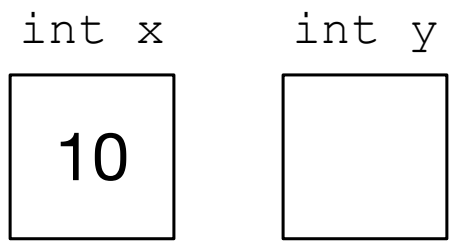
CIS 1068
Methods, References Again

primitives

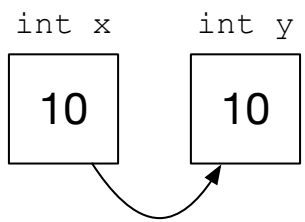
Simple Primitives

```
1  public class Funcs00 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          System.out.println("x=" + x + "y=" + y);
7      }
8  }
```



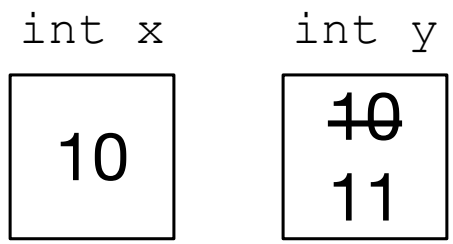
Copy What's in y into x

```
1  public class Funcs00 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          System.out.println("x=" + x + "y=" + y);
7      }
8  }
```



y is changed. x is not

```
1  public class Funcs00 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          System.out.println("x=" + x + "y=" + y);
7      }
8  }
```



Exact Same Thing, but with a Method

```
1   public class Funcs01 {
2       public static void func(int y) {
3           y++;
4       }
5
6       public static void main(String args[]) {
7           int x=10;
8           func(x);
9           System.out.println(x);
10          // System.out.println(y); scope error
11      }
12  }
```

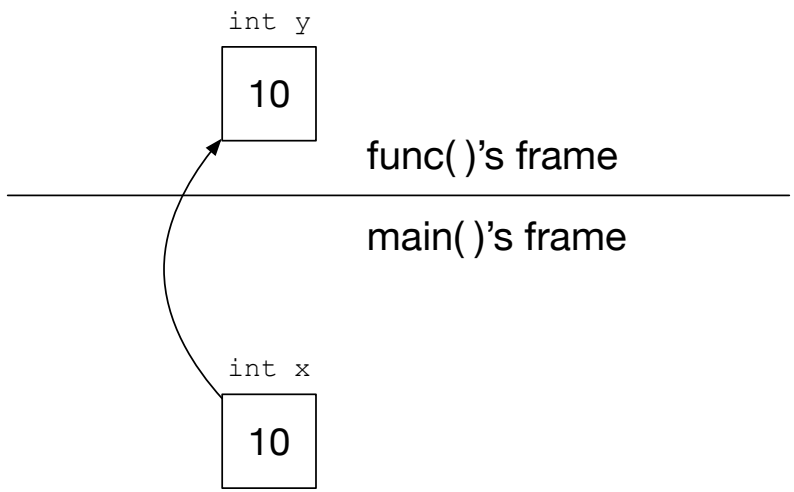
main( )'s frame

int x

10

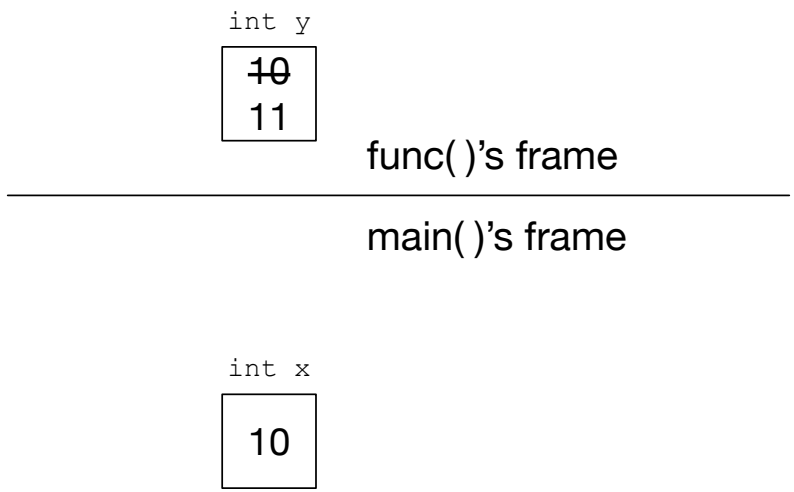
Copy What's in x into y

```
1  public class Funcs01 {
2      public static void func(int y) {
3          y++;
4      }
5
6      public static void main(String args[]) {
7          int x=10;
8          func(x);
9          System.out.println(x);
10         // System.out.println(y); scope error
11     }
12 }
```



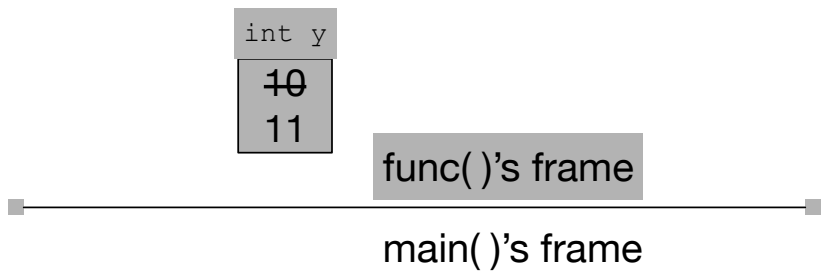
Modify y

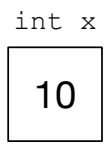
```
1  public class Funcs01 {
2      public static void func(int y) {
3          y++;
4      }
5
6      public static void main(String args[]) {
7          int x=10;
8          func(x);
9          System.out.println(x);
10         // System.out.println(y); scope error
11     }
12 }
```



Method finishes. Memory used by method is freed

```
1  public class Funcs01 {
2      public static void func(int y) {
3          y++;
4      }
5
6      public static void main(String args[]) {
7          int x=10;
8          func(x);
9          System.out.println(x);
10         // System.out.println(y); scope error
11     }
12 }
```

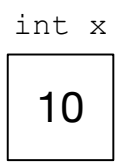




x Remains Unchanged

```
1   public class Funcs01 {
2       public static void func(int y) {
3           y++;
4       }
5
6       public static void main(String args[]) {
7           int x=10;
8           func(x);
9           System.out.println(x);
10          // System.out.println(y); scope error
11      }
12  }
```

main( )'s frame



What do we get when we create a method?

What do we get when we create a method?

- named hunk of code

What do we get when we create a method?

- named hunk of code
- a scope

What do we get when we create a method?

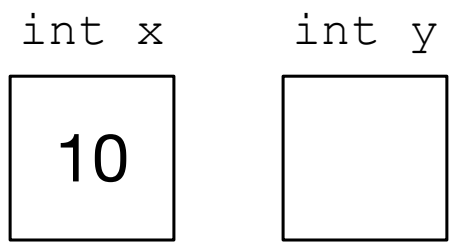
- named hunk of code
- a scope anything declared within the method is:
  - *local* to the method
  - meaning that
    - it's only visible within the method
    - can't access it outside the method
    - this includes parameters

Why create methods?

- small simple methods easier to test
- easier to understand
- when something is hard to read, it's easy to make a mistake
- manage complexity
- ease of code re-use

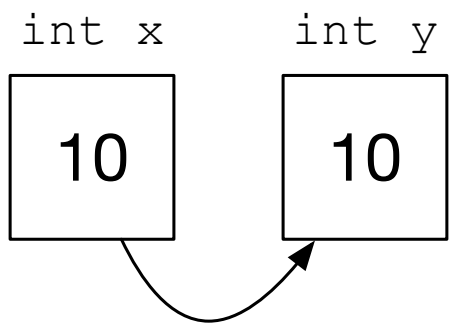
Same idea, but we copy something back

```
1  public class Funcs02 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          x=y;
7          System.out.println("x=" + x + "y=" + y);
8      }
9  }
```



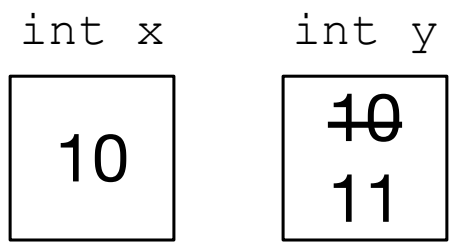
What's in x is copied into y

```
1  public class Funcs02 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          x=y;
7          System.out.println("x=" + x + "y=" + y);
8      }
9  }
```



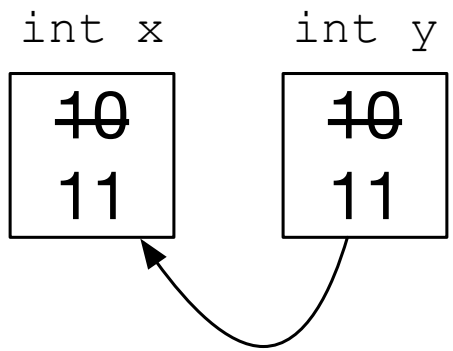
Change y

```
1  public class Funcs02 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          x=y;
7          System.out.println("x=" + x + "y=" + y);
8      }
9  }
```



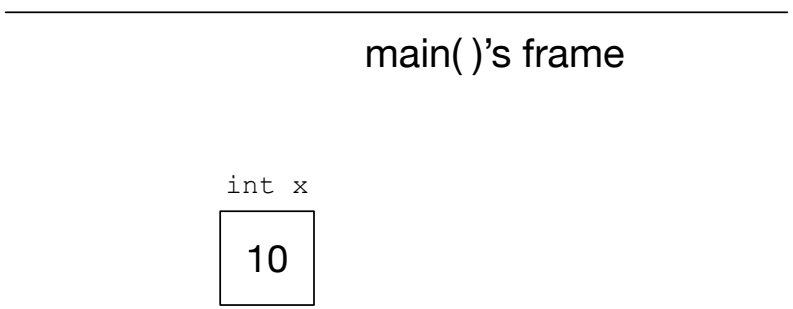
Copy what's in y back into x

```
1  public class Funcs02 {
2      public static void main(String args[]) {
3          int x=10;
4          int y=x;
5          y++;
6          x=y;
7          System.out.println("x=" + x + "y=" + y);
8      }
9  }
```



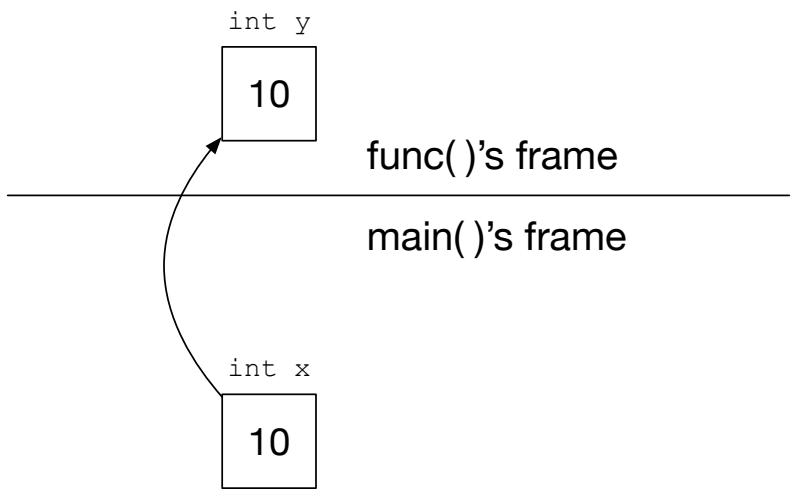
Exact same thing but with a method

```
1   public class Funcs03 {
2       public static int func(int y) {
3           y++;
4           return y;
5       }
6
7       public static void main(String args[]) {
8           int x=10;
9           x=func(x);
10
11          // System.out.println("x=" + x + ", y=" + y);
12      }
13  }
```



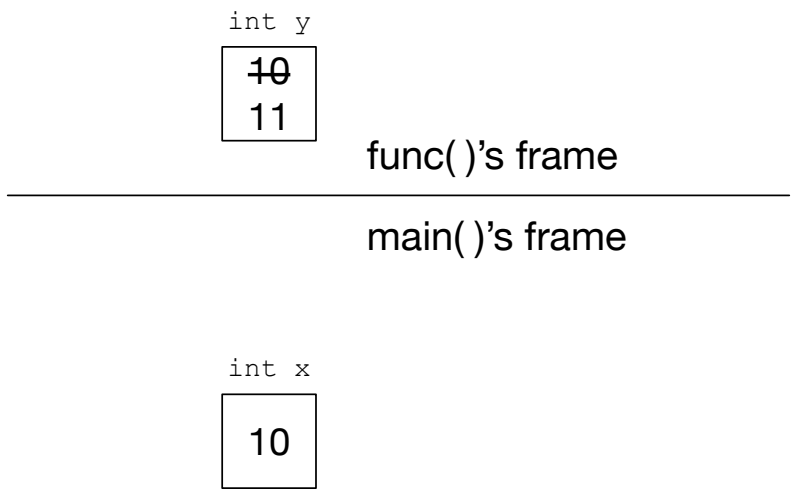
Copy what's in x into y

```
1  public class Funcs03 {
2      public static int func(int y) {
3          y++;
4          return y;
5      }
6
7      public static void main(String args[]) {
8          int x=10;
9          x=func(x);
10         // System.out.println("x=" + x + ", y=" + y);
11     }
12 }
```



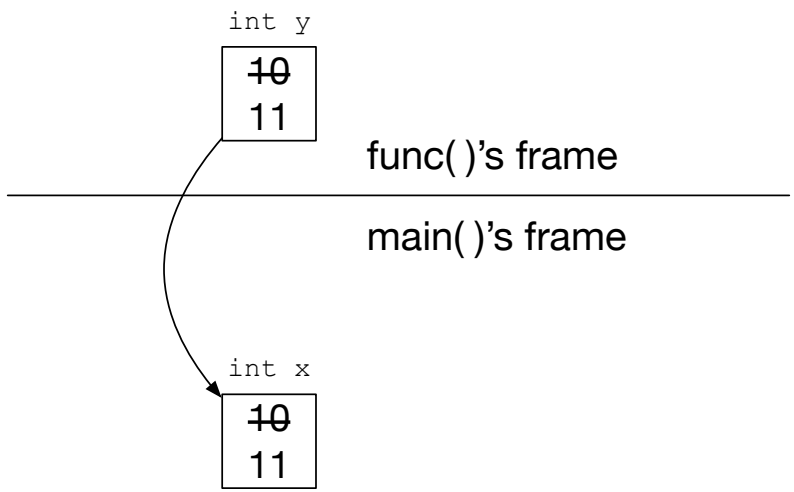
Modify y

```
1  public class Funcs03 {
2      public static int func(int y) {
3          y++;
4          return y;
5      }
6
7      public static void main(String args[]) {
8          int x=10;
9          x=func(x);
10         // System.out.println("x=" + x + ", y=" + y);
11     }
12 }
```



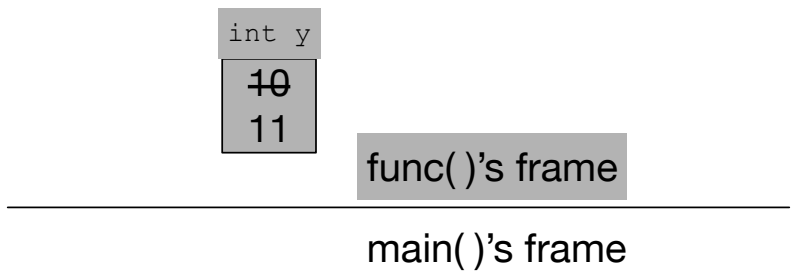
Return. Copy returned value to what's to the left of '='

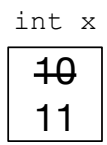
```
1   public class Funcs03 {
2       public static int func(int y) {
3           y++;
4           return y;
5       }
6
7       public static void main(String args[]) {
8           int x=10;
9           x=func(x);
10          // System.out.println("x=" + x + ", y=" + y);
11      }
12  }
```



Method is finished. Memory it occupied is freed

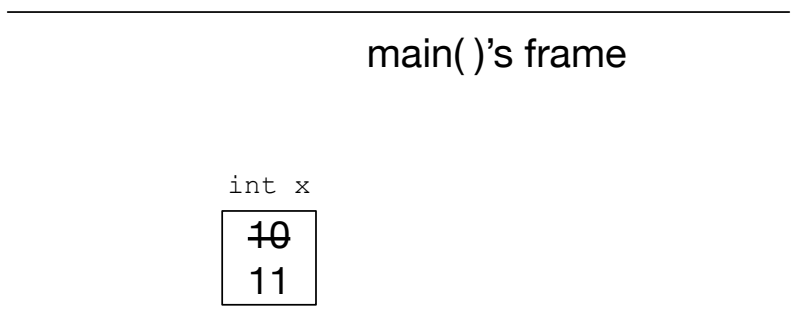
```
1  public class Funcs03 {
2      public static int func(int y) {
3          y++;
4          return y;
5      }
6
7      public static void main(String args[]) {
8          int x=10;
9          x=func(x);
10         // System.out.println("x=" + x + ", y=" + y);
11     }
12 }
```





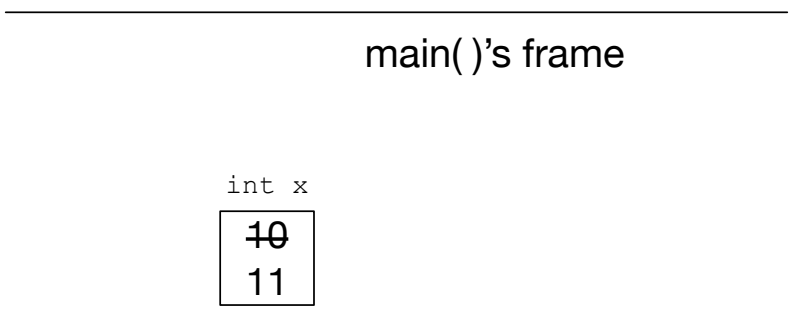
Method is finished. Memory it occupied is freed

```
1   public class Funcs03 {
2       public static int func(int y) {
3           y++;
4           return y;
5       }
6
7       public static void main(String args[]) {
8           int x=10;
9           x=func(x);
10          // System.out.println("x=" + x + ", y=" + y);
11      }
12  }
```



y was local to `func`. It's inaccessible here

```
1   public class Funcs03 {
2       public static int func(int y) {
3           y++;
4           return y;
5       }
6
7       public static void main(String args[]) {
8           int x=10;
9           x=func(x);
10          /* This would be a compiler error */
11          // System.out.println("x=" + x + ", y=" + y);
12      }
13  }
```



references

References

```
1  import java.awt.*;
2
3  public class Funcs04 {
4      public static void main(String args[]) {
5          Point p1 = new Point(10,20);
6          Point p2 = p1;
7
8          p2.x++;
9
10         System.out.println("p1 = " + p1 + ", p2=" + p2);
11     }
12 }
```

References

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```

p1 and p2 are *not* Points

References

```
1  import java.awt.*;
2
3  public class Funcs04 {
4      public static void main(String args[]) {
5          Point p1 = new Point(10,20);
6          Point p2 = p1;
7
8          p2.x++;
9
10         System.out.println("p1 = " + p1 + ", p2=" + p2);
11     }
12 }
```

p1 and p2 are *not* Points

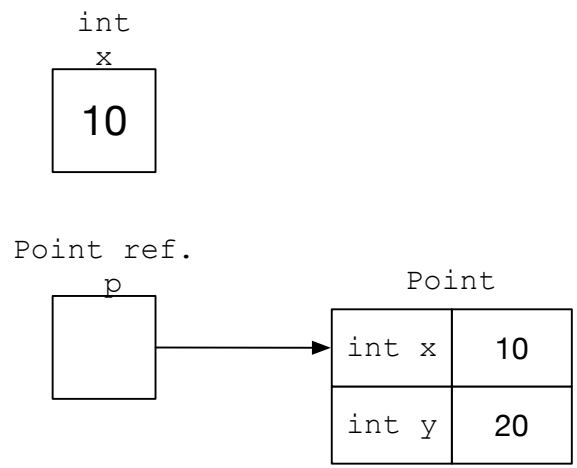
- They're *references* to Points
- Each contains the *location* of a Point

References

Recall all data types in Java one of two categories:

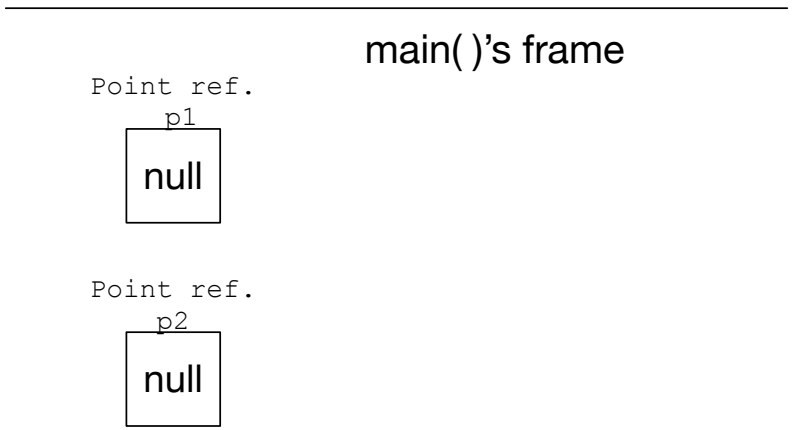
1. primitives
2. objects

- **primitive-types**
  - `int x;`
  - `x` contains an integer
- **object-types**
  - `Point p = new Point();`
  - `p` is *not* a `Point`
  - holds *location* of `Point`
  - `new Point()` creates `Point`



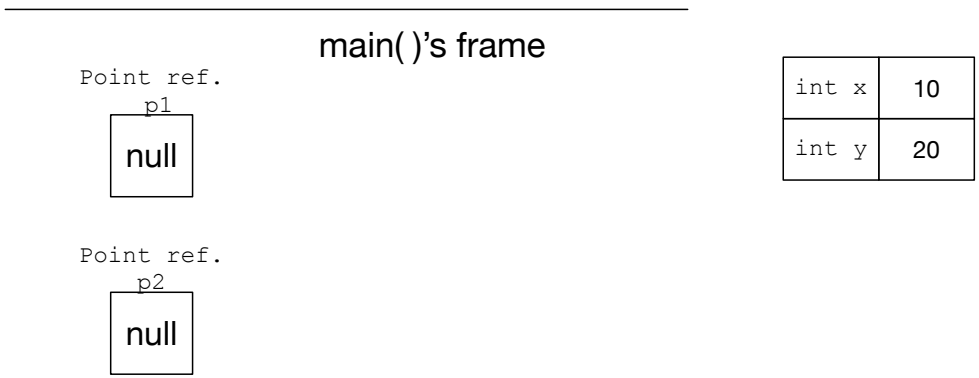
Before the new operator, we have no Points

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



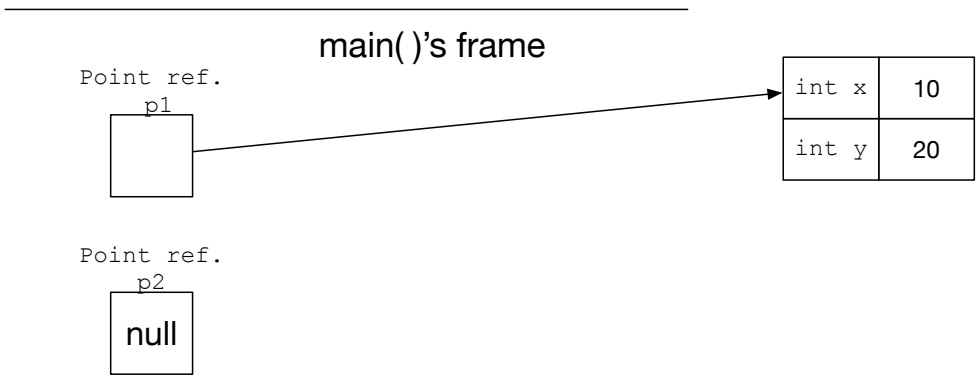
new Point creates a point

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



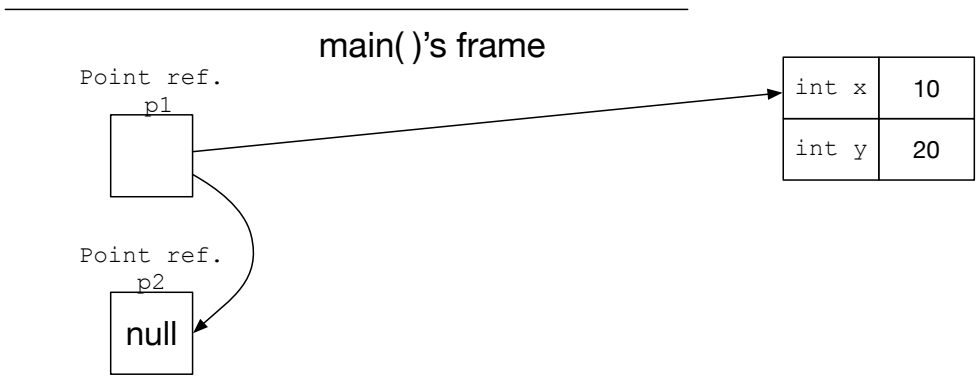
p1 now contains its location

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



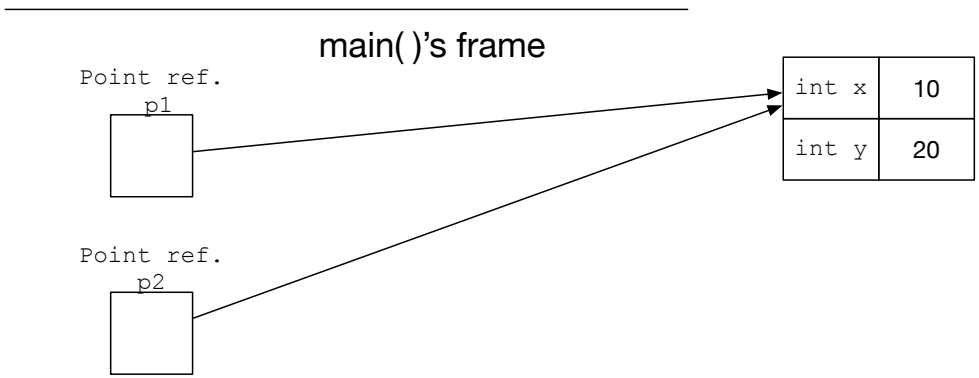
Copy what's in p1 (location of the Point) into p2

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



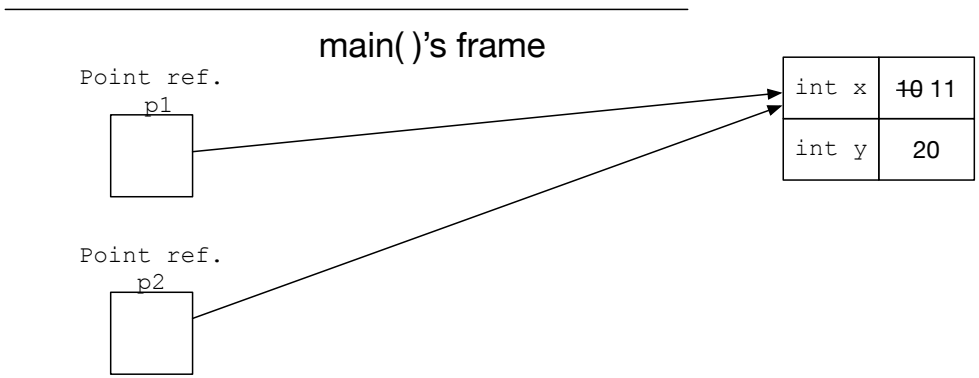
p1 and p2 reference the same Point object

```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



update the Point's x

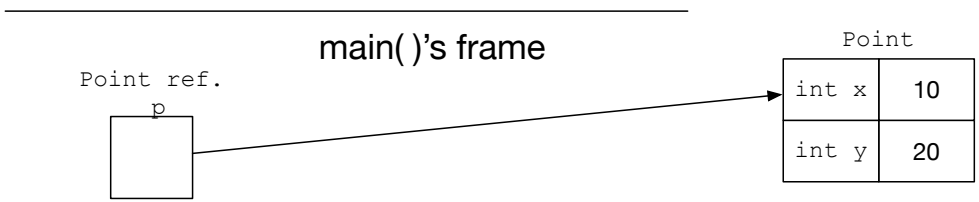
```
1   import java.awt.*;
2
3   public class Funcs04 {
4       public static void main(String args[]) {
5           Point p1 = new Point(10,20);
6           Point p2 = p1;
7
8           p2.x++;
9
10          System.out.println("p1 = " + p1 + ", p2=" + p2);
11      }
12  }
```



References and Methods

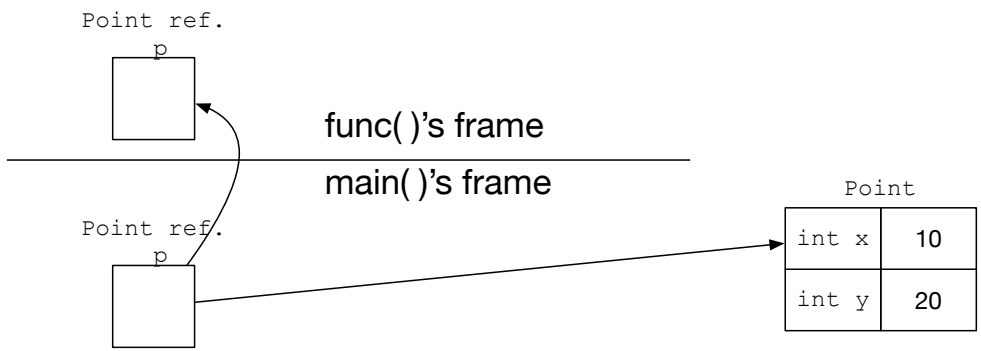
Same as before, but with a method

```
1  public class Funcs05 {
2      public static void func(Point p) {
3          p.x++;
4      }
5
6      public static void main(String args[]) {
7          Point p = new Point(10,20);
8          func(p);
9          System.out.println("p = " + p);
10     }
11 }
```



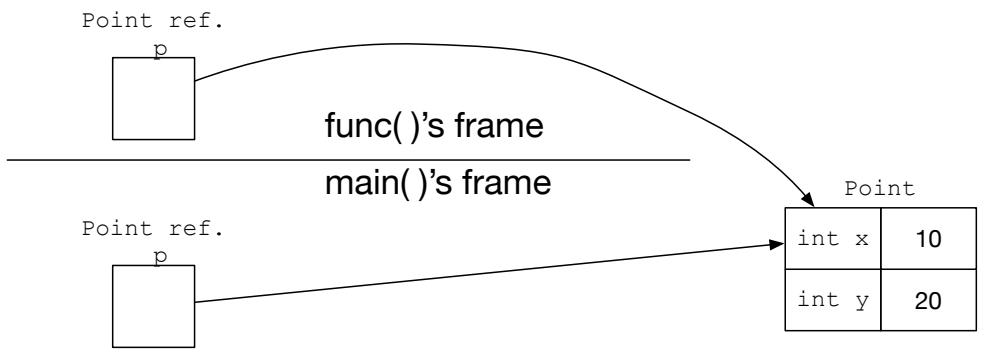
Copy what's in main's p to func's p

```
1   public class Funcs05 {
2       public static void func(Point p) {
3           p.x++;
4       }
5
6       public static void main(String args[]) {
7           Point p = new Point(10,20);
8           func(p);
9           System.out.println("p = " + p);
10      }
11  }
```



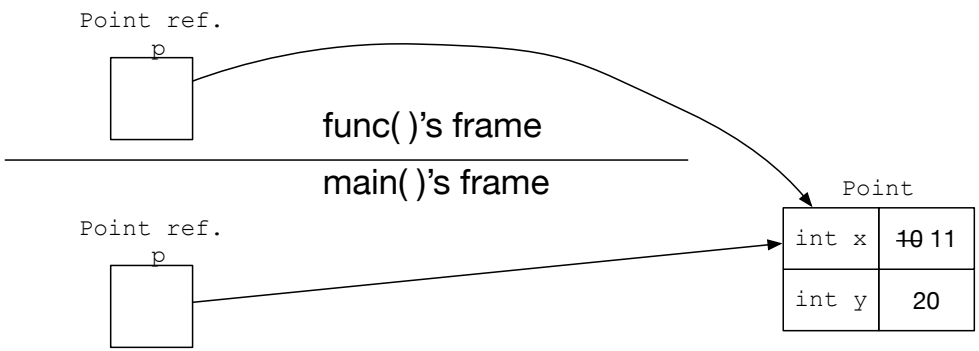
Both reference the same Point object

```
1   public class Funcs05 {
2       public static void func(Point p) {
3           p.x++;
4       }
5
6       public static void main(String args[]) {
7           Point p = new Point(10,20);
8           func(p);
9           System.out.println("p = " + p);
10      }
11  }
```



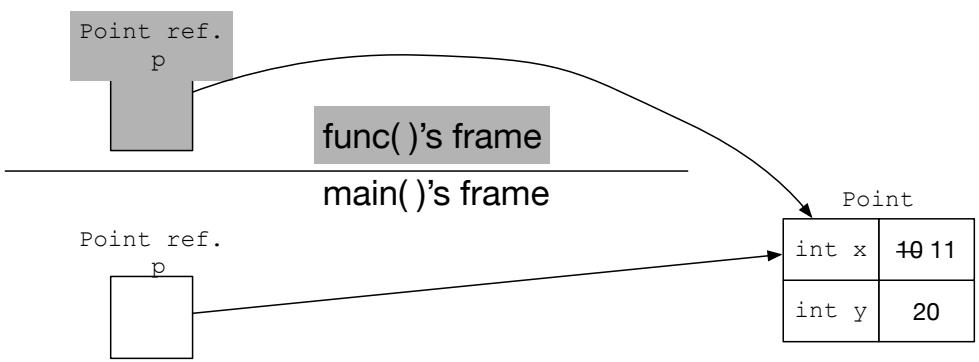
Update

```
1   public class Funcs05 {
2       public static void func(Point p) {
3           p.x++;
4       }
5
6       public static void main(String args[]) {
7           Point p = new Point(10,20);
8           func(p);
9           System.out.println("p = " + p);
10      }
11  }
```



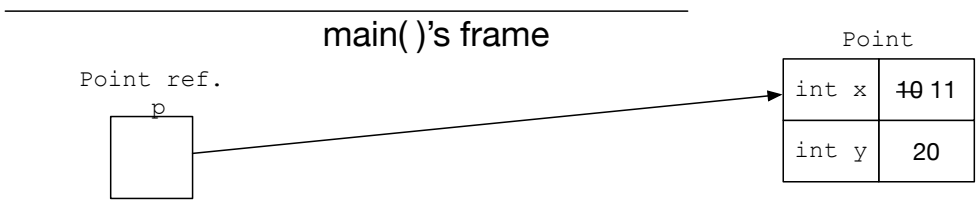
Method's Finished. Free Memory

```
1   public class Funcs05 {
2       public static void func(Point p) {
3           p.x++;
4       }
5
6       public static void main(String args[]) {
7           Point p = new Point(10,20);
8           func(p);
9           System.out.println("p = " + p);
10      }
11  }
```



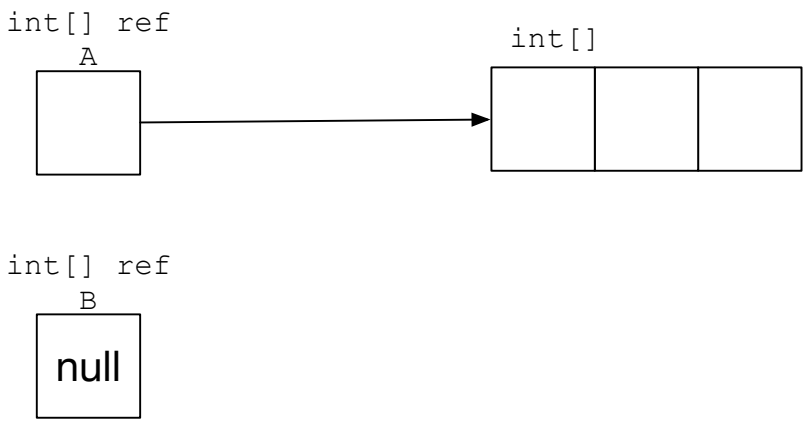
Method's Finished. Free Memory

```
1   public class Funcs05 {
2       public static void func(Point p) {
3           p.x++;
4       }
5
6       public static void main(String args[]) {
7           Point p = new Point(10,20);
8           func(p);
9           System.out.println("p = " + p);
10      }
11  }
```



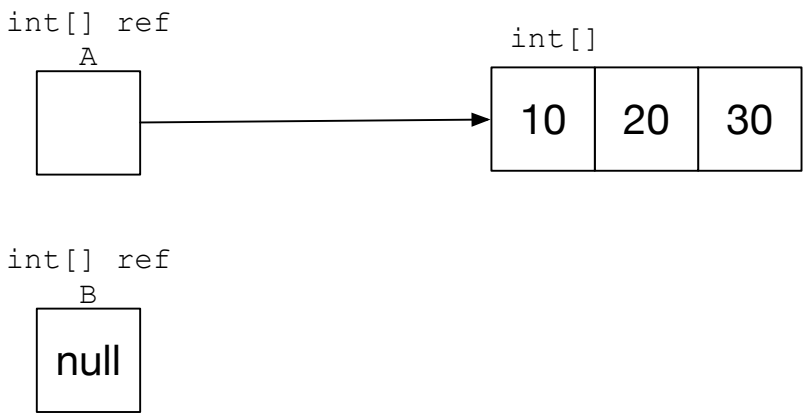
Array names are also references

```
1  import java.util.Arrays;
2
3  public class Funcs06 {
4      public static void main(String args[]) {
5          int A[] = new int[3];
6          A[0]=10; A[1]=20; A[2]=30;
7          int B[] = A;
8
9          B[0]++;
10         System.out.println("A=" + Arrays.toString(A));
11         System.out.println("B=" + Arrays.toString(B));
12     }
13 }
```



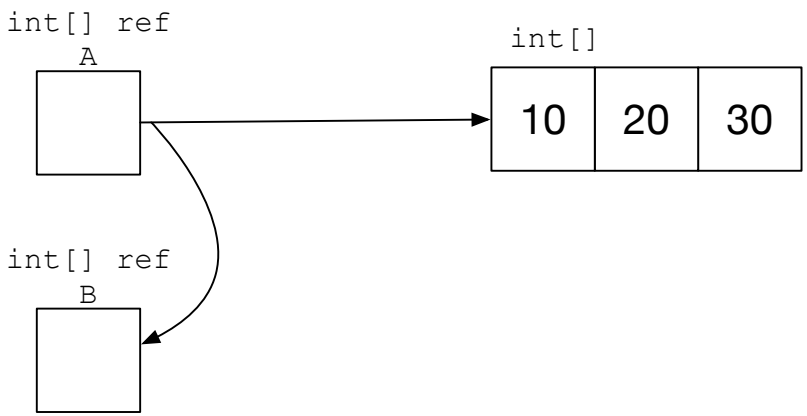
Initialize

```
1  import java.util.Arrays;
2
3  public class Funcs06 {
4      public static void main(String args[]) {
5          int A[] = new int[3];
6          A[0]=10; A[1]=20; A[2]=30;
7          int B[] = A;
8
9          B[0]++;
10         System.out.println("A=" + Arrays.toString(A));
11         System.out.println("B=" + Arrays.toString(B));
12     }
13 }
```



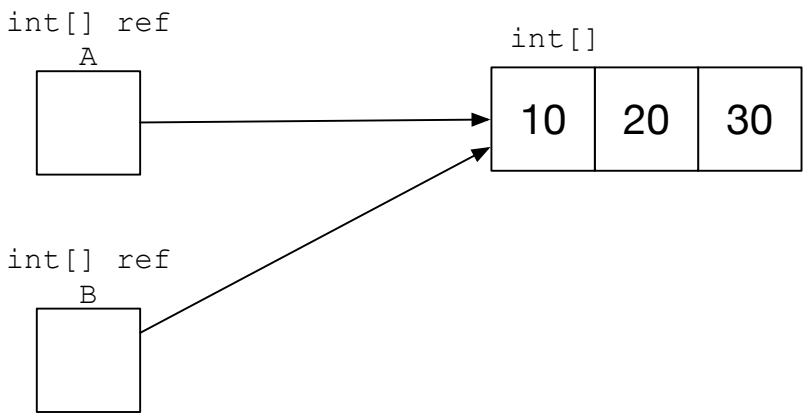
Copy What's in A into B

```
1  import java.util.Arrays;
2
3  public class Funcs06 {
4      public static void main(String args[]) {
5          int A[] = new int[3];
6          A[0]=10; A[1]=20; A[2]=30;
7          int B[] = A;
8
9          B[0]++;
10         System.out.println("A=" + Arrays.toString(A));
11         System.out.println("B=" + Arrays.toString(B));
12     }
13 }
```



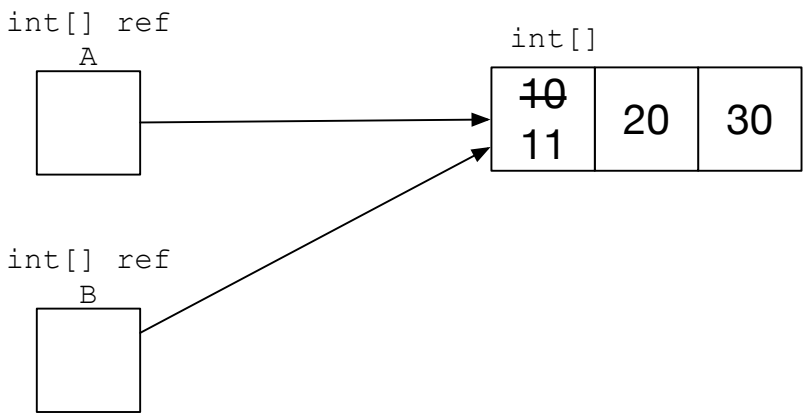
A and B refer to the same array

```
1  import java.util.Arrays;
2
3  public class Funcs06 {
4      public static void main(String args[]) {
5          int A[] = new int[3];
6          A[0]=10; A[1]=20; A[2]=30;
7          int B[] = A;
8
9          B[0]++;
10         System.out.println("A=" + Arrays.toString(A));
11         System.out.println("B=" + Arrays.toString(B));
12     }
13 }
```



Update

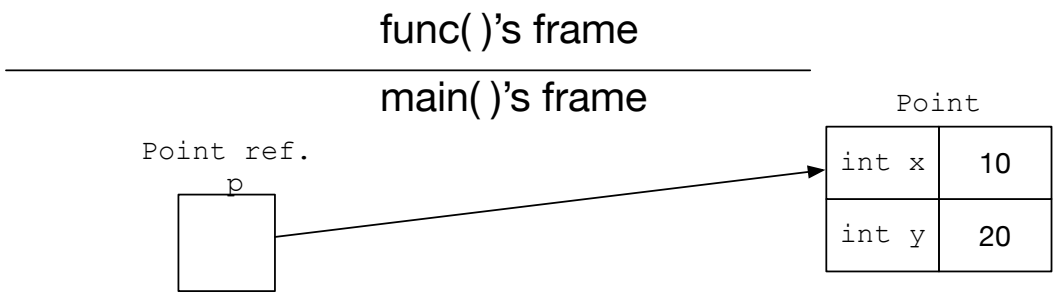
```
1  import java.util.Arrays;
2
3  public class Funcs06 {
4      public static void main(String args[]) {
5          int A[] = new int[3];
6          A[0]=10; A[1]=20; A[2]=30;
7          int B[] = A;
8
9          B[0]++;
10         System.out.println("A=" + Arrays.toString(A));
11         System.out.println("B=" + Arrays.toString(B));
12     }
13 }
```



More References and Methods

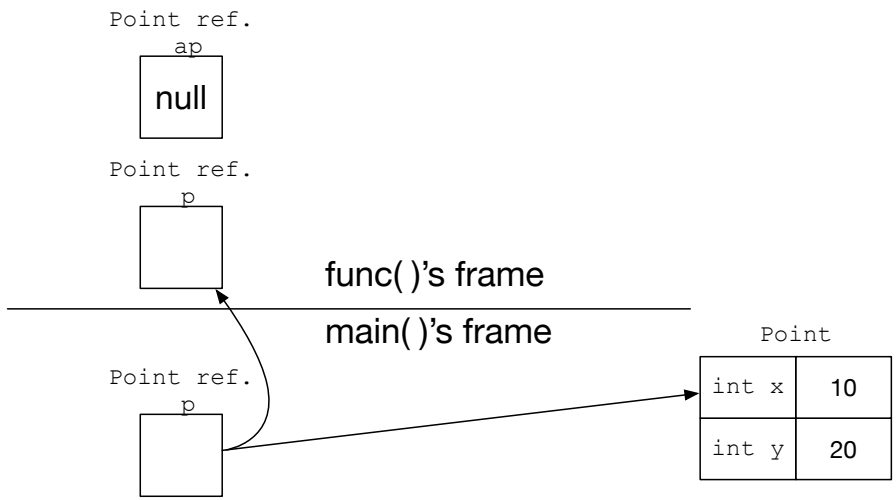
p has location of the Point object

```
1   import java.awt.*;
2
3   public class Funcs07 {
4       public static void func(Point p) {
5           Point ap = new Point(p.x+1, p.y+1);
6           p = ap;
7       }
8
9       public static void main(String args[]) {
10          Point p = new Point(10,20);
11          func(p);
12          System.out.println("p = " + p);
13      }
14  }
```



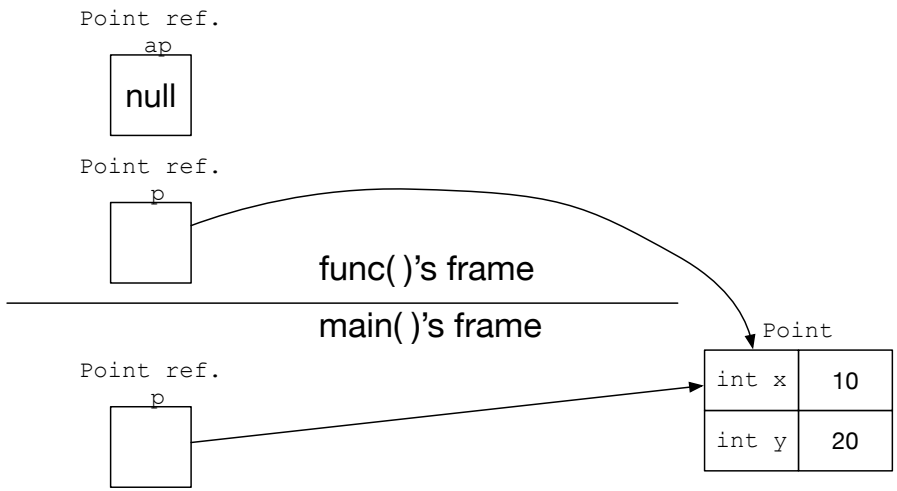
Copy main's p into func's p

```
1  public class Funcs07 {
2      public static void func(Point p) {
3          Point ap = new Point(p.x+1, p.y+1);
4          p = ap;
5      }
6
7      public static void main(String args[]) {
8          Point p = new Point(10,20);
9          func(p);
10         System.out.println("p = " + p);
11     }
12 }
```



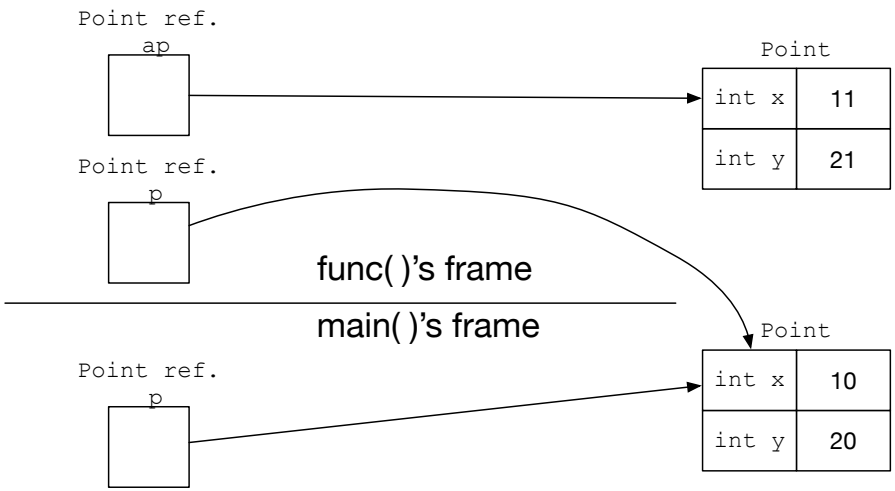
Both refer to same Point object

```
1  public class Funcs07 {
2      public static void func(Point p) {
3          Point ap = new Point(p.x+1, p.y+1);
4          p = ap;
5      }
6
7      public static void main(String args[]) {
8          Point p = new Point(10,20);
9          func(p);
10         System.out.println("p = " + p);
11     }
12 }
```



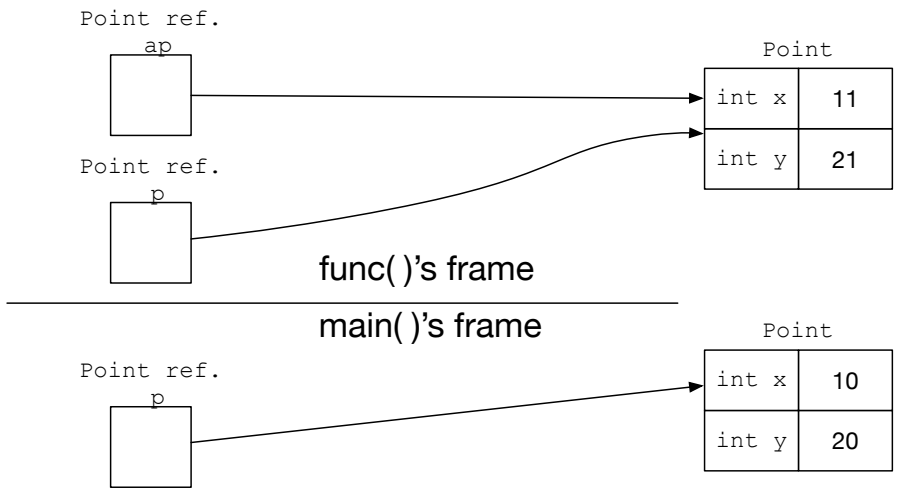
Create another Point. ap contains its location

```
1   public class Funcs07 {
2       public static void func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           p = ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



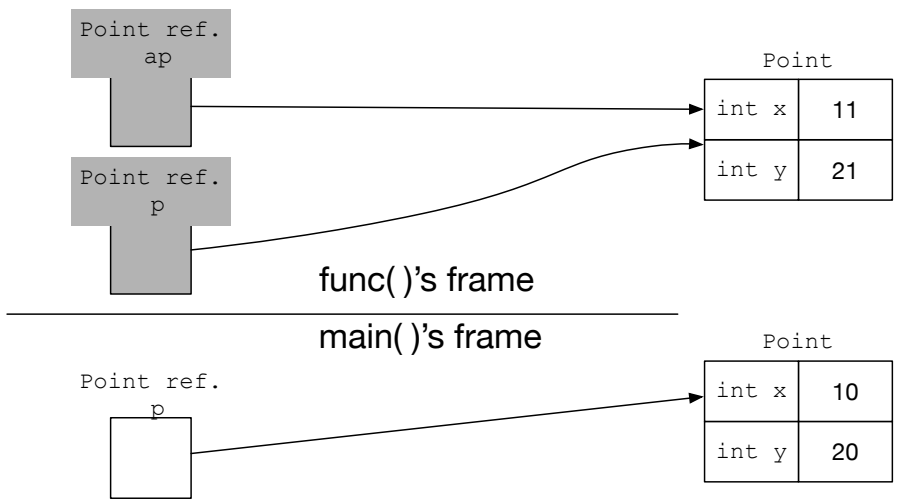
Copy ap into p. It's func's p, not main's!

```
1   public class Funcs07 {
2       public static void func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           p = ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



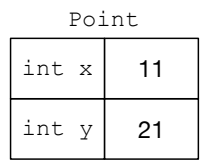
func's finished. free its p, ap

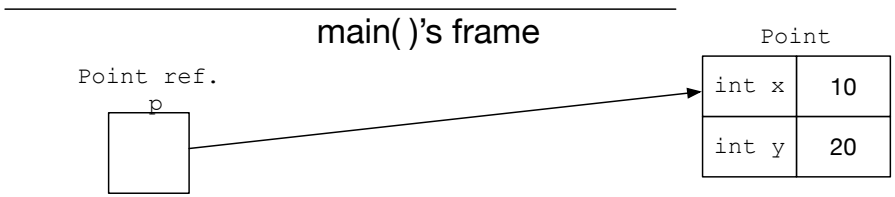
```
1   public class Funcs07 {
2       public static void func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           p = ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



func's finished. free its p, ap

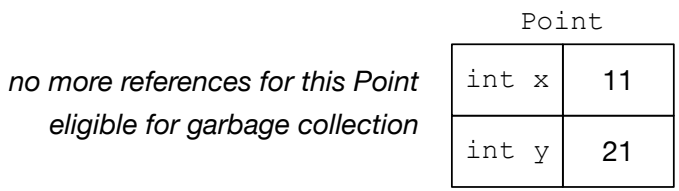
```
1   public class Funcs07 {
2       public static void func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           p = ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           func(p);
10          System.out.println("p = " + p);
11      }
12  }
```

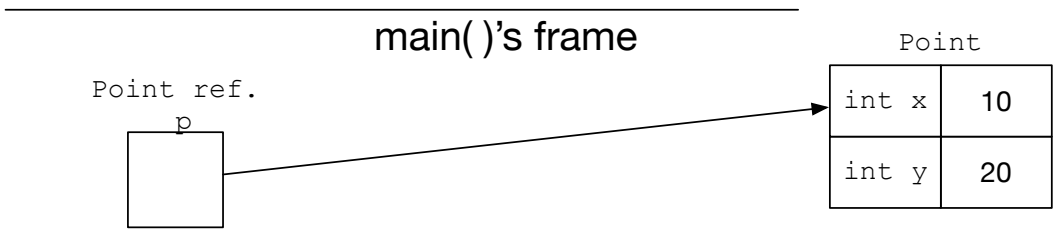




Nothing refers to 2nd Point

```
1  public class Funcs07 {
2      public static void func(Point p) {
3          Point ap = new Point(p.x+1, p.y+1);
4          p = ap;
5      }
6
7      public static void main(String args[]) {
8          Point p = new Point(10,20);
9          func(p);
10         System.out.println("p = " + p);
11     }
12 }
```





without a return statement

within a method, there's absolutely no way to change which `Point` `main`'s `p` references

returns

calling a method with parameters

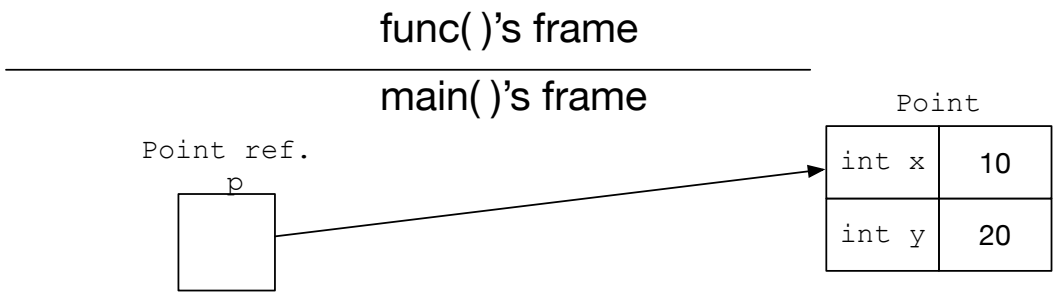
copy value of arguments to method

returns from methods with return statements

copy value back to caller

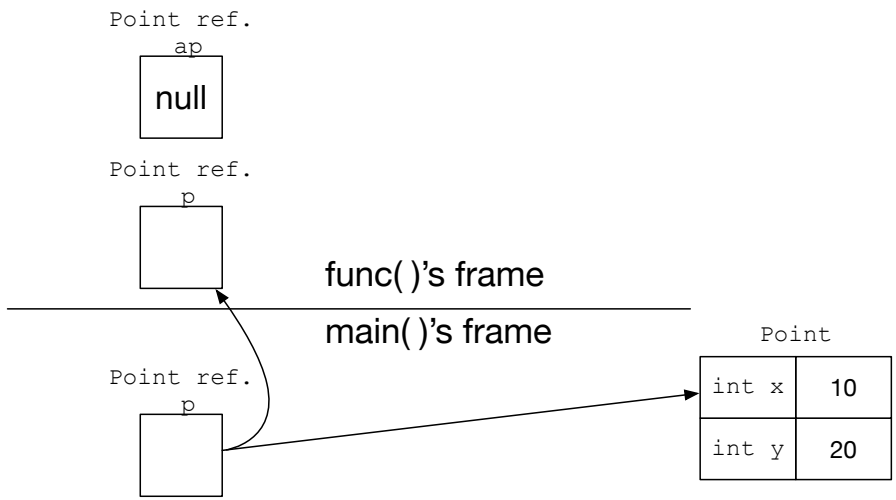
p contains location of Point object

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



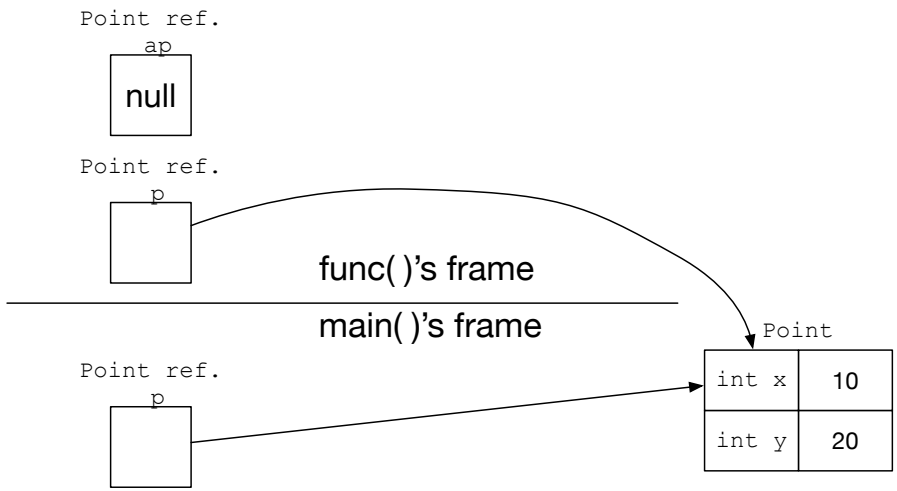
copy main's p into func's p

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



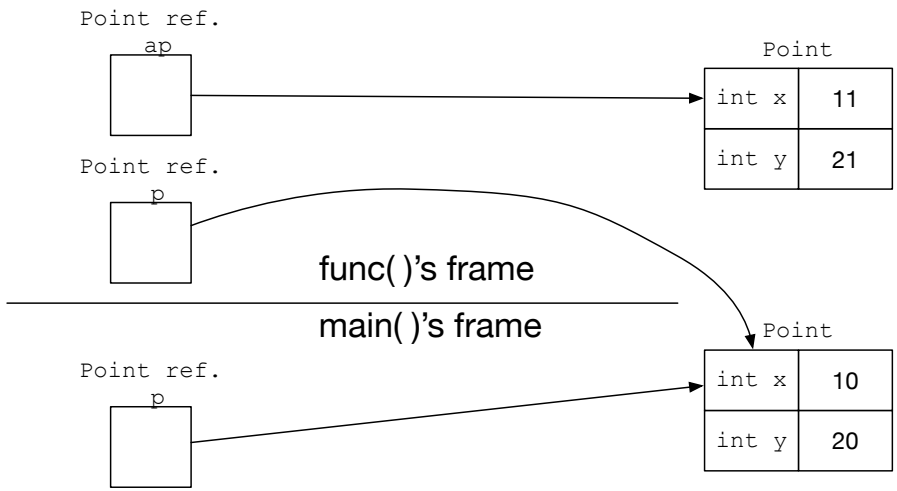
Both p's point to same Point object

```
1  public class Funcs08 {
2      public static Point func(Point p) {
3          Point ap = new Point(p.x+1, p.y+1);
4          return ap;
5      }
6
7      public static void main(String args[]) {
8          Point p = new Point(10,20);
9          p=func(p);
10         System.out.println("p = " + p);
11     }
12 }
```



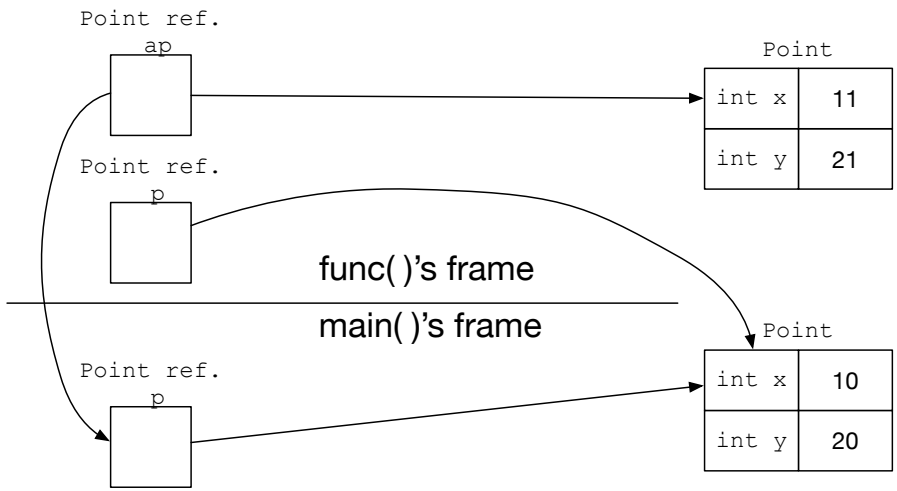
Create new Point

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



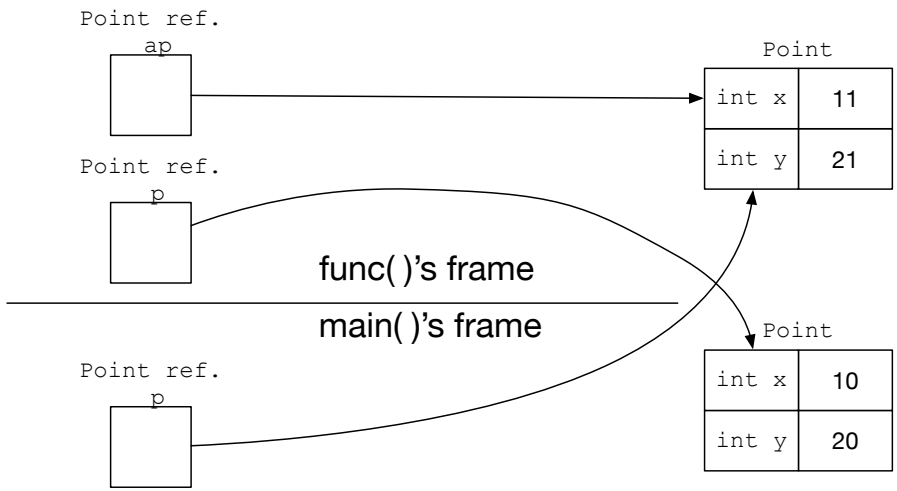
Return. Copy ap into main's p

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



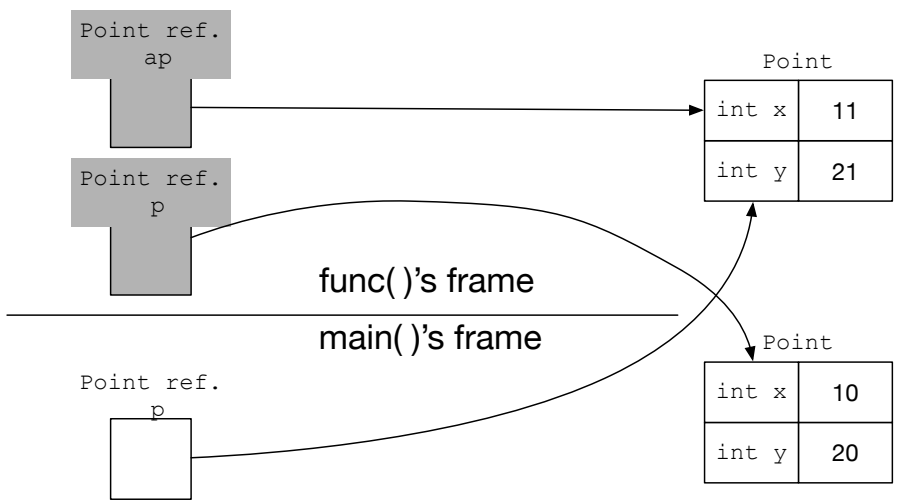
ap and main's p both refer to same Point object

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



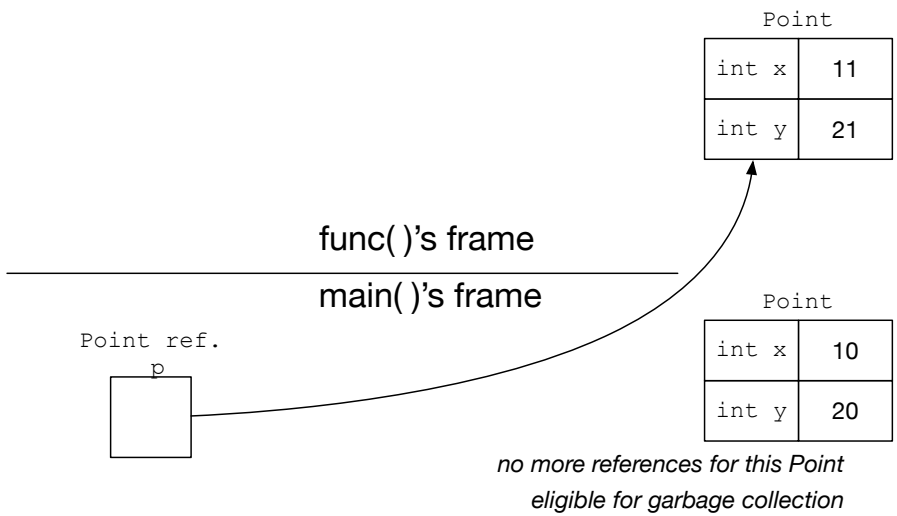
func's finished. free its p, ap

```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



main's p changed. nothing refers to old Point

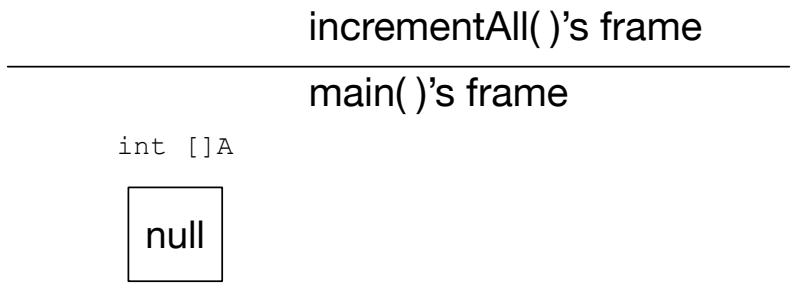
```
1   public class Funcs08 {
2       public static Point func(Point p) {
3           Point ap = new Point(p.x+1, p.y+1);
4           return ap;
5       }
6
7       public static void main(String args[]) {
8           Point p = new Point(10,20);
9           p=func(p);
10          System.out.println("p = " + p);
11      }
12  }
```



Array names are references

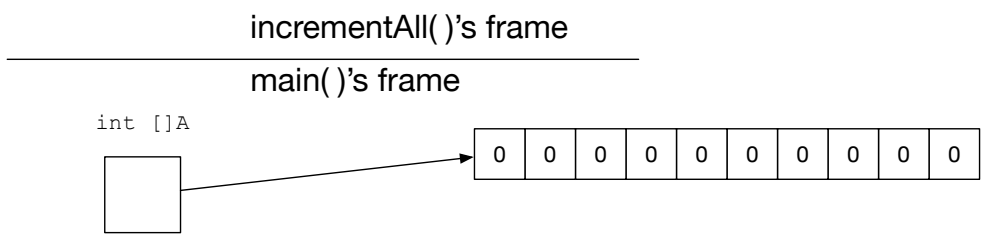
A is a reference to an array, *not* an array

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



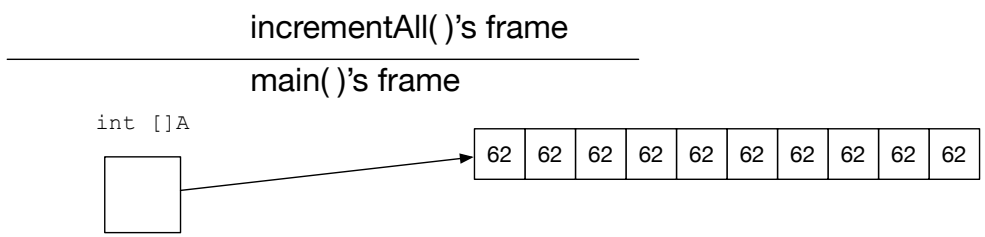
new operator creates the array

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



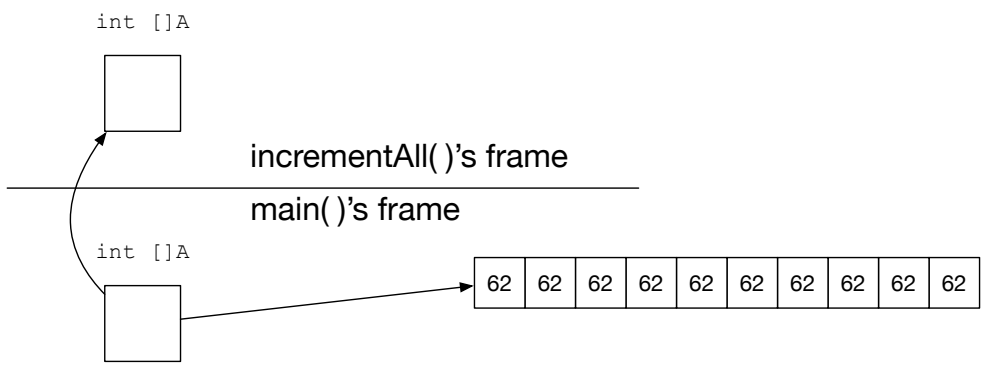
Update

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



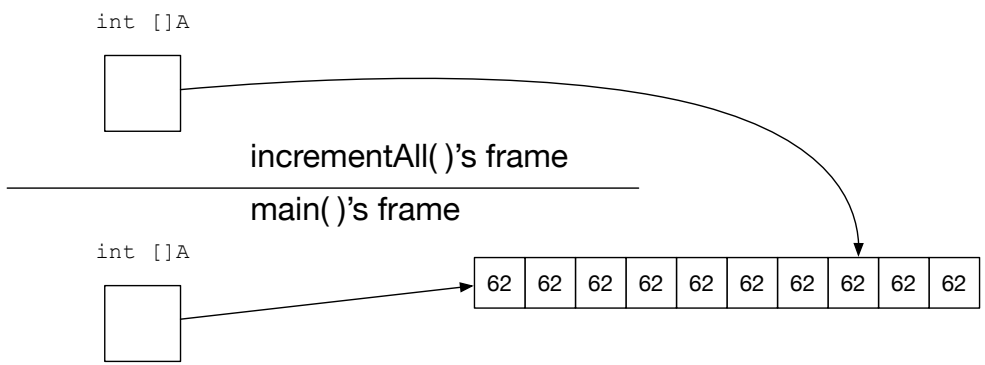
Copy

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



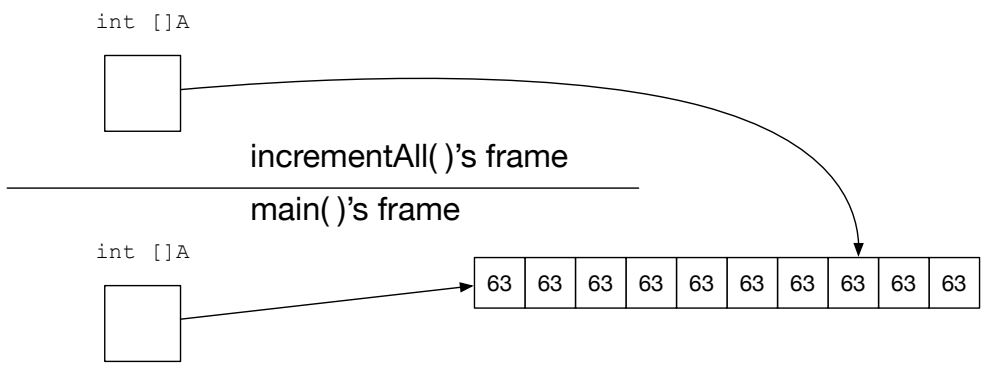
Both references point to same array

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



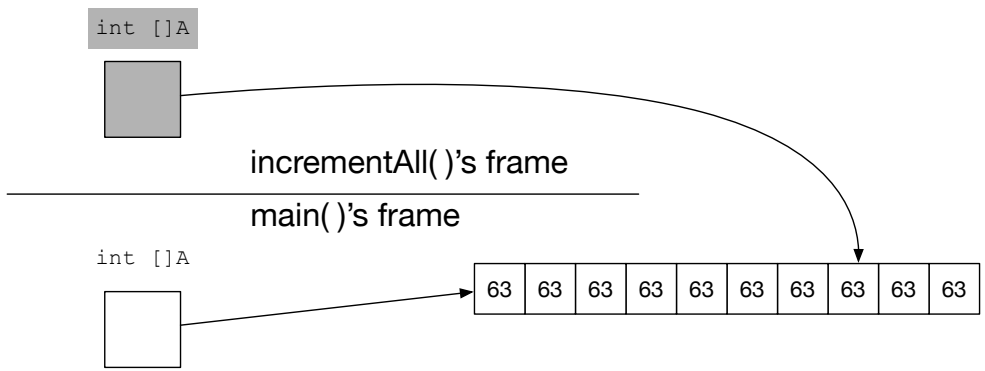
Update

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



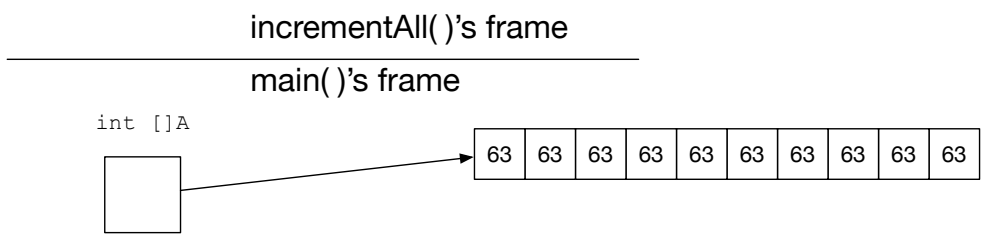
incrementAll finished. free memory

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```



incrementAll finished. free memory

```
1   public class Funcs09 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           for (int i=0; i<A.length; i++) {
6               A[i]++;
7           }
8       }
9
10      public static void main(String args[]) {
11          int A[] = new int[SIZE];
12          for (int i=0; i<A.length; i++) {
13              A[i]=62;
14          }
15          incrementAll(A);
16          System.out.println(Arrays.toString(A));
17      }
18  }
```

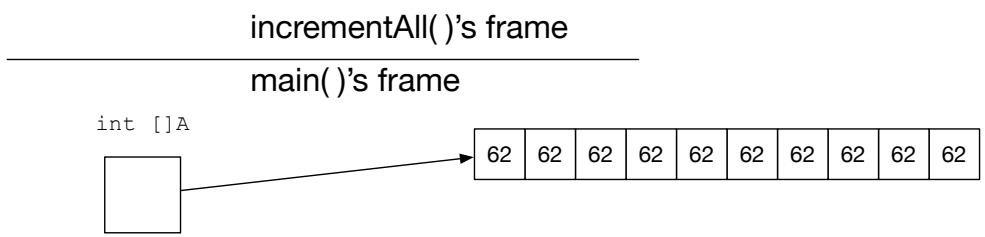


without a return statement

within a method, there's absolutely no way to change which array `main`'s `A` references

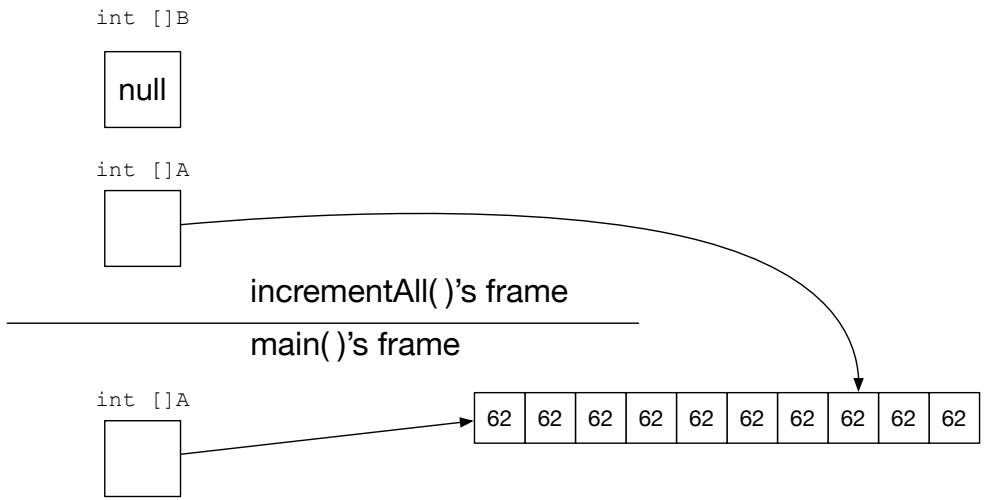
Initialize

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



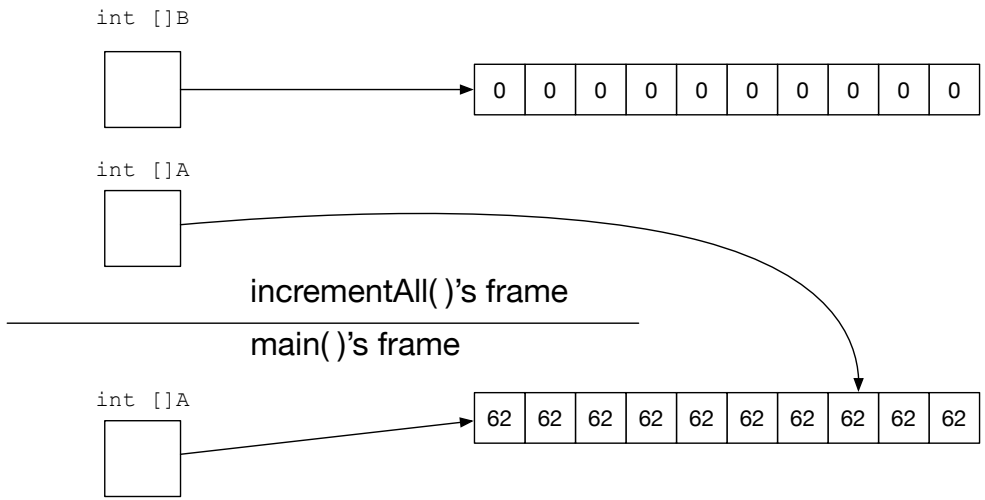
Copy reference

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



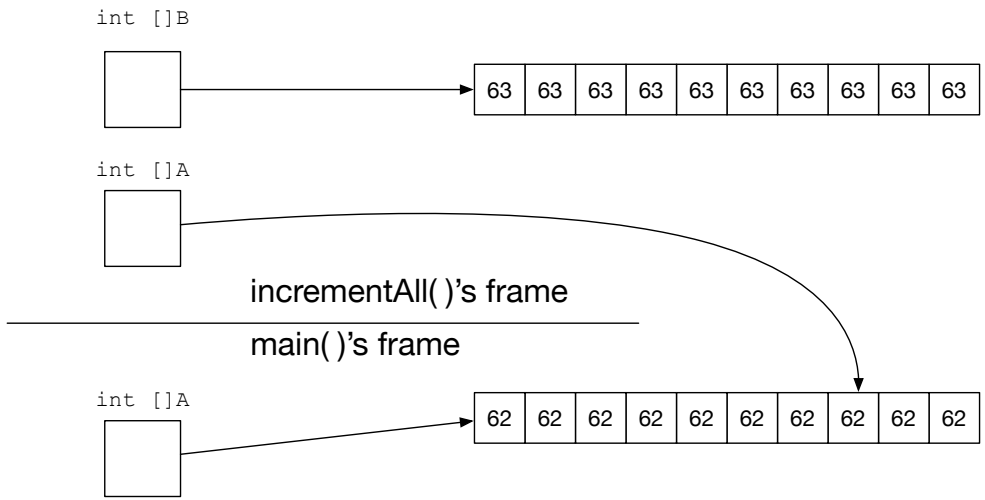
Create new array

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



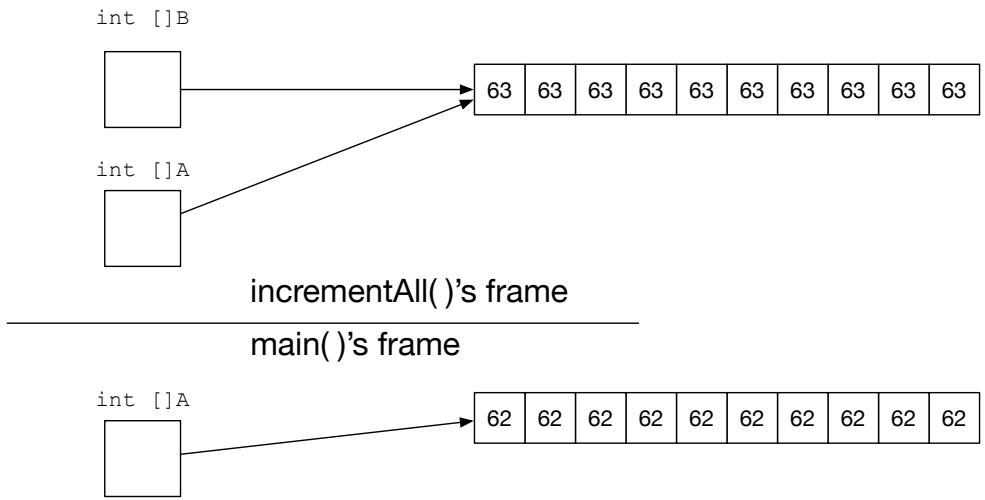
Modify new array

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



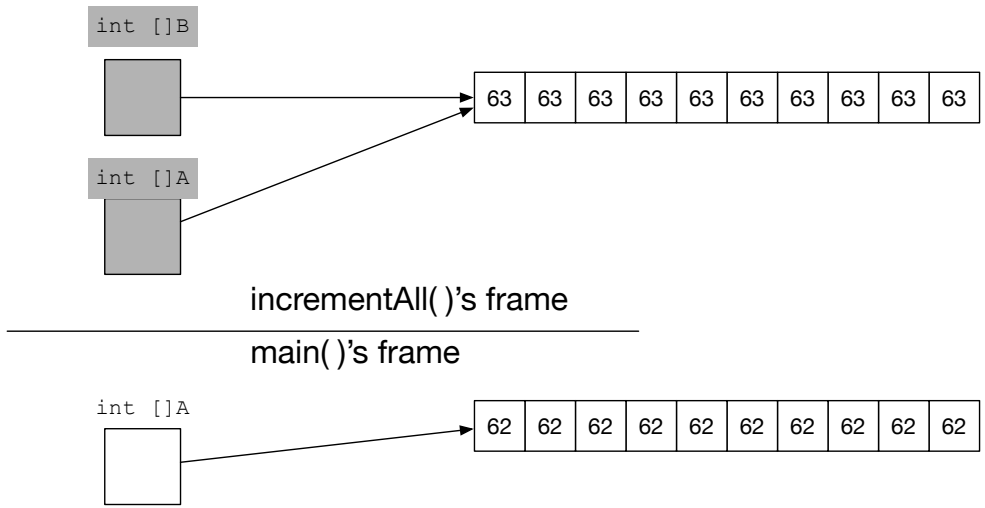
Change reference. incrementAll's A changed, *not* main's

```
 1  public class Funcs10 {
 2      public static final int SIZE=10;
 3
 4      public static void incrementAll(int A[]) {
 5          int B[] = new int[A.length];
 6          for (int i=0; i<A.length; i++) {
 7              B[i]=A[i]+1;
 8          }
 9          A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



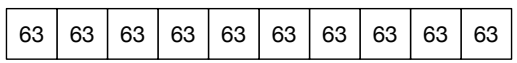
method finished. free its A and B

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



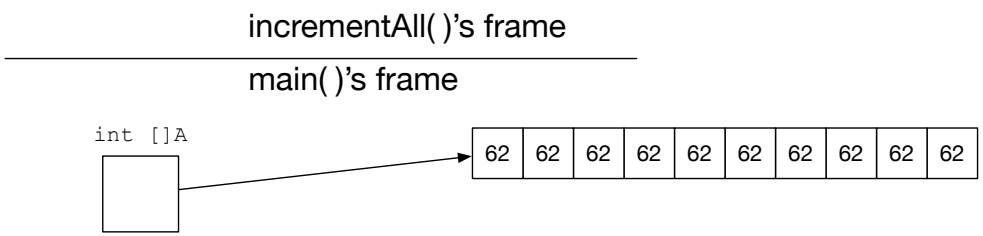
method finished. free its A and B

```
1   public class Funcs10 {
2       public static final int SIZE=10;
3
4       public static void incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           A=B;
10      }
11
12      public static void main(String args[]) {
13          int A[] = new int[SIZE];
14          for (int i=0; i<A.length; i++) {
15              A[i]=62;
16          }
17          incrementAll(A);
18          System.out.println(Arrays.toString(A));
19      }
20  }
```



nothing pointing to this

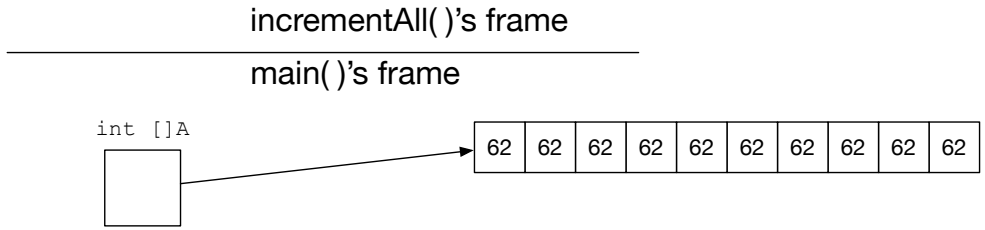
eligible for garbage collection



With a return, copy value back to caller

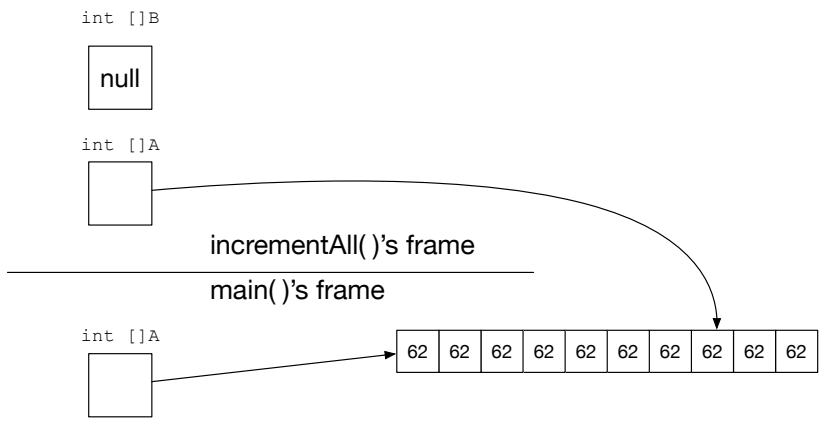
Same but with a return

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



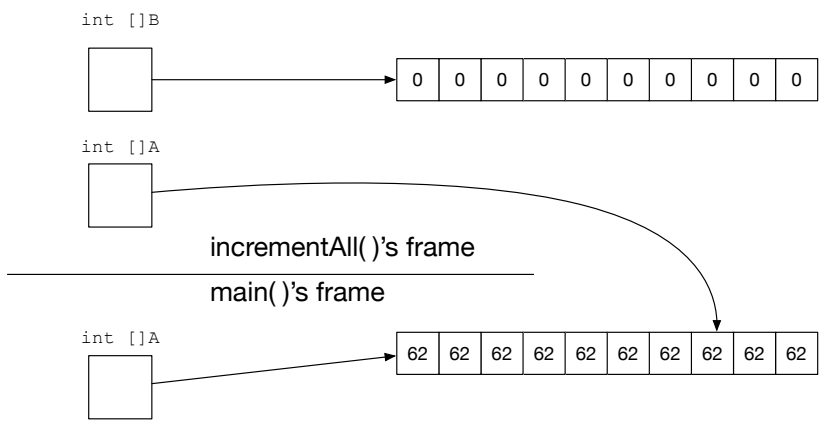
Copy reference

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



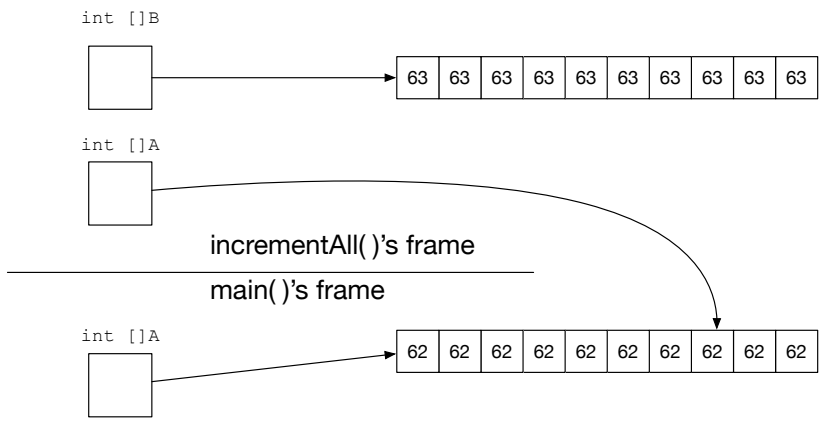
Create new array

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



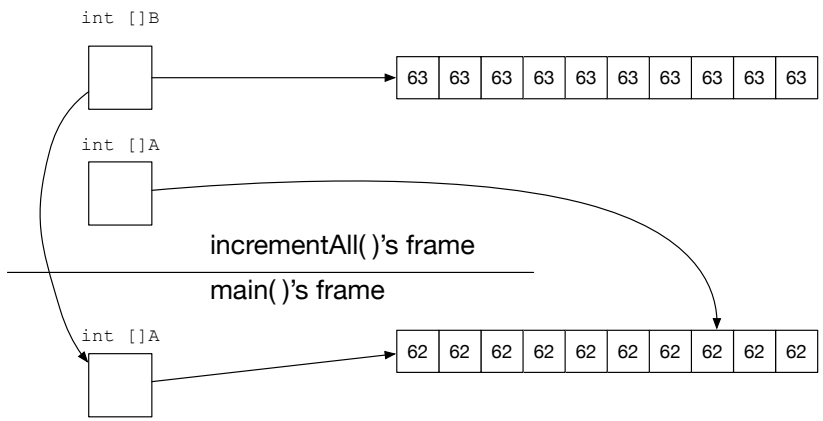
Update new array

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



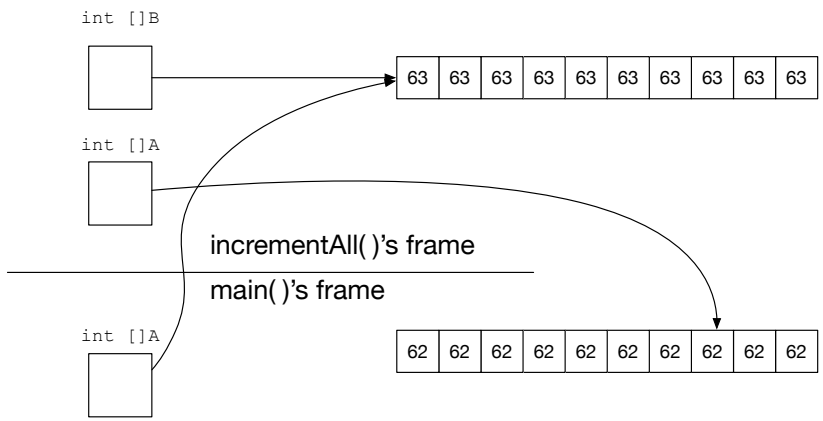
Return. Copy back reference

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



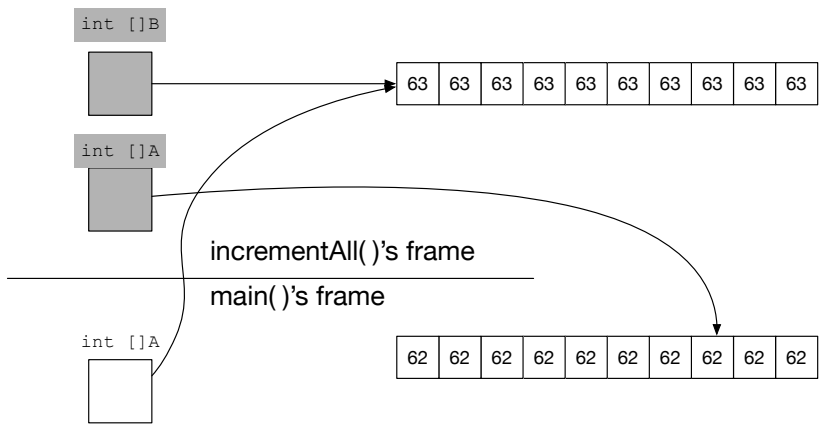
Return. Copy back reference

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



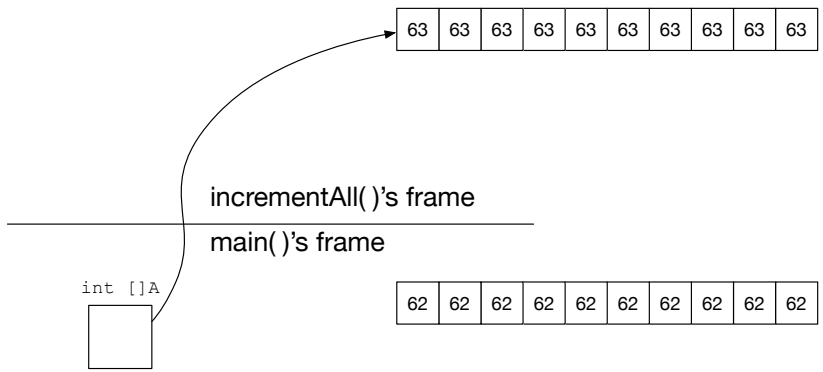
Method's finished. Free its A, B.

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



Method's finished. Free its A, B.

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```



Nothing left pointing to the original

```
1   public class Funcs11 {
2       public static final int SIZE=10;
3
4       public static int[] incrementAll(int A[]) {
5           int B[] = new int[A.length];
6           for (int i=0; i<A.length; i++) {
7               B[i]=A[i]+1;
8           }
9           return B;
10      }
11
12
13      public static void main(String args[]) {
14          int[] A = new int[SIZE];
15          for (int i=0; i<A.length; i++) {
16              A[i]=62;
17          }
18          A=incrementAll(A);
19          System.out.println(Arrays.toString(A));
20      }
21  }
```