More about OOP

COMP2026

PROBLEM SOLVING USING OBJECT ORIENTED PROGRAMMING

Overview

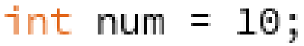
- Primitive type vs Reference type
- Memory Models

Primitives vs. References

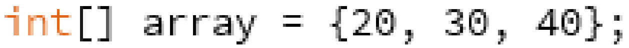
- **Primitive variables store values**
  - byte, short, int, long, float, double, boolean, char

- **Reference variable store address**
  - Scanner, String, int[], double[], etc.

Primitives vs. References

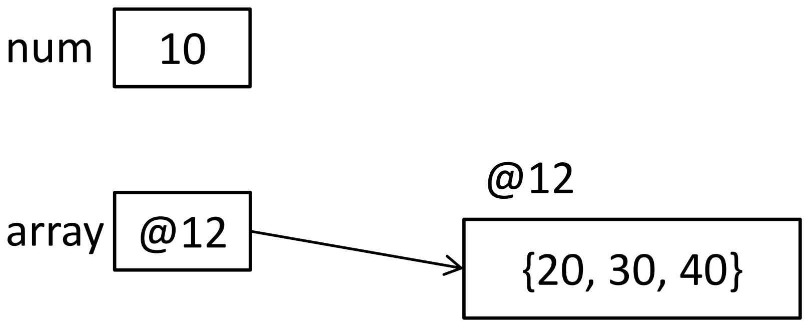


```
int num = 10;
```



```
int[] array = {20, 30, 40};
```

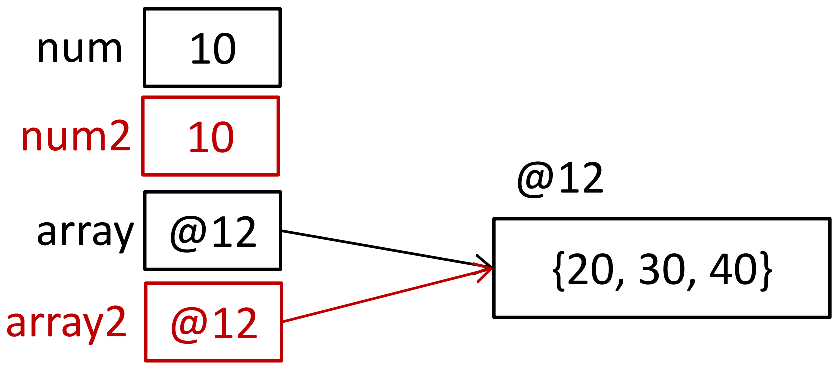
Memory Model



Primitives vs. References

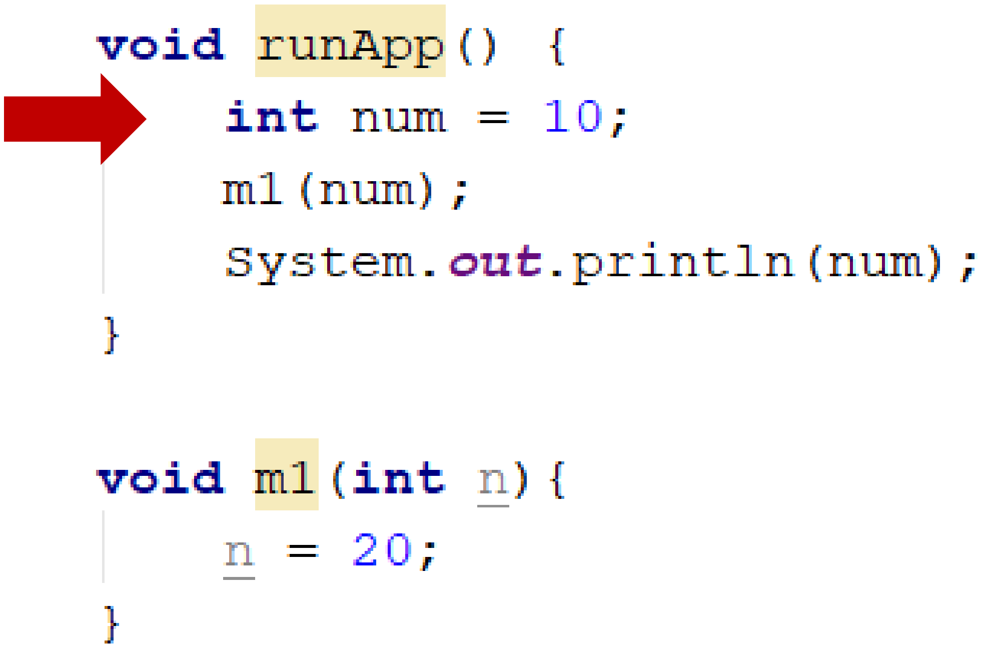
```
int num = 10;

int[] array = {20, 30, 40};

int num2 = num;

int[] array2 = array;
```

Memory Model



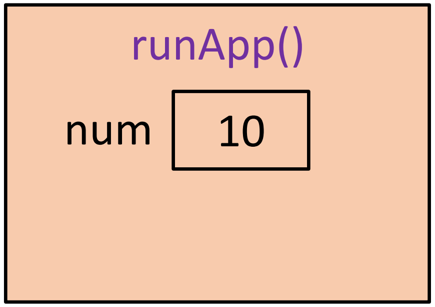
Primitive type: Value is copied
Reference type: Address is copied

Calling Method I

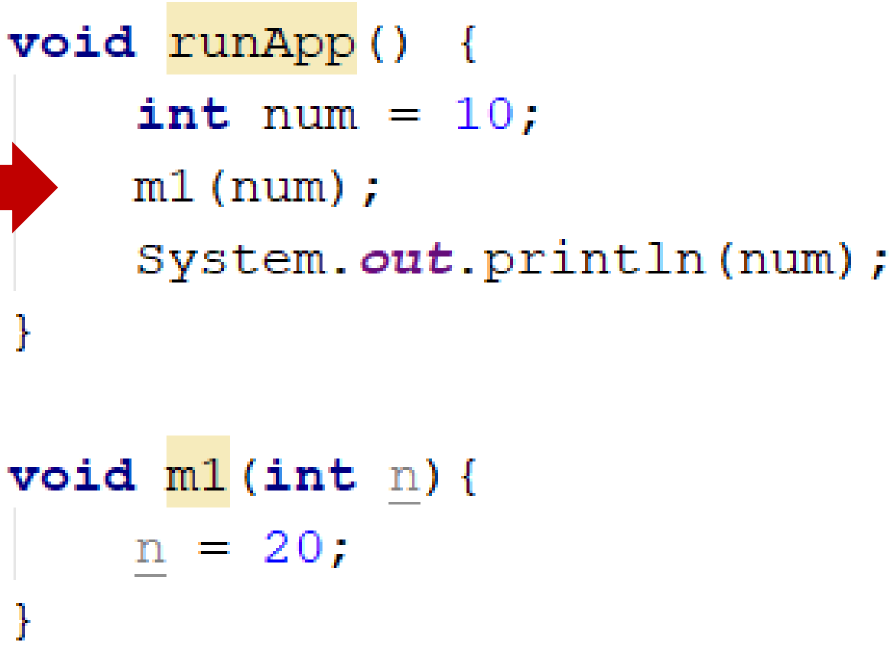


```
void runApp() {
    int num = 10;
    m1(num);
    System.out.println(num);
}

void m1(int n){
    n = 20;
}
```

Memory Model

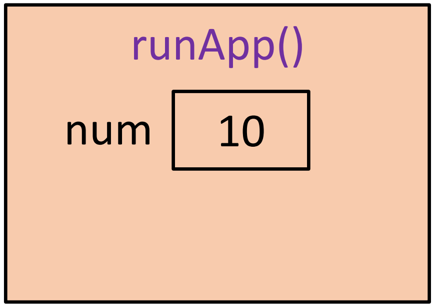


Calling Method I

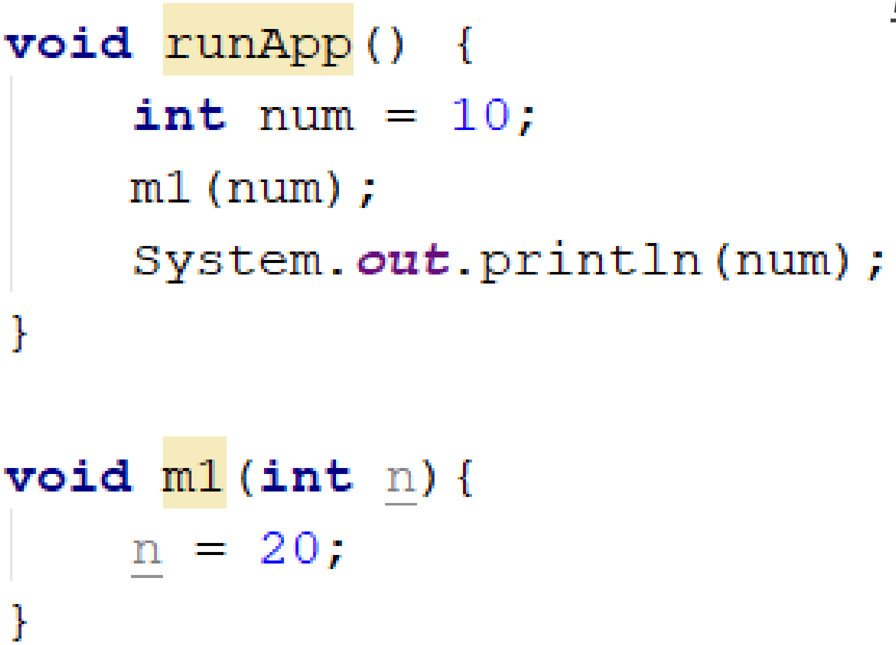


```
void runApp() {
    int num = 10;
    m1(num);
    System.out.println(num);
}

void m1(int n){
    n = 20;
}
```

Memory Model

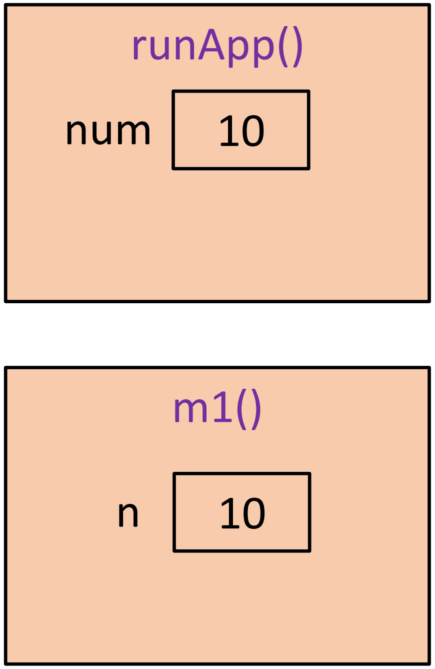


Calling Method I



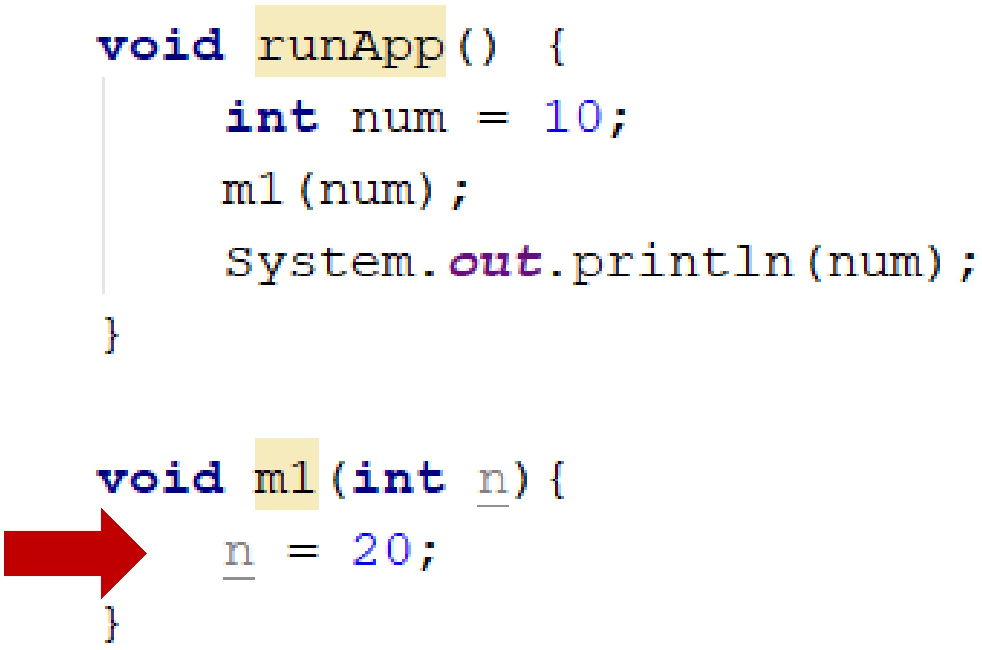
```
void runApp() {
    int num = 10;
    m1(num);
    System.out.println(num);
}

void m1(int n){
    n = 20;
}
```

Memory Model



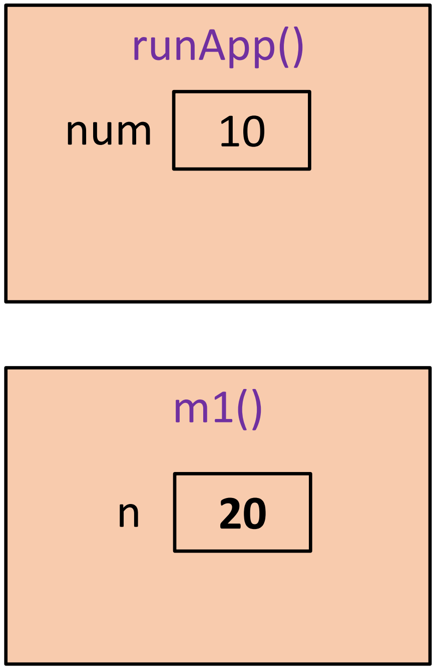
Value of **num** is copied into **n**

Calling Method I



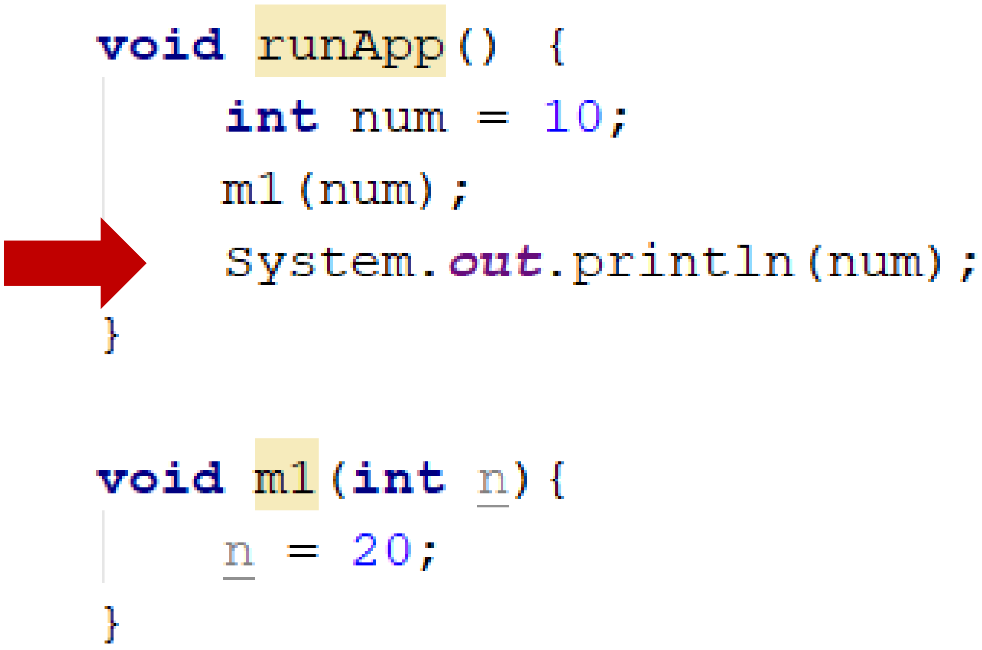
```
void runApp()  {
     int num = 10;
     m1(num);
     System.out.println(num);
}

void m1(int n){
     n = 20;
}
```

Memory Model



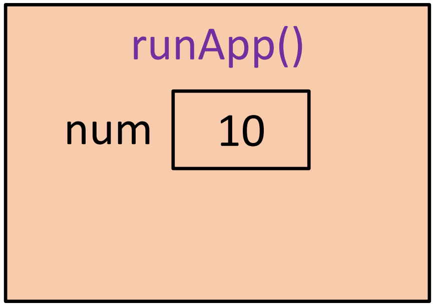
Value of **n** changes to 20

Calling Method I



```
void runApp() {
    int num = 10;
    m1(num);
    System.out.println(num);
}

void m1(int n){
    n = 20;
}
```

Memory Model

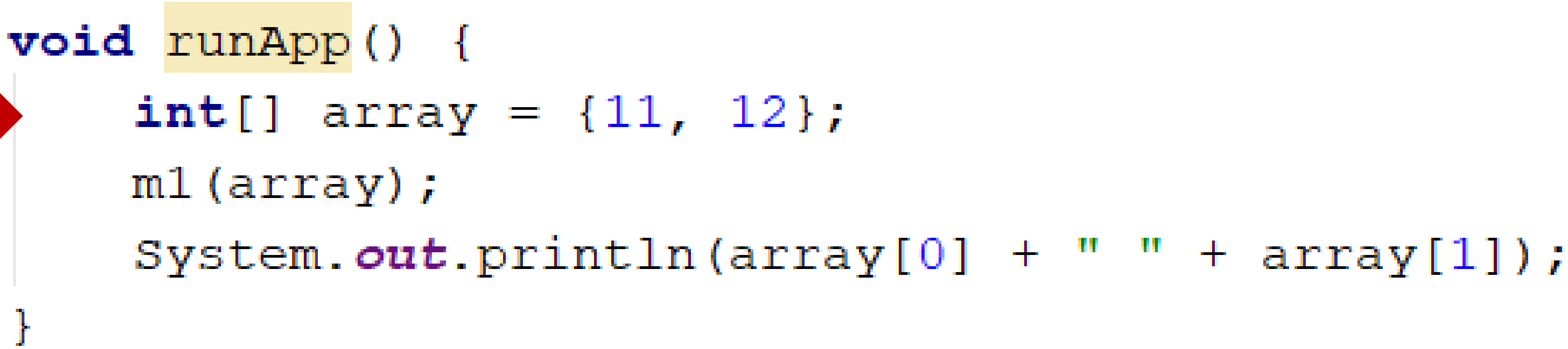


runApp()

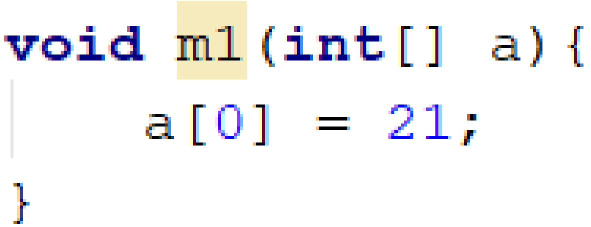
num 10

All the variables in m1() are no longer reachable

Calling Method II

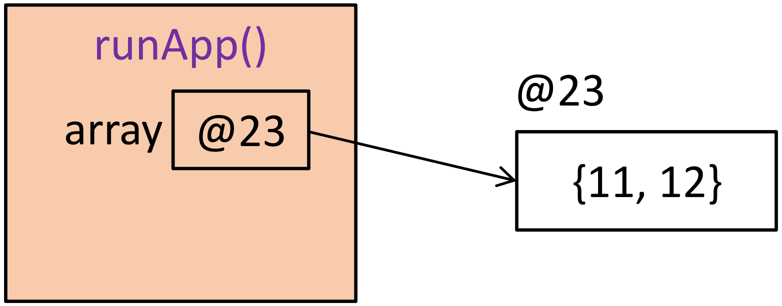


```
void runApp() {
    int[] array = {11, 12};
    m1(array);
    System.out.println(array[0] + " " + array[1]);
}
```

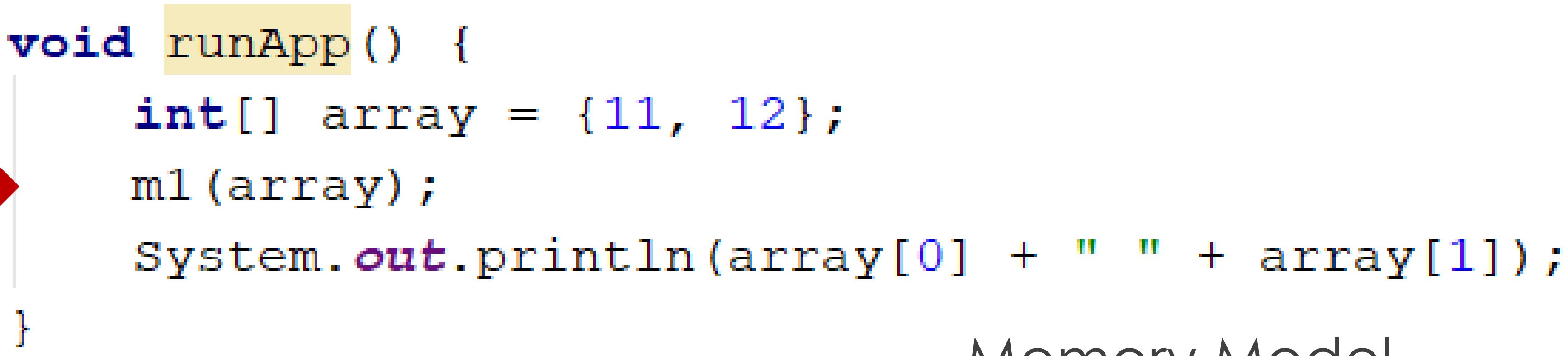


```
void m1(int[] a){
    a[0] = 21;
}
```

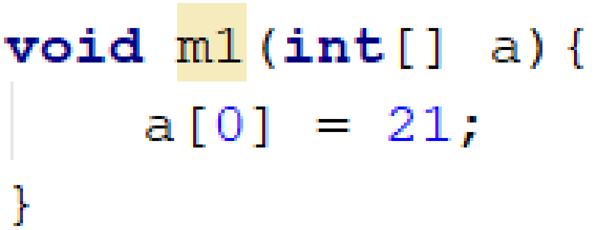
Memory Model



Calling Method II

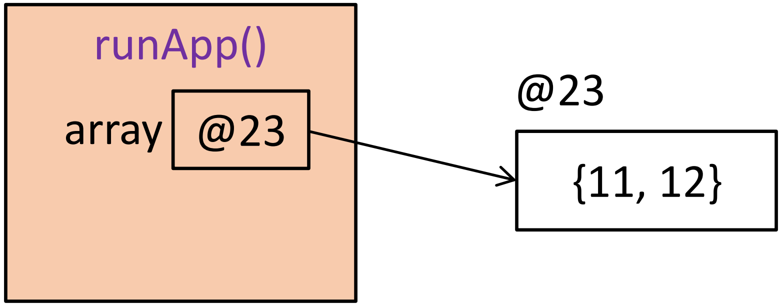


```
void runApp() {
    int[] array = {11, 12};
    m1(array);
    System.out.println(array[0] + " " + array[1]);
}
```

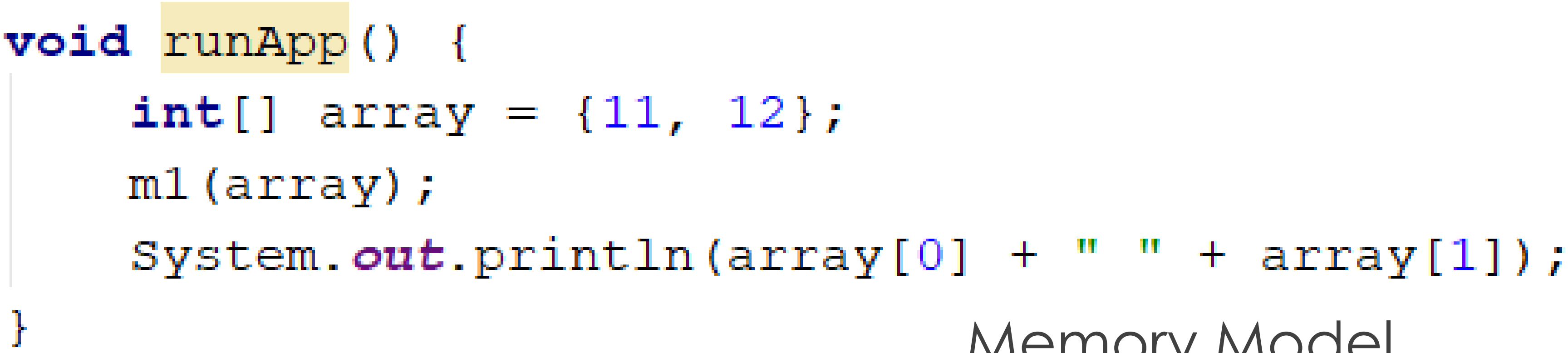


```
void m1(int[] a){
    a[0] = 21;
}
```

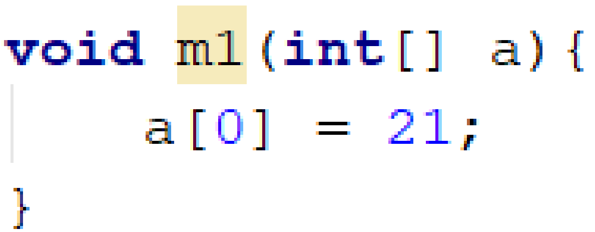
Memory Model



Calling Method II

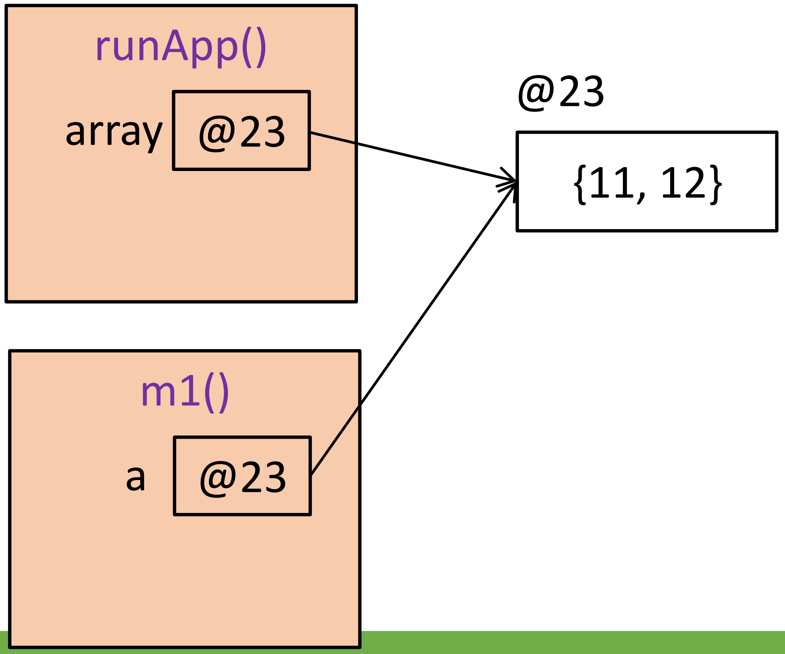


```
void runApp() {
    int[] array = {11, 12};
    m1(array);
    System.out.println(array[0] + " " + array[1]);
}
```



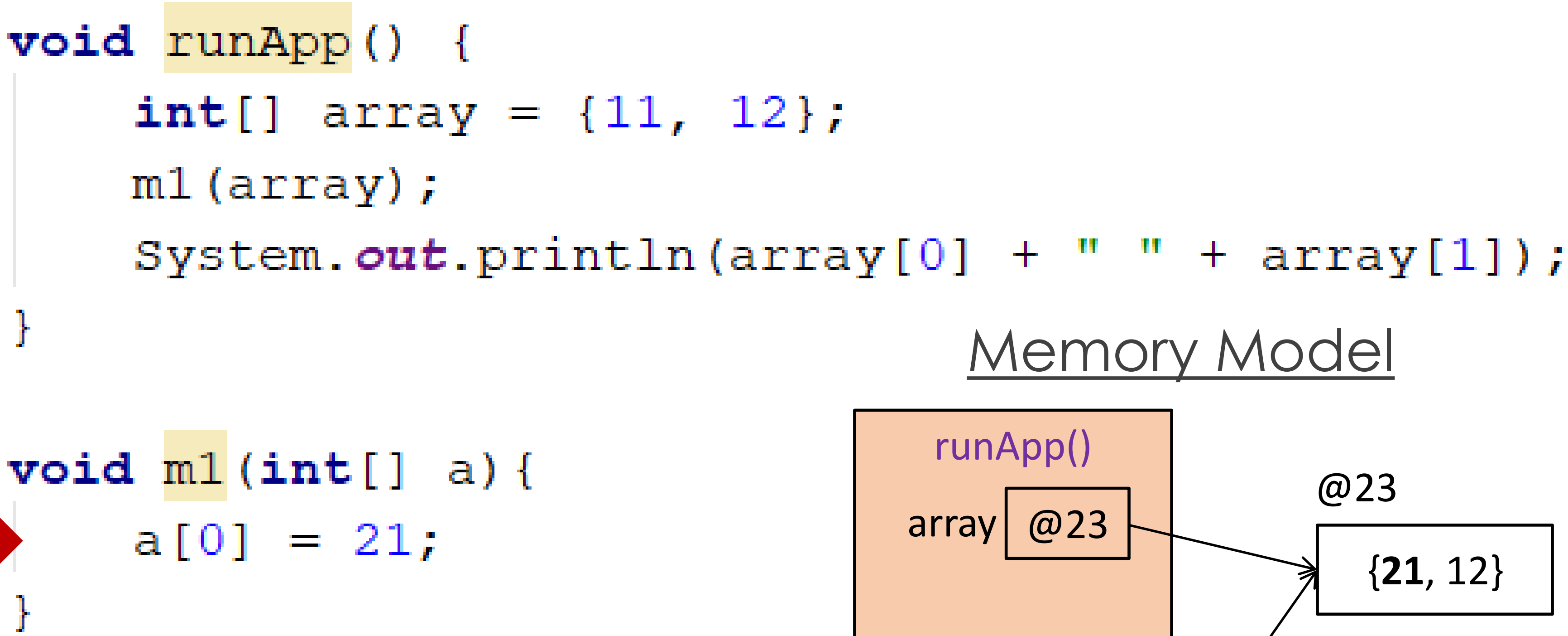
```
void m1(int[] a){
    a[0] = 21;
}
```

Memory Model



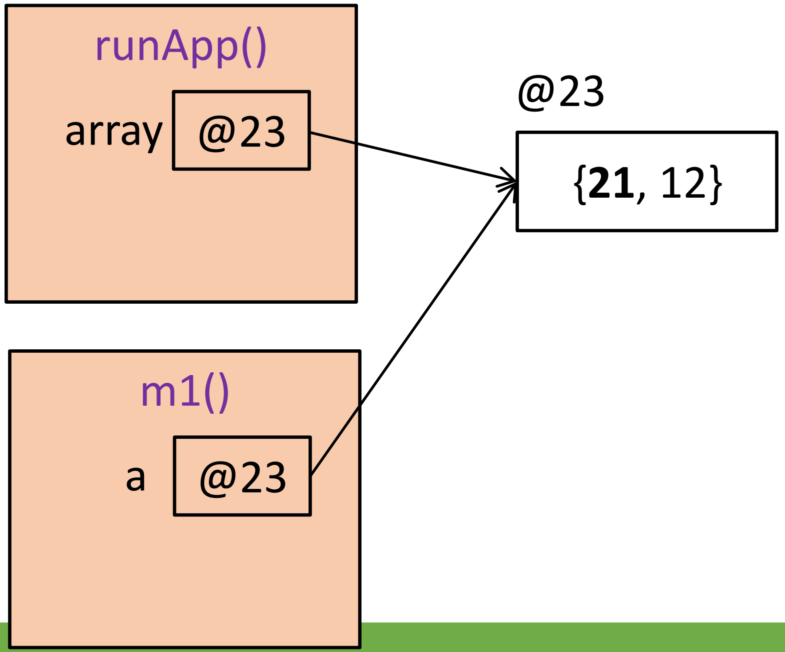
Address of **array** is copied into **a**. **array** and **a** are sharing the same copy of array content.

Calling Method II



```
void runApp() {
    int[] array = {11, 12};
    m1(array);
    System.out.println(array[0] + " " + array[1]);
}


void m1(int[] a){
    a[0] = 21;
}
```

Memory Model



runApp()

array @23

@23

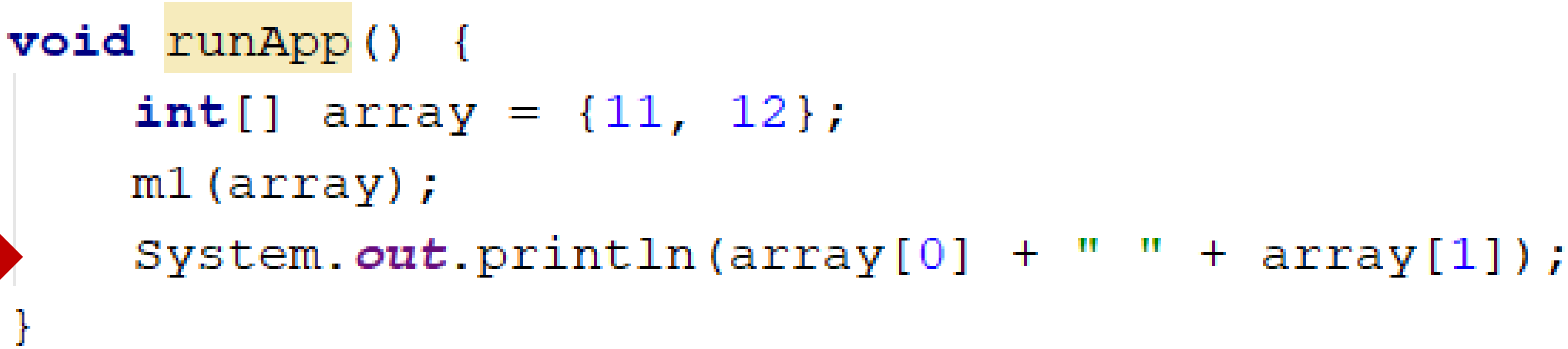
{**21**, 12}

m1()

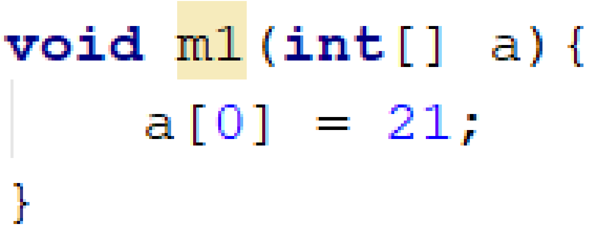
a @23

@23, a[0] changes to 21

Calling Method II

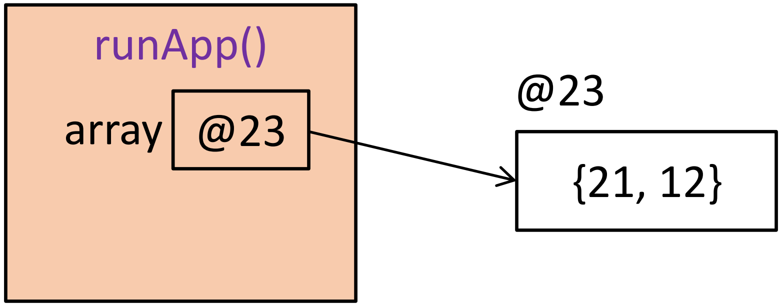


```
void runApp() {
    int[] array = {11, 12};
    m1(array);
    System.out.println(array[0] + " " + array[1]);
}
```



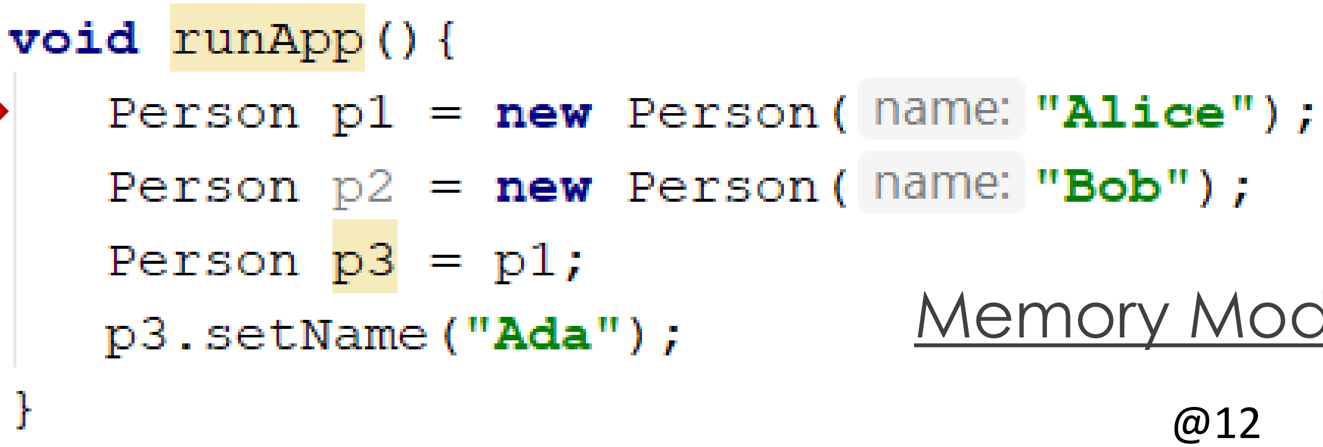
```
void m1(int[] a){
    a[0] = 21;
}
```

Memory Model



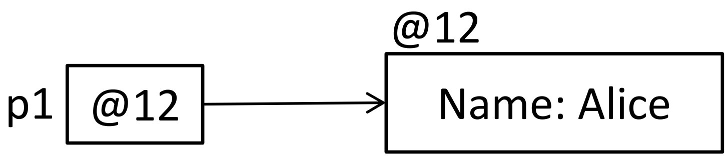
Variable **a** in m1() is no longer reachable, but the array content @23 is still in use by **array**.

Objects

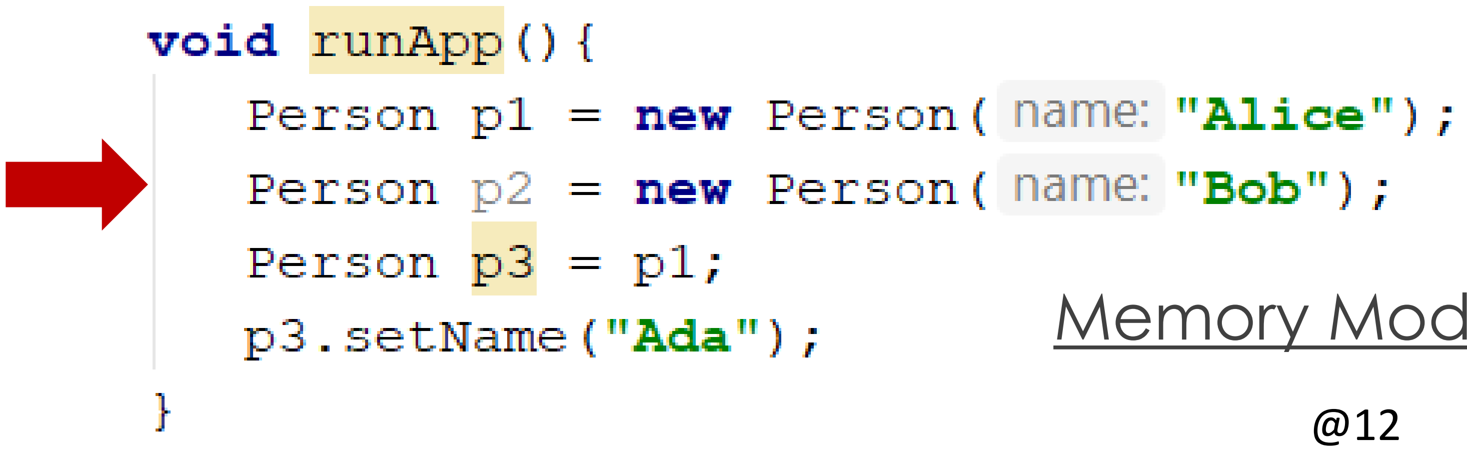


```
void runApp(){
    Person p1 = new Person( name: "Alice");
    Person p2 = new Person( name: "Bob");
    Person p3 = p1;
    p3.setName("Ada");
}
```

Memory Model

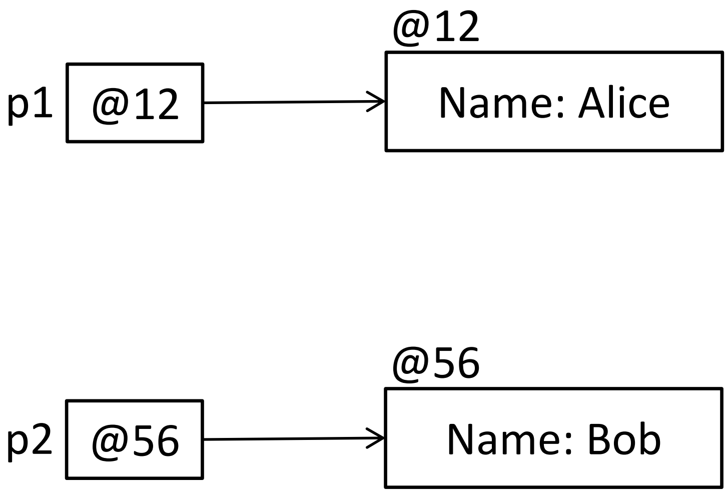


Objects

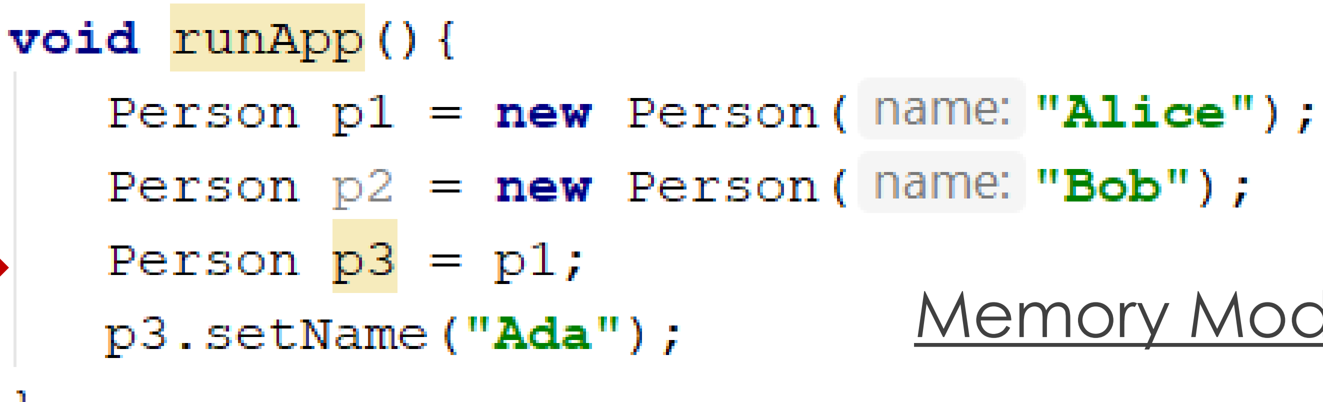


```
void runApp(){
    Person p1 = new Person( name: "Alice");
    Person p2 = new Person( name: "Bob");
    Person p3 = p1;
    p3.setName("Ada");
}
```

Memory Model

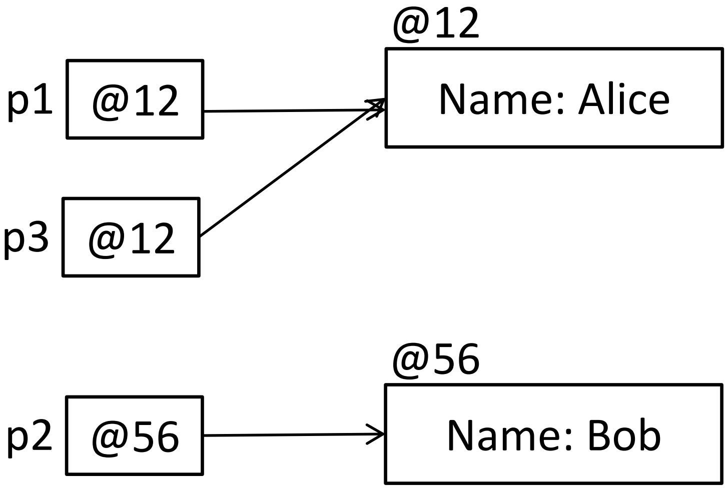


Objects

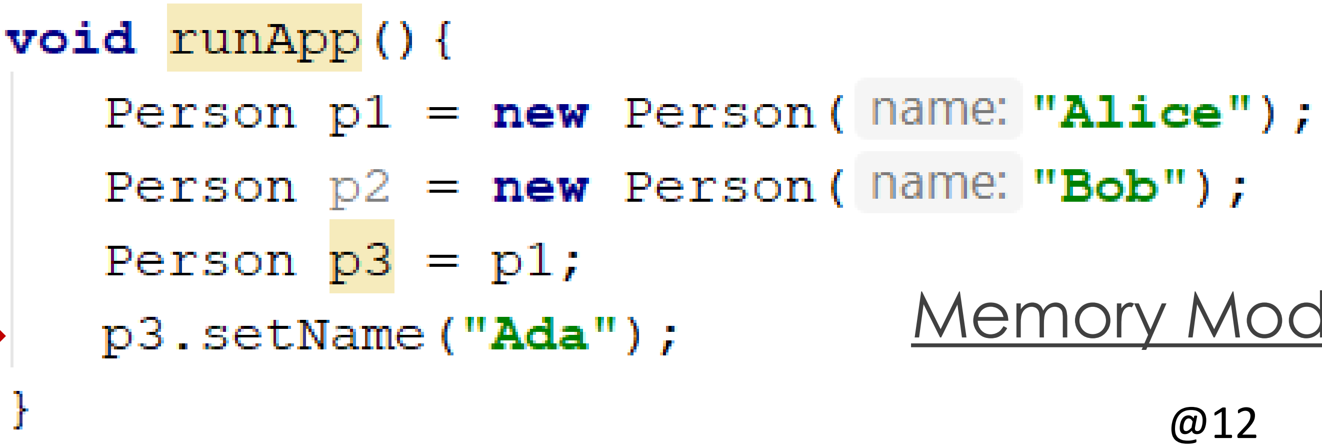


```
void runApp(){
    Person p1 = new Person( name: "Alice");
    Person p2 = new Person( name: "Bob");
    Person p3 = p1;
    p3.setName("Ada");
}
```

Memory Model

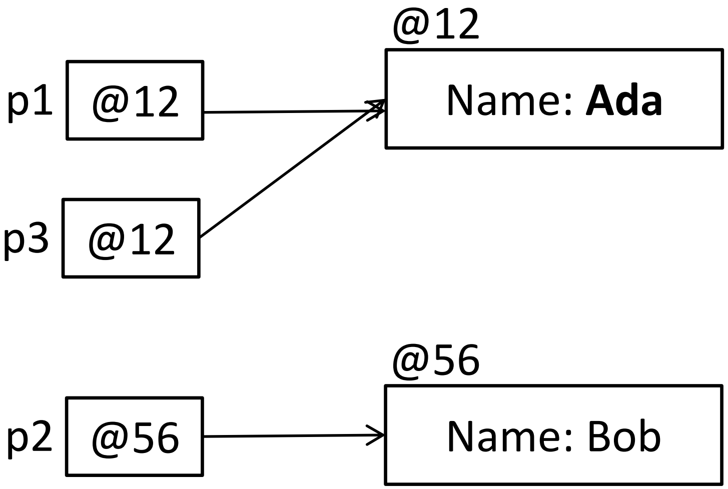


Objects



```
void runApp(){
    Person p1 = new Person( name: "Alice");
    Person p2 = new Person( name: "Bob");
    Person p3 = p1;
    p3.setName("Ada");
}
```

Memory Model



Part A
Discovery Exercises

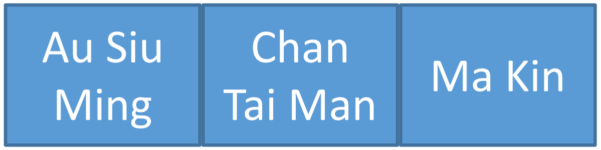
Type your answer in **XXXXXXXX_lab09.docx**

Part B
Programming Exercises

Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array

phoneBook:



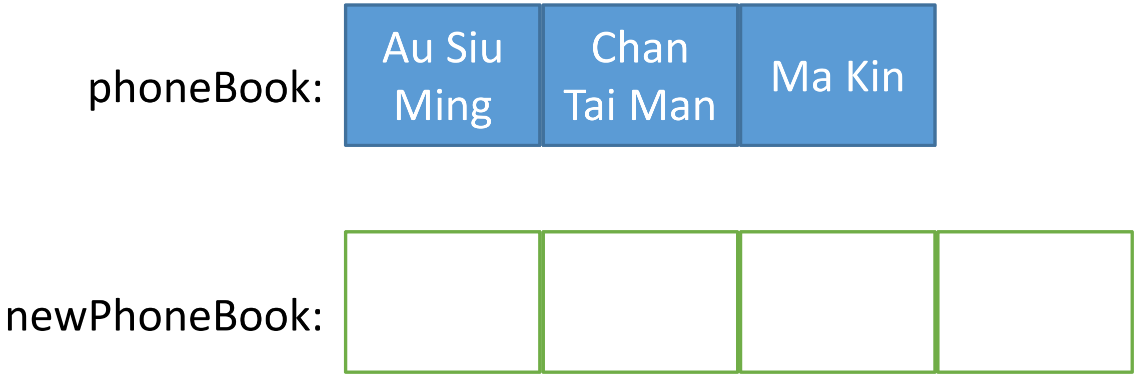
Au Siu Ming

Chan Tai Man

Ma Kin

Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Create a new array with larger size



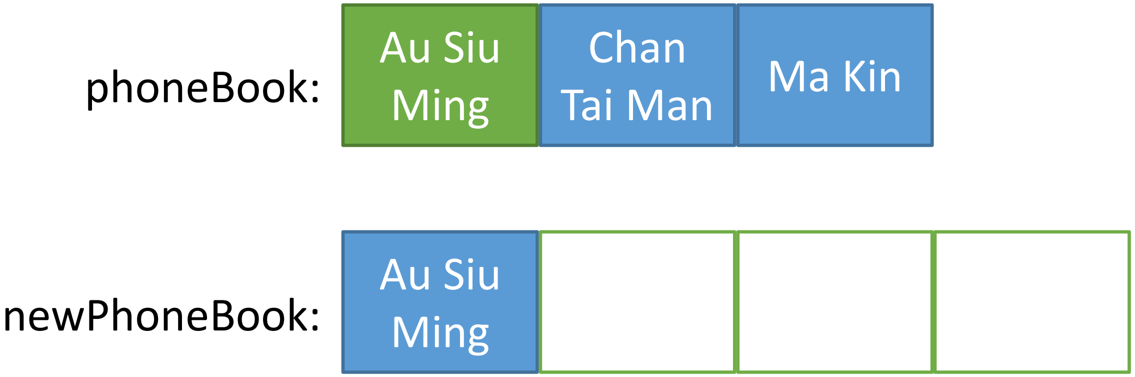
phoneBook:

Au Siu Ming	Chan Tai Man	Ma Kin

newPhoneBook:

Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Compare “Koo Ka Ka” with “Au Siu Ming”, “Koo Ka Ka” should go after it
    - Copy “Au Siu Ming” into the new array

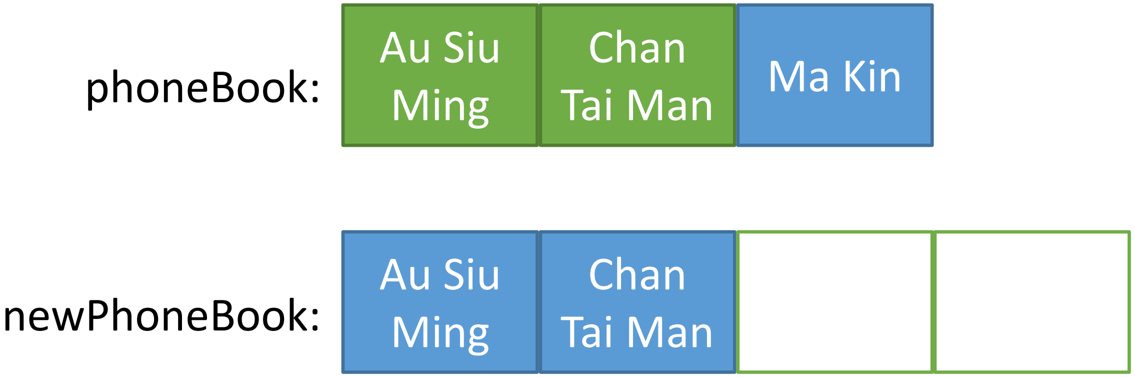


phoneBook: Au Siu Ming | Chan Tai Man | Ma Kin

newPhoneBook: Au Siu Ming | | |

Hints for Task 2

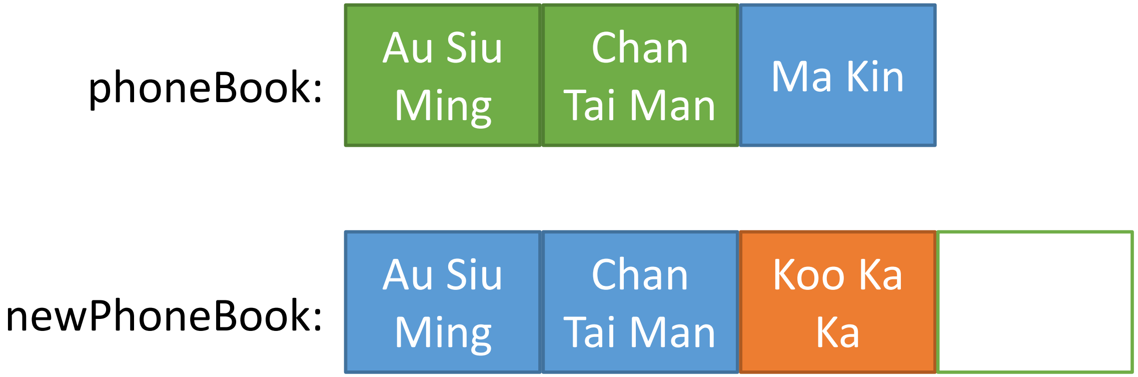
- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Compare “Koo Ka Ka” with “Chan Tai Man”, “Koo Ka Ka” should go after it
    - Copy “Chan Tai Man” into the new array



phoneBook:	Au Siu Ming	Chan Tai Man	Ma Kin	
newPhoneBook:	Au Siu Ming	Chan Tai Man		

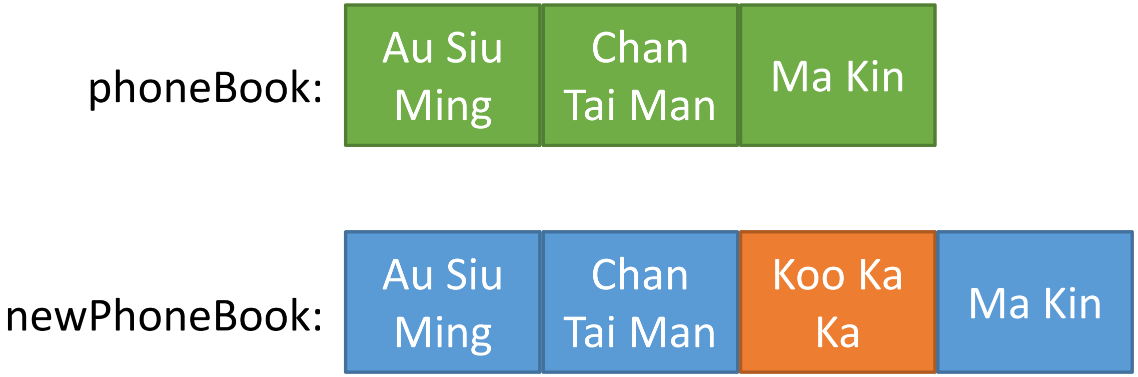
Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Compare “Koo Ka Ka” with “Ma Kin”, “Koo Ka Ka” should go before it
    - Put “Koo Ka Ka” into the new array



Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Copy the remaining parts from the phoneBook to the newPhoneBook



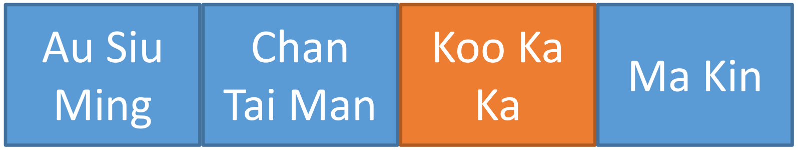
phoneBook:	Au Siu Ming	Chan Tai Man	Ma Kin	
newPhoneBook:	Au Siu Ming	Chan Tai Man	Koo Ka Ka	Ma Kin

Hints for Task 2

- How to maintain lexicographical order by name?
  - Suppose we add “Koo Ka Ka” into the array
    - Then assign newPhoneBook to phoneBook

```
phoneBook = newPhoneBook
```

phoneBook:



Au Siu Ming

Chan Tai Man

Koo Ka Ka

Ma Kin

Lab Exercise Submission

- Submit the following to Moodle
  - ***XXXXXXXX_ lab09.docx***
  - ***XXXXXXXX_lab09.zip***

*Replace "***XXXXXXXX***" with your **student ID**

Deadline: Before next Monday noon

References

- Dean, J., & Dean, R. (2008). Introduction to programming with Java: A problem solving approach. Boston: McGraw-Hill.
- Forouzan, B. A., & Gilberg, R. F. (2007). Computer science: A structured programming approach using C (3rd ed.). Boston, MA: Thomson Course Technology.
- Gaddis, T. (2016). Starting out with Java (6th ed.). Pearson.
- Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.
- Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.
- Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education
- Xavier, C. (2011). Java programming: A practical approach. New Delhi: Tata McGraw Hill.
- Zakhour, S., Kannan, S., & Gallardo, R. (2013). The Java tutorial: A short course on the basics (5th ed.).
- yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming