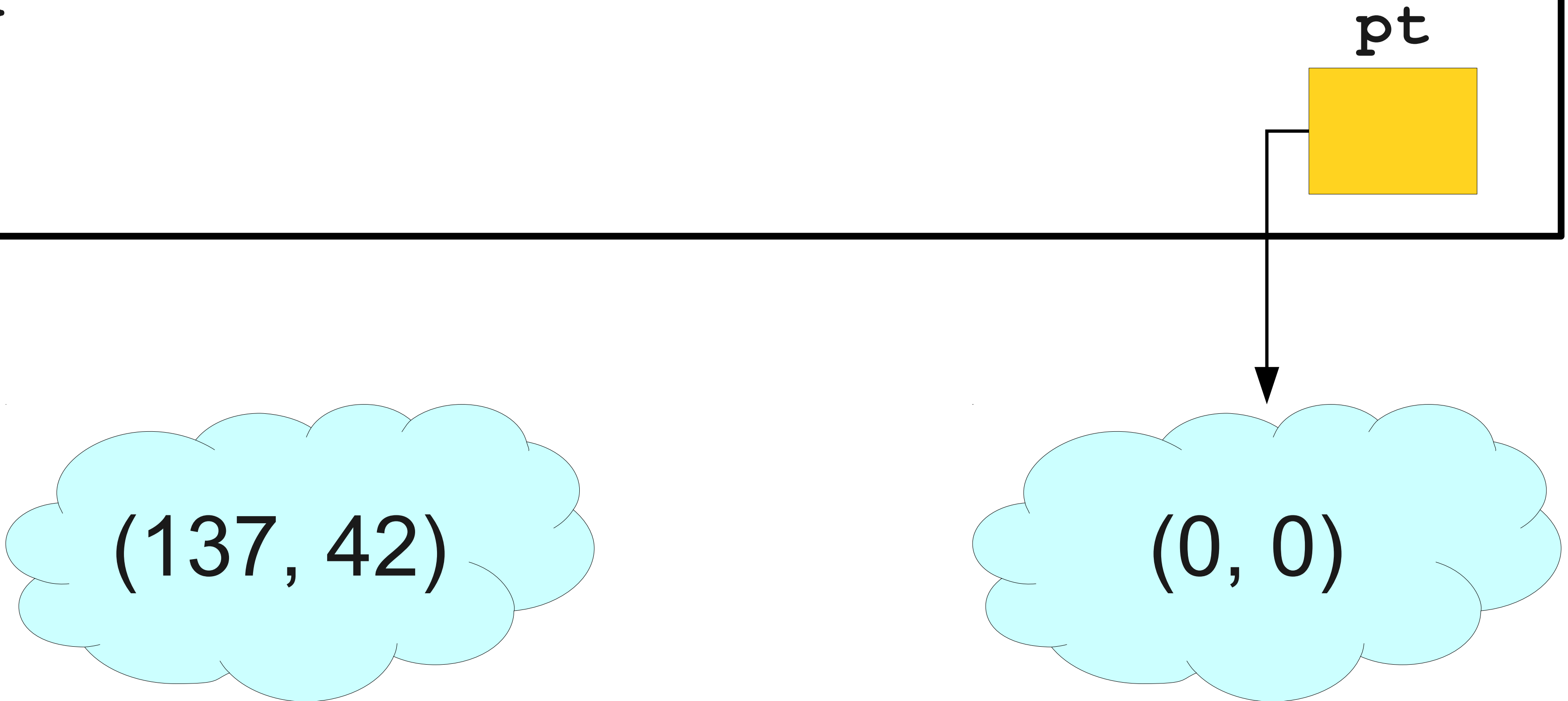
Pass by Reference, Take II

```
GPoint d = chooseRandomPoint();
GPoint result =
    new GPoint((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);

pt = result;
```



Pass by Reference, Take II

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

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



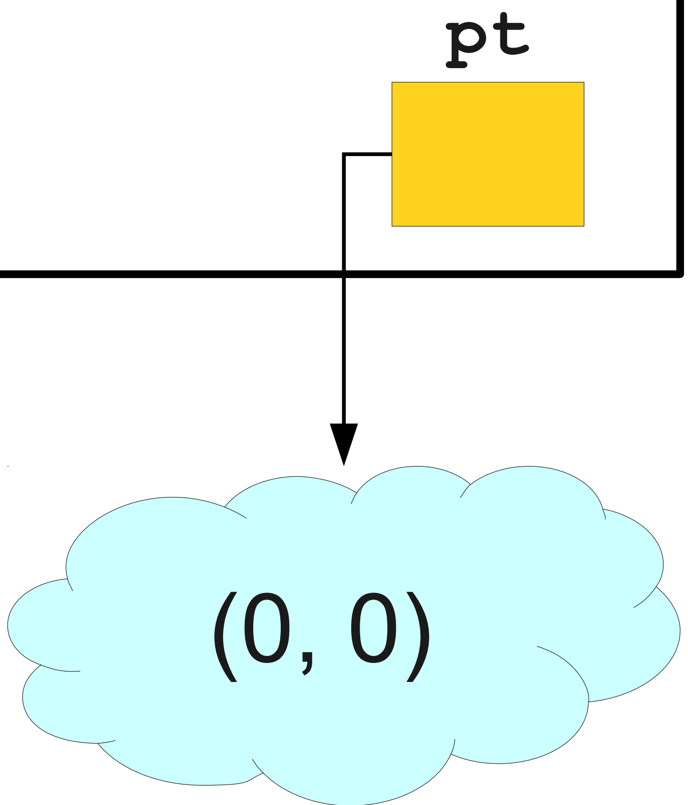
Parameter Passing

- Primitive types (**`int`**, **`double`**, **`boolean`**, etc.) are passed by **value**.
    - Changes made to them do not reflect in the caller.
- Objects (**`GRect`**, **`GOval`**, **`GPoint`**, etc.) are passed by **reference**.
    - Changes made to the referenced objects do reflect in the caller.
    - You cannot change **which object** is being referenced, though.

One Final Approach…

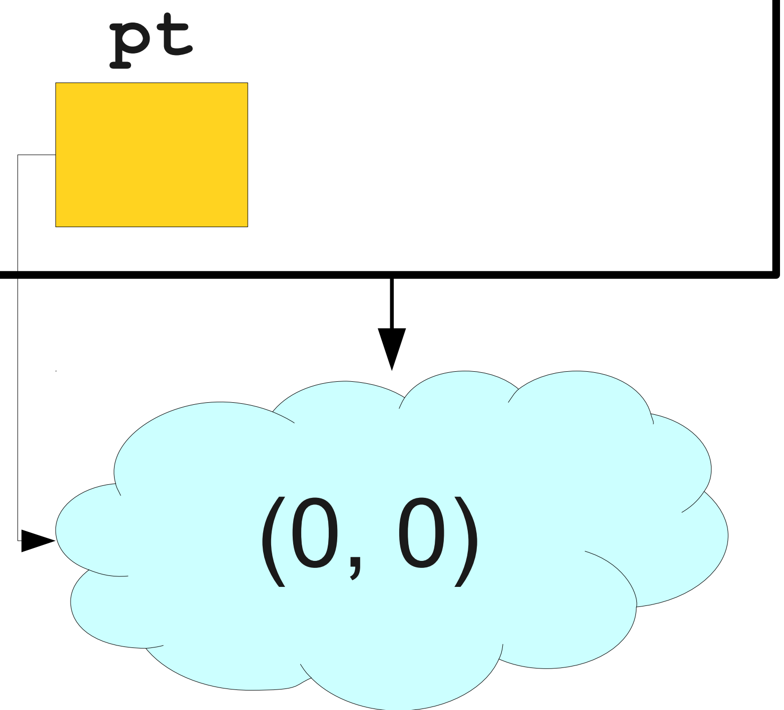
Returning Objects

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

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



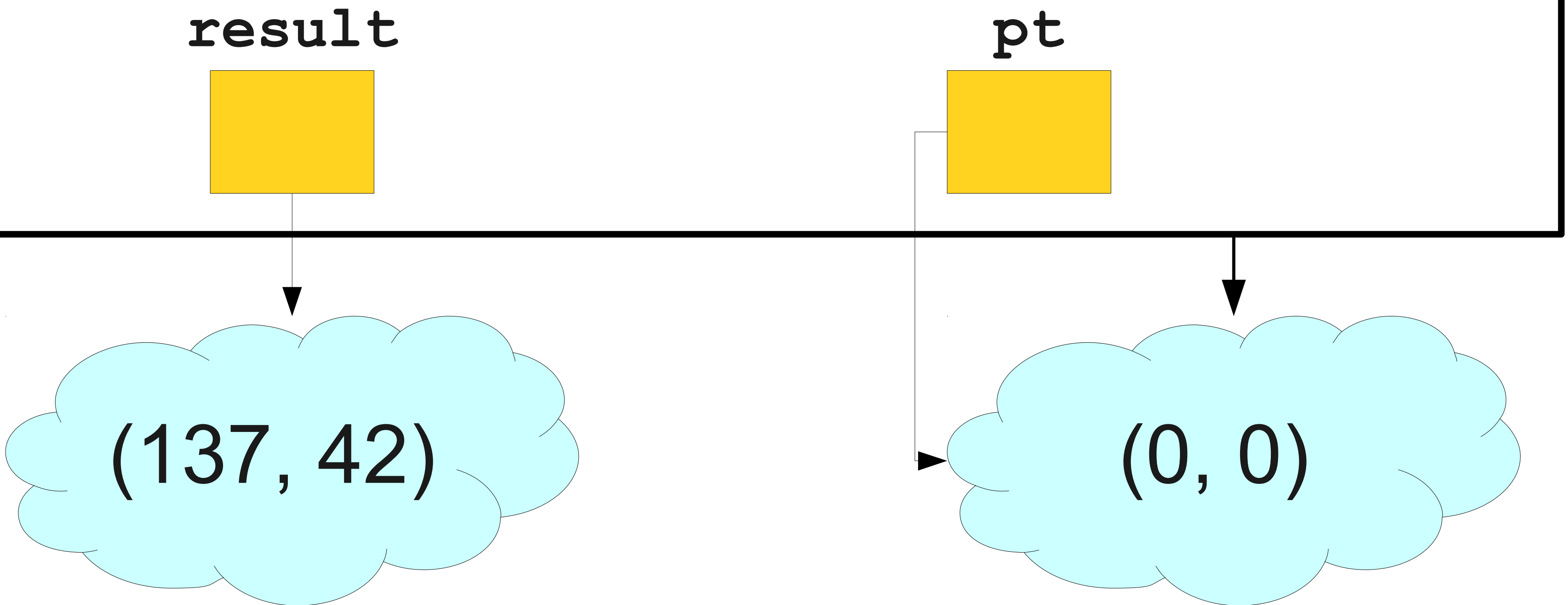
Returning Objects

```
GPoint d = chooseRandomPoint();
GPoint result =
    new GPoint((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);

return result;
```



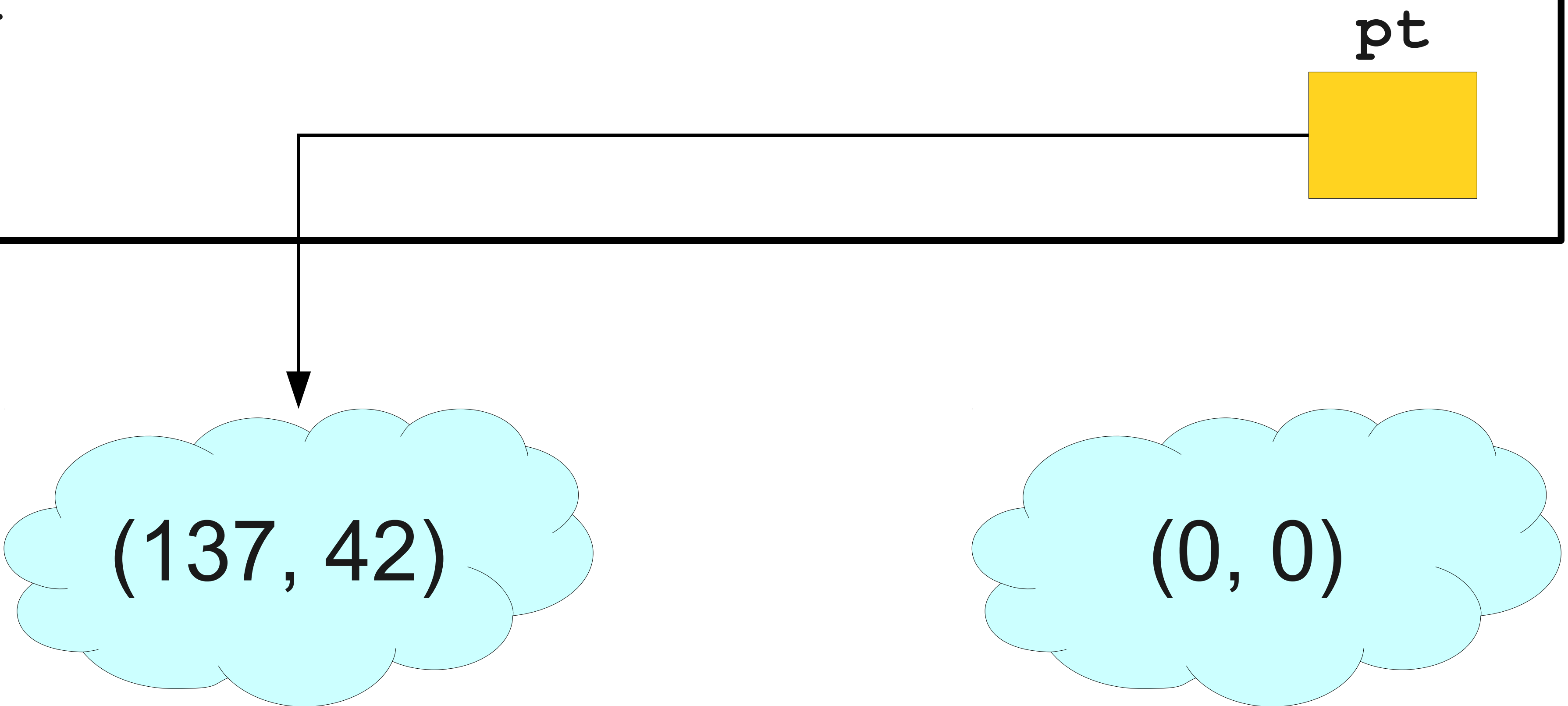
Returning Objects

```
GPoint d = chooseRandomPoint();
GPoint result =
    new GPoint((pt.getX() + d.getX()) / 2.0,
               (pt.getY() + d.getY()) / 2.0);

return result;
```



Returning Objects

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

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



Working with Text

The Data Type `char`

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

```
char letterA = 'A';
char plus = '+'
char zero = '0';
```

Escape Sequences

- An **escape sequence** is a sequence of characters in a program's source code that represents a single logical character.
- Examples:
  - **`\t`**: Horizontal tab
  - **`\n`**: Newline
  - **`\'`**: Single quote
  - **`\"`**: Double quote

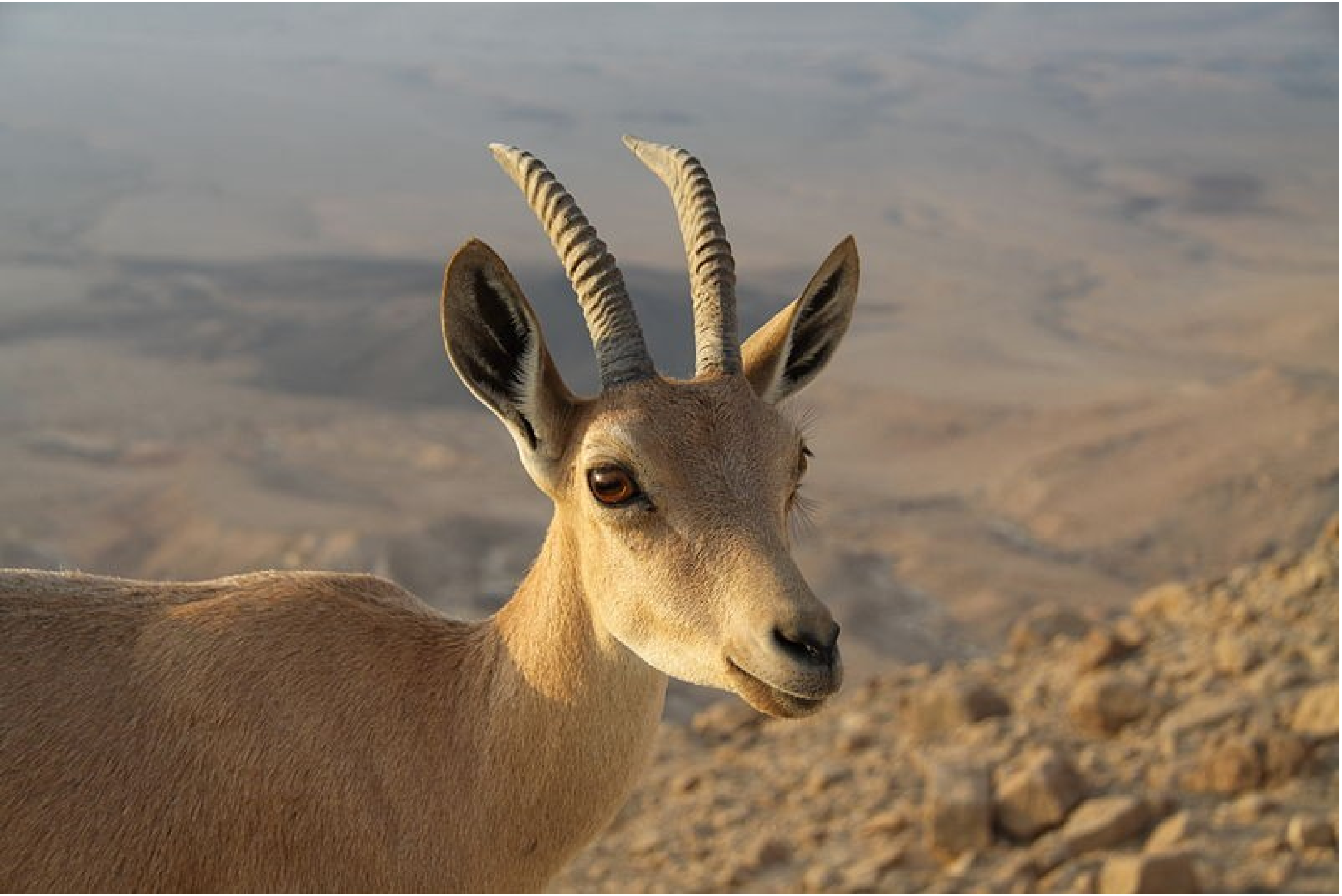
The ASCII Subset of Unicode

	0	1	2	3	4	5	6	7
00*x*	\000	\001	\002	\003	\004	\005	\006	\007
01*x*	\b	\t	\n	\011	\f	\r	\016	\017
02*x*	\020	\021	\022	\023	\024	\025	\026	\027
03*x*	\030	\031	\032	\033	\034	\035	\036	\037
04*x*	*space*	!	"	#	$	%	&	'
05*x*	(	)	*	+	,	-	.	/
06*x*	0	1	2	3	4	5	6	7
07*x*	8	9	:	;	<	=	>	?
10*x*	@	A	B	C	D	E	F	G
11*x*	H	I	J	K	L	M	N	O
12*x*	P	Q	R	S	T	U	V	W
13*x*	X	Y	Z	[	\	]	^	_
14*x*	`	a	b	c	d	e	f	g
15*x*	h	i	j	k	l	m	n	o
16*x*	p	q	r	s	t	u	v	w
17*x*	x	y	z	{	|	}	~	\177

The letter A, for example, has the Unicode value 101_8, which is the sum of the row and column labels.

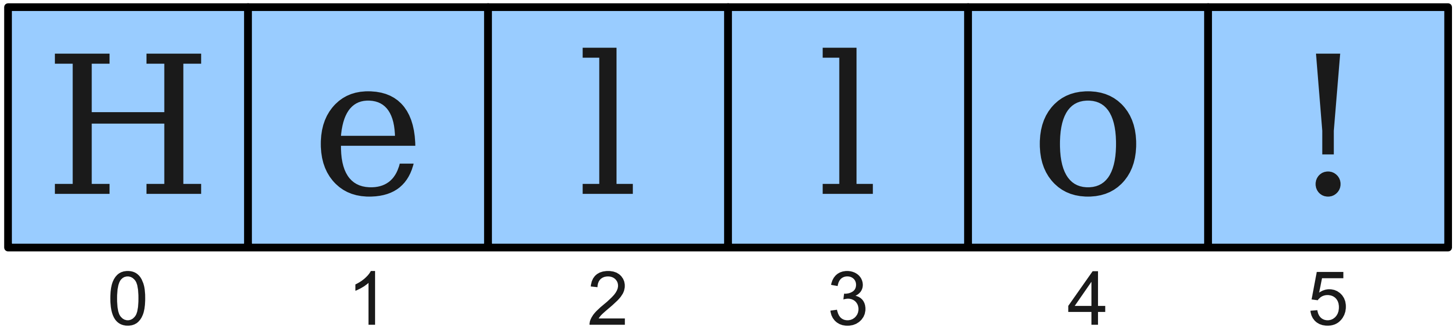
Based on slides by Eric Roberts and Mehran Sahami

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

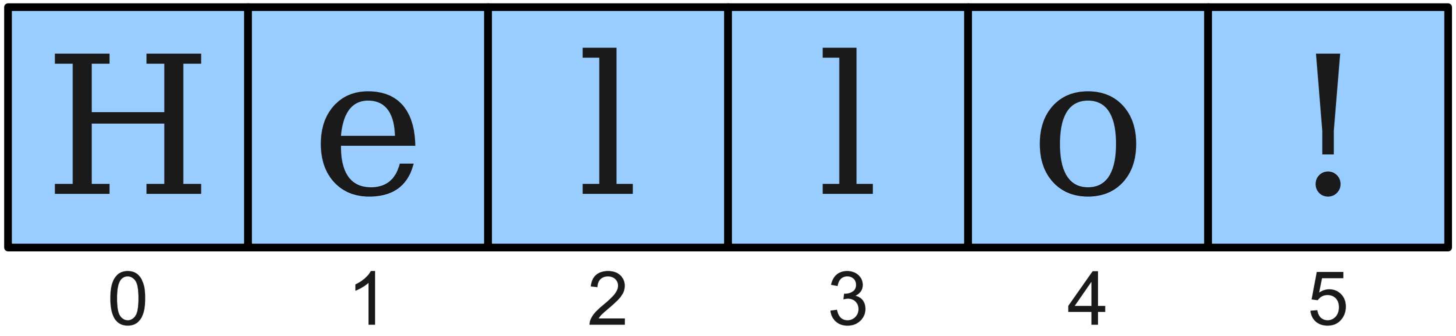


http://upload.wikimedia.org/wikipedia/commons/e/ea/Capra_ibex_nubiana_near_Mitzpe_Ramon_in_summer_2011_%284%29.JPG

Hello!



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



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

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