ITEC1620
Object-Based Programming

Lecture 14

References II

Using Classes

- What was necessary before using a class?
  - Construction

```
Random random = new Random();
Scanner scan = new Scanner(System.in);
```

Constructors

- Syntax has ()
  - () indicate a method
- Constructors are a special method that return a new instance of a class
  - May also pass in parameters

Constructors II

- Special method, special syntax

<access modifier> Classname
(<parameters>)

- No return type – must be instance of class
- Identifier must be same as class

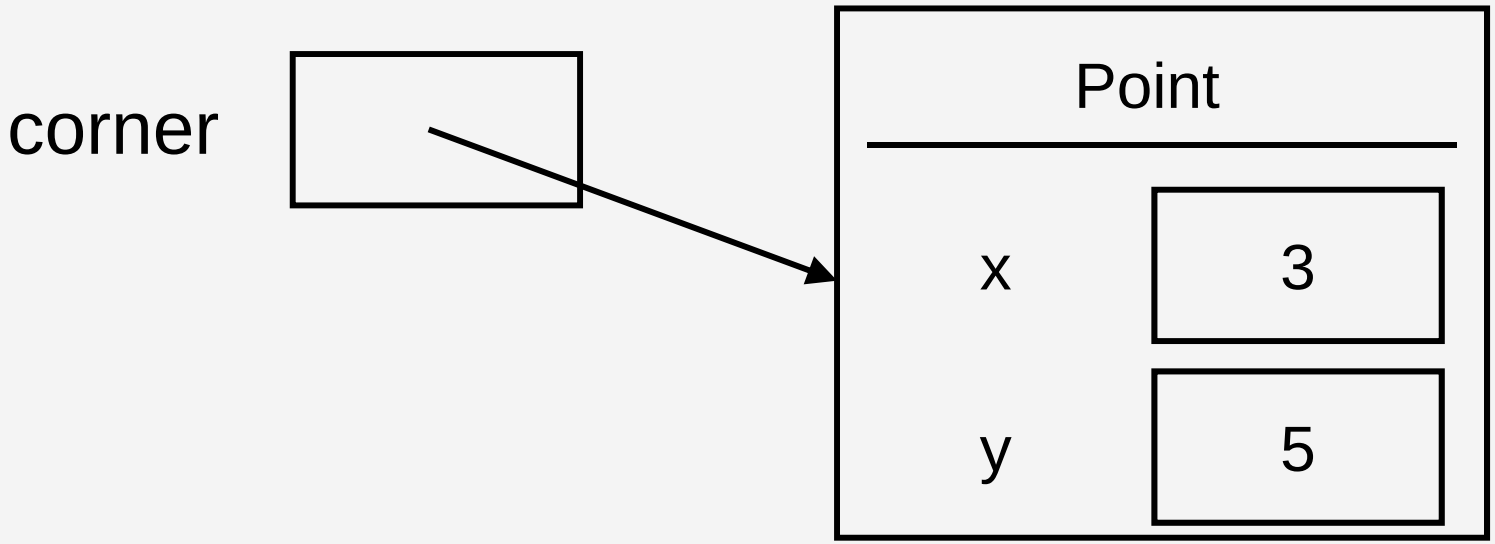
Example

```
public class Point
{
   public int x;
   public int y;

   public Point (int x, int y)
   // parameters get assigned to instance
   variables
}
```

Example II

Point corner = new Point (3, 5);



Review

- Java does not know how to initialize your datatype
- You must use a constructor
  - A constructor is a special method that has the same name as the class
- The constructor allocates space in memory for all the fields of the class
  - Space allocation requires “new”

Review II

- You must use the constructor before using your objects

```
Point pointA;
pointA.x = 0;            // error

Point pointB = new Point();
pointB.x = 0;            // OK
```

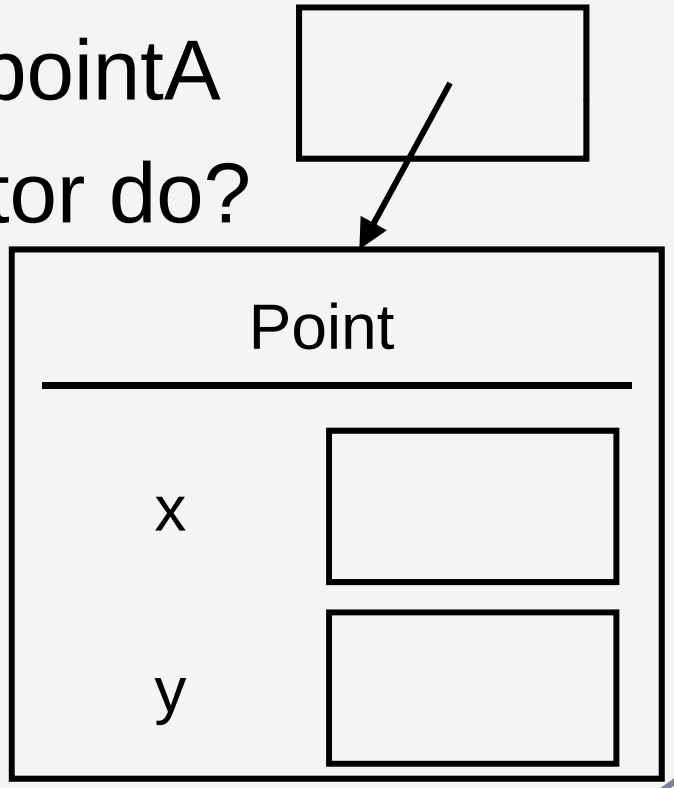
Review III

- What does the declaration do?

  Point pointA;

- What does the constructor do?

  pointA = new Point();



Constructor Actions

- Allocates memory space for all fields
  - Numerical fields initialized to 0
  - Boolean fields initialized to false
  - Non-primitive (object) fields initialized to null
- Additional initializations from parameters
- Returns a reference to the new object

Actions in Detail

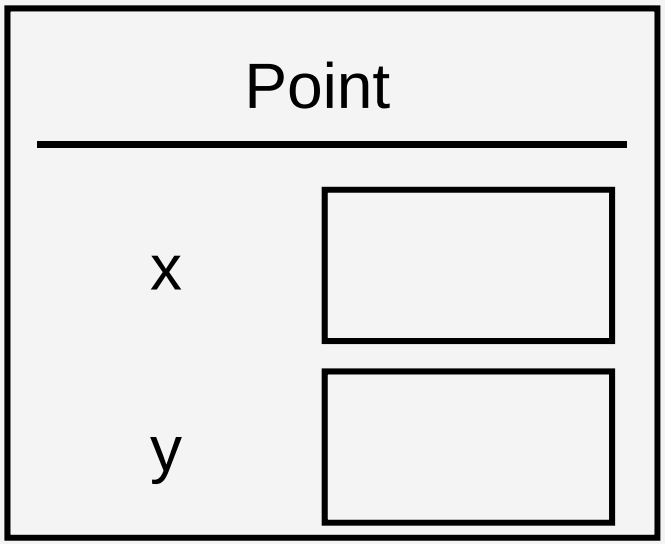
Point corner = new Point (3, 5);

- Declaration

corner 

Actions in Detail II

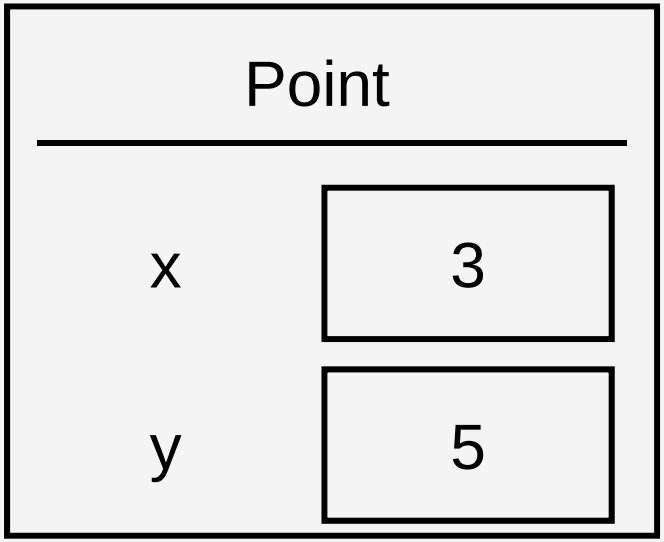
- Allocates memory space for all fields
  - Occurs before any additional initializations



Actions in Detail III

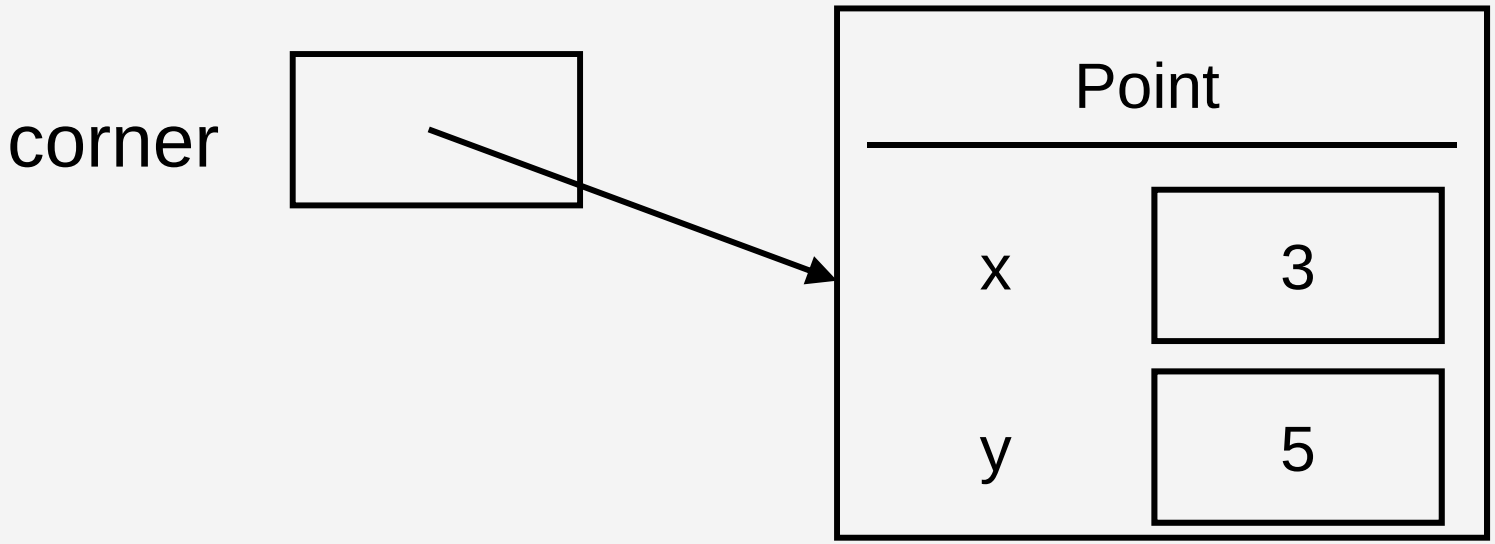
- Initializations from parameters

(3, 5)



Actions in Detail IV

- Returns a reference to the new object



Questions?

Constructors and Static Methods

- A constructor is a (special) method in a class that can be called without an instance
  - It is called to create the instance
- A method that can be called without the instance is static (i.e. stateless)
  - All constructors are static by default

Null Reference

- Null is a reference to no object

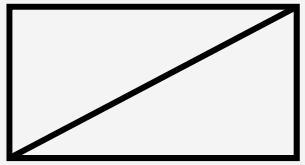
  Point corner;

  corner

  - Uninitialized

  Point corner = null;

  corner



  - Initialized to null

Static Methods

- Normal methods can access the fields of an object
  - void setX (int newX);
- Static methods can be called without a constructor
  - No constructor to create fields, no access to fields

Static Methods II

- Static methods already seen

System.out.println();

Math.sqrt();

Overloading

- An overloaded method/constructor has the same identifier with different parameters
- Constructors must be overloaded since the identifier is fixed to the class name
- Methods can be overloaded to provide the “same” function under different circumstances

Method Overloading

- System.out.println()
  - Can handle nothing, Strings, numbers…
- System.out.println()
- System.out.printStringln(“hello”)
- System.out.printIntln(5)
- System.out.printFloatln(5.5)

Method Overloading II

- Division

public int divide (int dividend, int divisor)

public double divide (double dividend,
double divisor)

Parameter Matching and Promotion

- 5/2
  - 5 is int, 2 is int
  - Call divide with int parameters and return an int
- 5/2.0
  - 5 is int, 2.0 is double
  - double does not match int
  - int can match double with promotion
  - Call divide with double parameters and return a double

Pass-by-copy

- All parameters in JAVA are passed by copy
- Since methods only have a copy of the original values, methods cannot cause side-effects

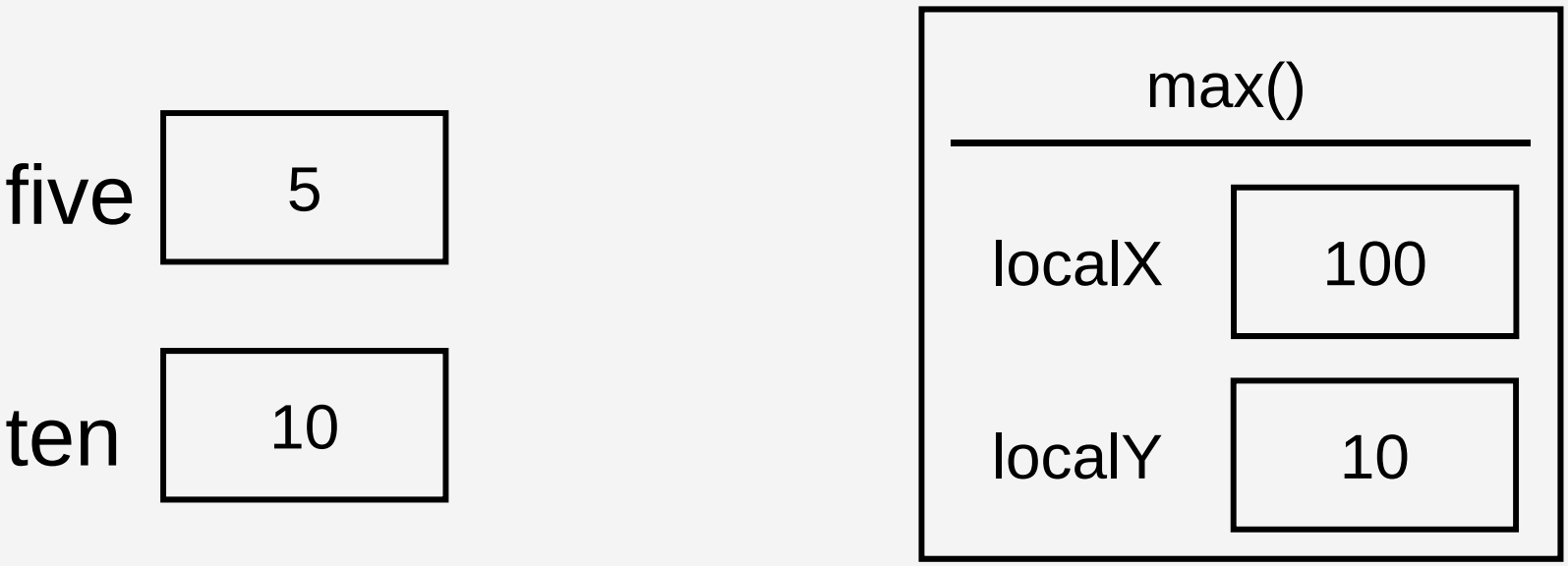
Pass-by-copy II

```
int five = 5;
int ten = 10;
int largest = max (five, ten);
```

- Expect five to be 5 and ten to be 10 after method call

Pass-by-copy III

- max() only works with local variables
  - Not five, ten
- Local variables “die” after method ends



Pass-by-copy IV

- Pass-by-copy is safe
  - Good encapsulation
  - No side-effects

Pass-by-reference

- All parameters in JAVA are passed by copy
- What happens if we pass an object?
  - Object identifiers contain references
  - Passing a copy of an object passes a copy of the object reference
  - This is pass-by-reference!

Pass-by-reference II

```
AnInt intA = new AnInt();
AnInt intB = new AnInt();
intA.setValue(5);
intB.setValue(intA.value);
   // guaranteed safe – intA value pass-by-copy
intB.setValue(intA);
   // maybe unsafe – intA value pass-by-
   reference
```

Pass-by-reference III

public void setValue (int newValue)

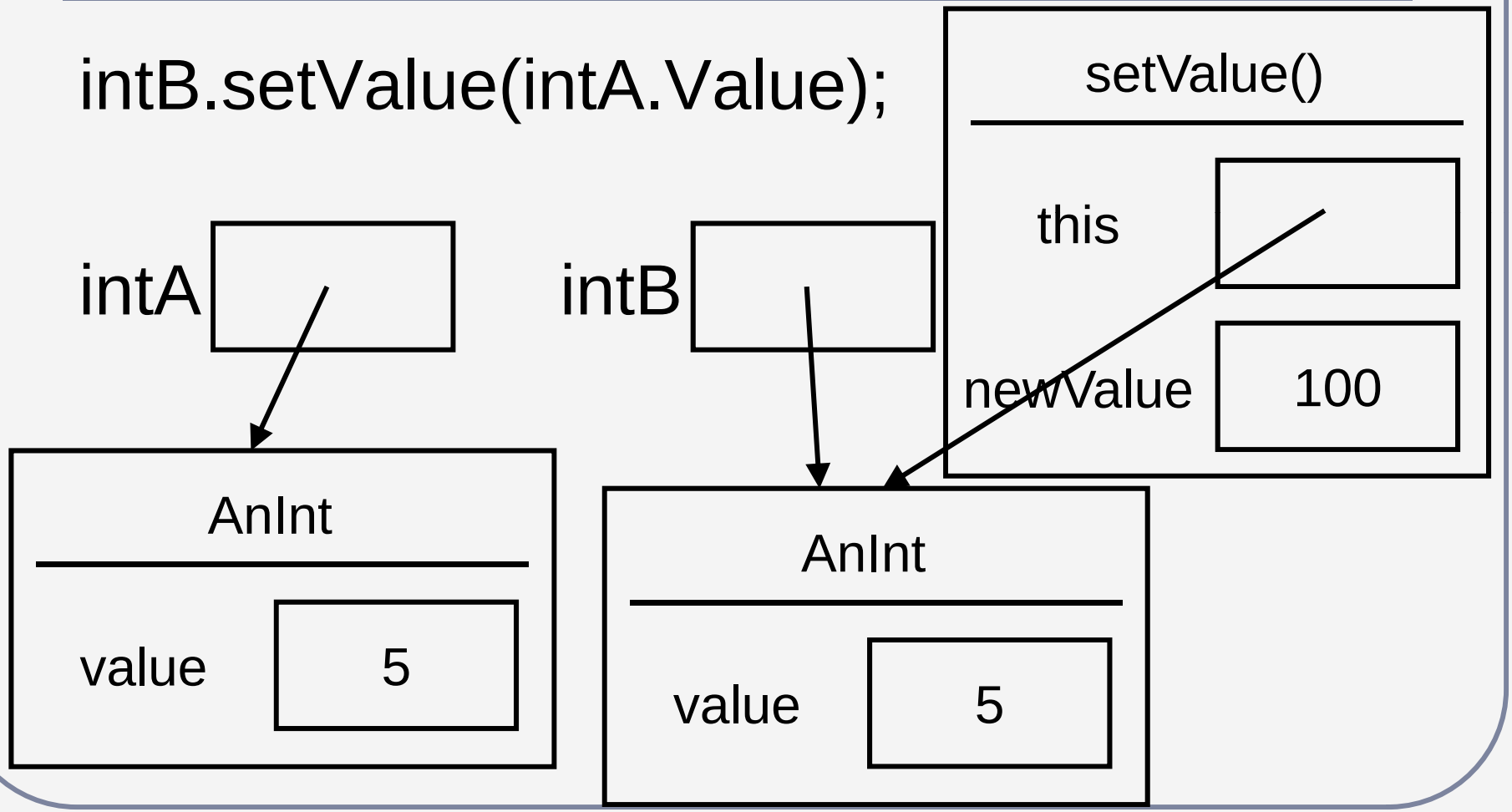
- newValue is only local copy
  - No outside effects

public void setValue (AnInt newValue)

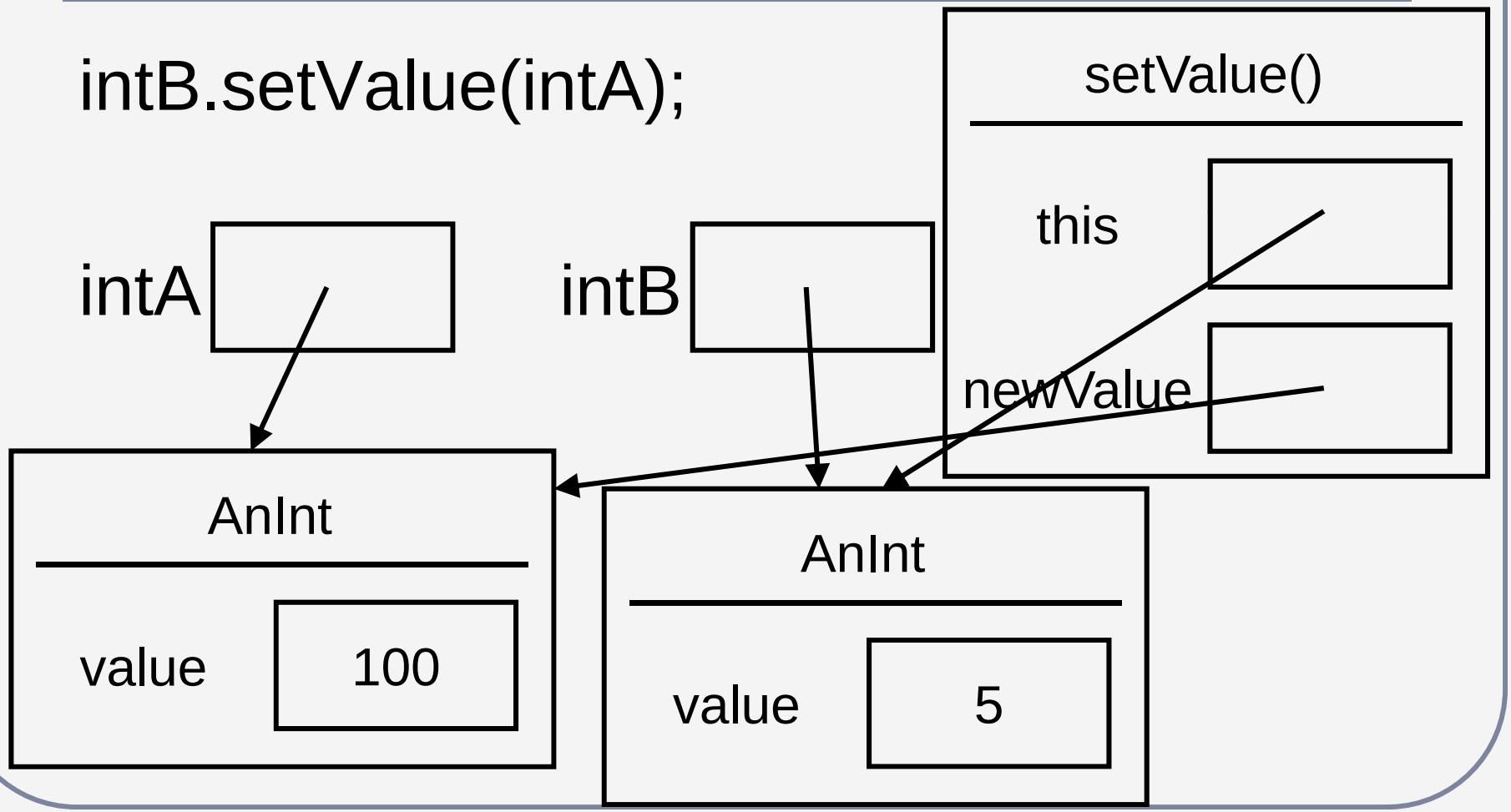
- newValue is a local copy
  - Copy can still reference outside data!

Pass-by-reference IV

intB.setValue(intA.Value);



Pass-by-reference V



Questions?

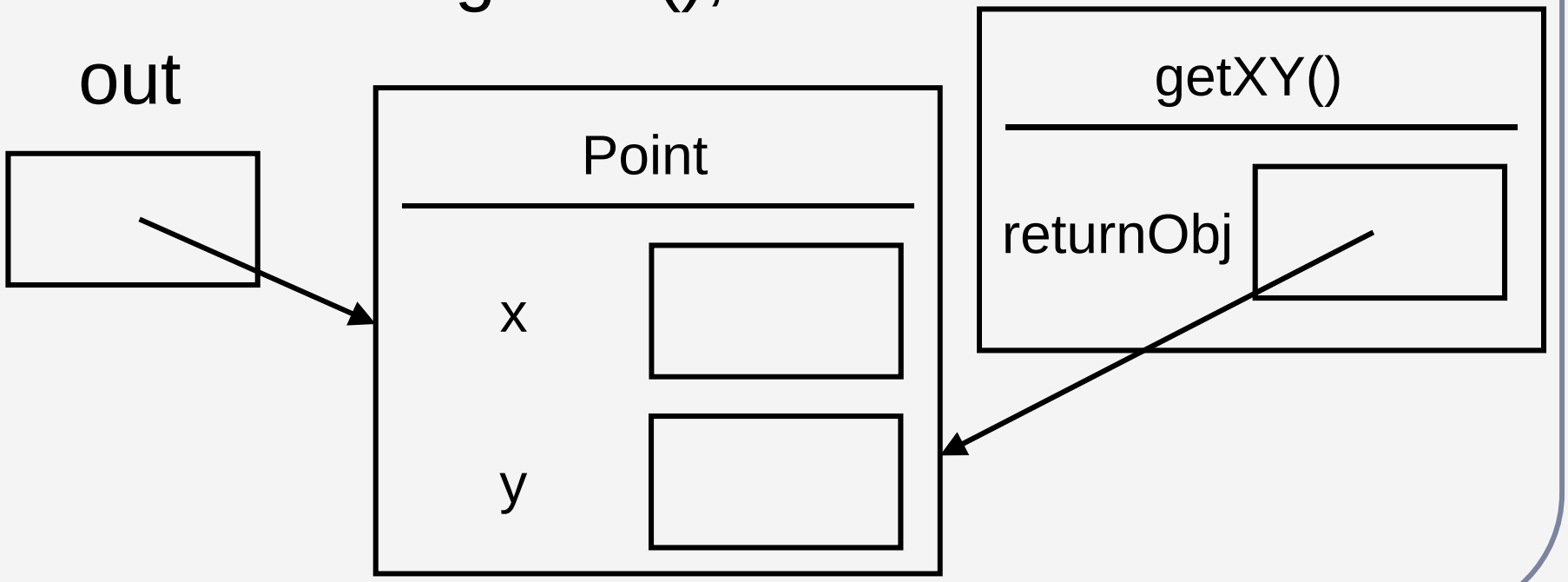
Exploiting Pass-by-Reference

- Methods have one return value
- Objects can contain multiple values
- Make return type an object

Exploiting Pass-by-Reference II

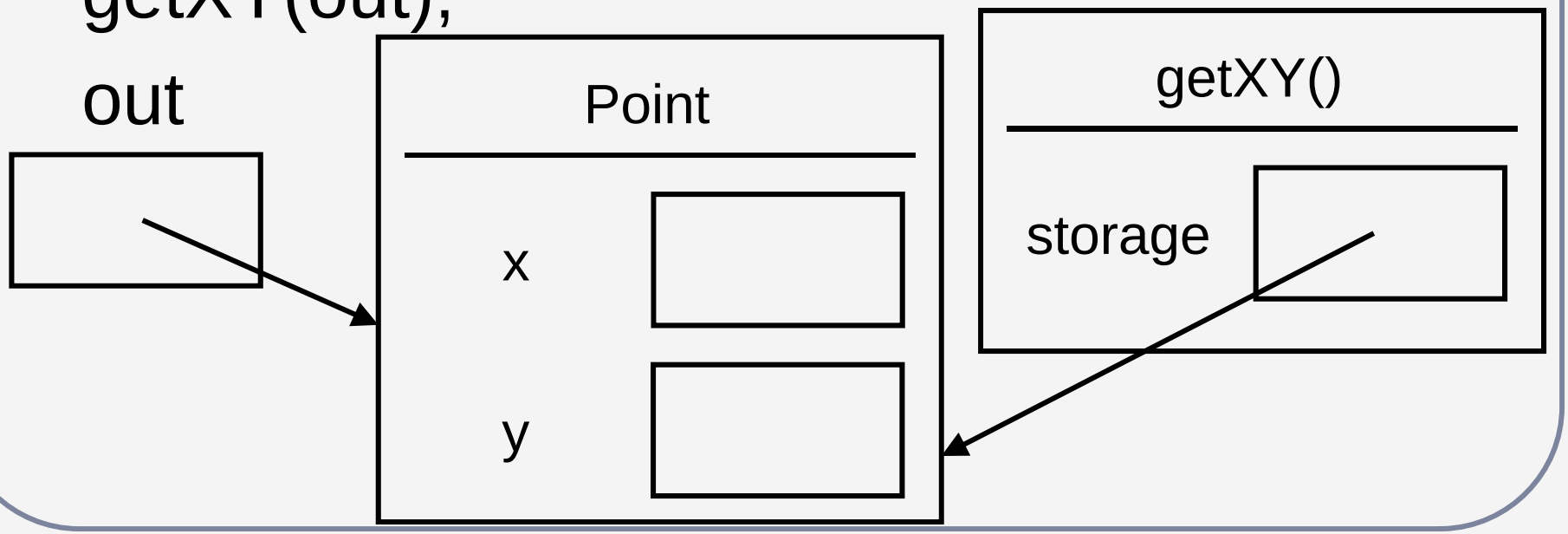
public static Point getXY();

Point out = getXY();



Exploiting Pass-by-Reference III

```
public static void getXY (Point storage)
Point out = new Point();
getXY(out);
```



Readings and Assignments

- Text sections (5th and 6th edition)
  - 6.8
- Text sections (7th edition)
  - 7.8