5.3 Objects and References

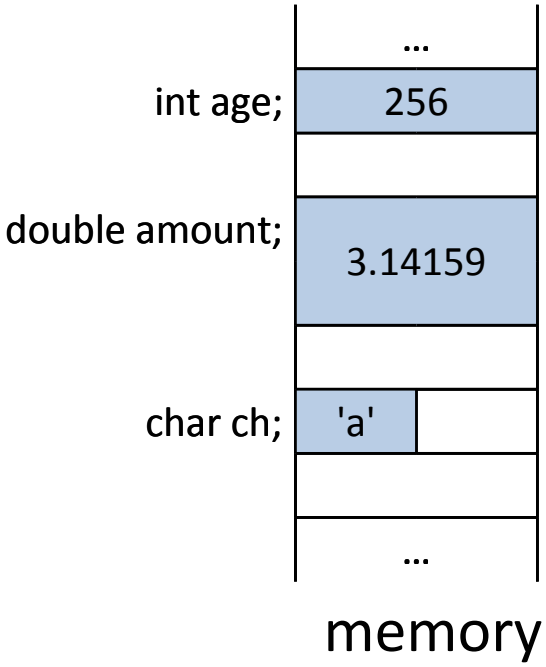
Variables of a Class Type

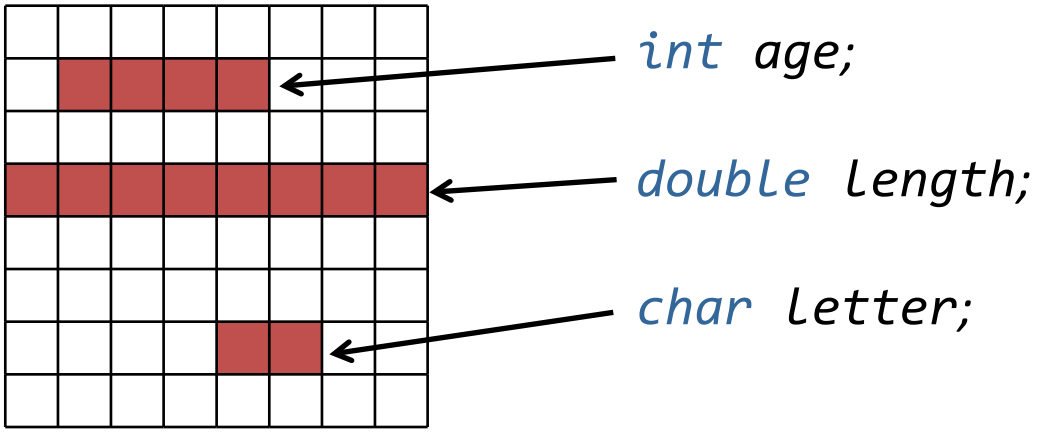
- All variables are implemented as a memory location

- For a variable, it has the memory location assigned
  - If the variable is a **primitive type**,
    - A data value is stored in the memory location
  - If the variable is a **class type**,
    - The value stored in the memory location contains memory address of object named by the variable

Variables of primitive type

- When declaring a primitive variable, a certain amount of memory is allocated based on the declared type
- Actual data values are saved in the allocated memory



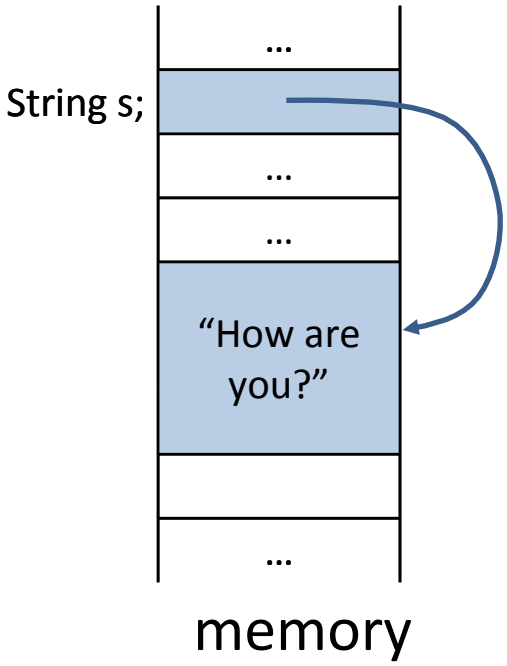


Variables of a Class Type

- Behave differently from variables of a primitive type
  - Scanner keyboard = new Scanner(System.in);
  - Student jack = new Student();
  - String s = “SWDM is nice!”;
- At least, you have seen that you can not easily compare two strings
  - *string1 == string2; //BAD*
  - *string1.equals(string2); //GOOD*

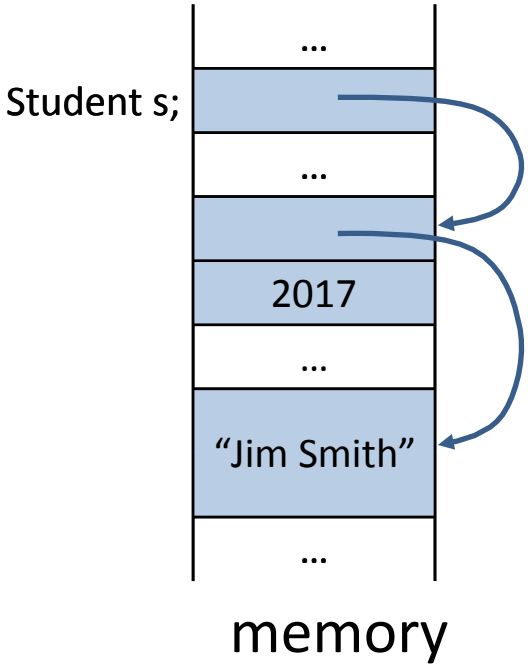
Variables of a Class Type

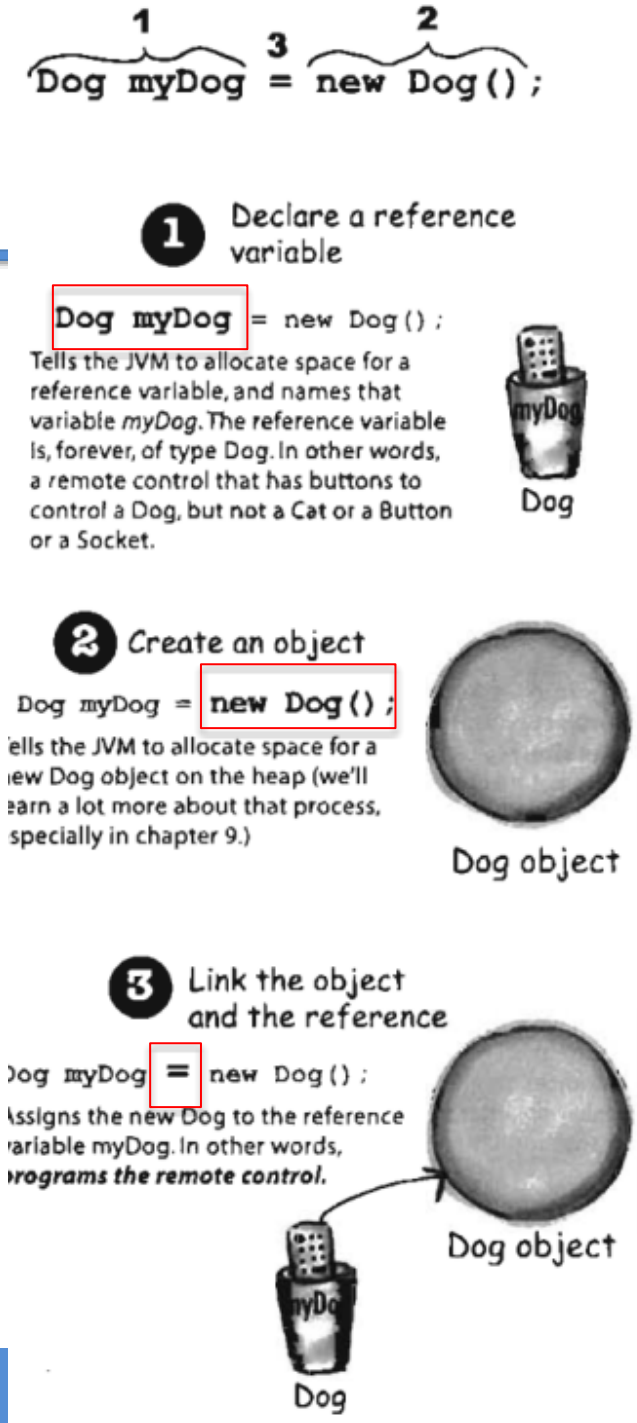
- In a class type variable, the **address** pointing to the actual object is saved (not the object itself)
  - String s;
  - s="How are you?";



Variables of a Class Type

- Variables of class type *cont'd*
  - Student s = new Student();
  - s.setStudent("Jim Smith", 2017);





Example: Books

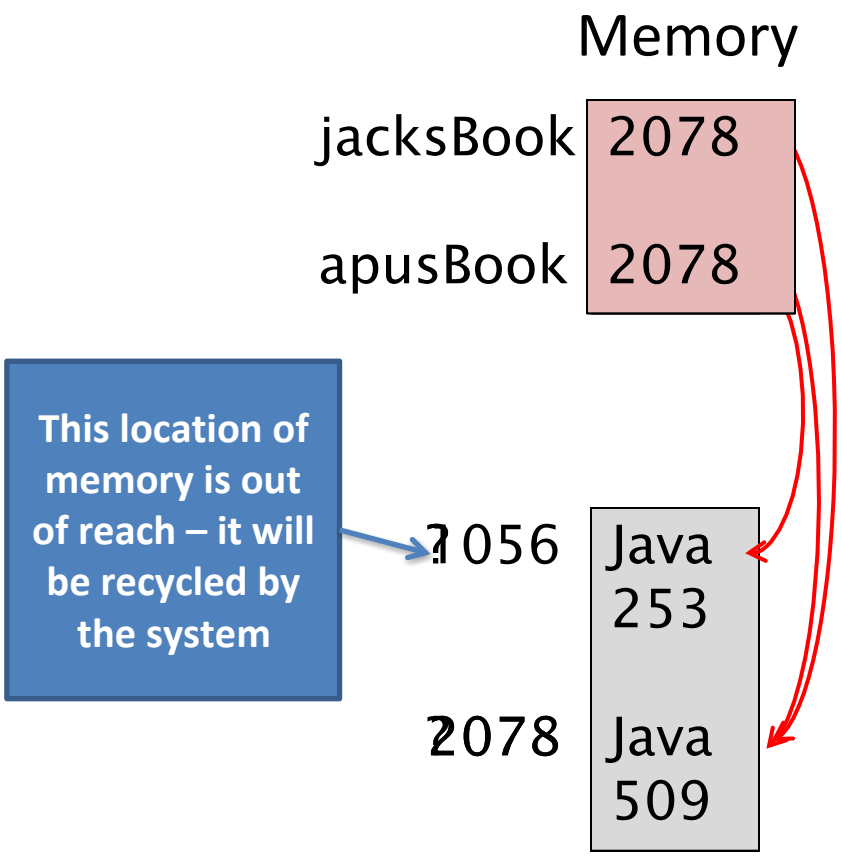
- Assume that we have a class named Book

```
Book jacksBook = new Book(“Java”);
Book apusBook = new Book(“Java”);
```

vs.

```
Book jacksBook = new Book(“Java”);
Book apusBook = jacksBook;
```

Objects in Memory




```
Book jacksBook;
Book apusBook;

jacksBook = new Book("Java");
apusBook = new Book("Java");

jacksBook.setPage(137);
apusBook.setPage(253);

apusBook = jacksBook;
apusBook.setPage(509);
```

jacksBook now has 509 pages!

Remember

- Variables of a class type contain memory addresses
  - NOT objects themselves
- It is dangerous to use assign operator “=” and equal-to operator “==” on class type variables

== vs. equals() for Strings Explained

- String is a class type
- What happens when you have

```
String s1 = new String("Hello");
String s2 = new String("Hello");
boolean strEqual = (s1 == s2);
```

- strEqual is [redacted]! Why?
- [redacted]

== vs. equals() for Strings Explained

- String is a class type
- What happens when you have

```
String s1 = new String("Hello");
String s2 = s1;
boolean strEqual = (s1 == s2);
```

- strEqual is [redacted]!
- [redacted]

== vs. equals() for Strings Explained

- String is a class type
- What happens when you have

```
String s1 = new String("Hello");
String s2 = new String("Hello");
boolean strEqual = (s1.equals(s2));
```

- strEqual is [redacted] Why?
- [redacted]

Defining equals() method

- We cannot use == to compare two objects
- We must write a method for a given class which will make the comparison as needed

```
public class Book {
    private String name;
    private int page;
    public boolean equals(Book book){
        return (this.name.equals(book.name) &&
                this.page == book.page);
    }
}
```

Defining equals() method

- Every class has a default .equals() method if it is not explicitly written
  - It use "==" to check every pair of instance variables
- You decide what it means for two objects of a specific class type to be considered equal
  - Perhaps books are equal if the names and page numbers are equal
  - Perhaps only if the names are equal
  - Put this logic inside .equals() method

Boolean-Valued Methods

- Methods can return a value of type **`boolean`**
- Use a **`boolean`** value in the **`return`** statement
- Note method from listing 5.19

```
/**
 Precondition: This object and the argument otherSpecies
 both have values for their population.
 Returns true if the population of this object is greater
 than the population of otherSpecies; otherwise, returns false.
*/
public boolean isPopulationLargerThan(Species otherSpecies)
{
    return population > otherSpecies.population;
}
```

Lab: define equal method

```
import java.util.Scanner;
public class Species {
    private String name;
    private int population;
    private double growthRate;

    /*The definition of the methods readInput, writeOutput, and predictPopulation
      go here.They are the same as in Listing 5.3 and Listing 5.6 . >
      < The definition of the methods setSpecies, getName, getPopulation,
      and getGrowthRate go here.They are the same as in Listing 5.11 . > */

    public boolean equals (Species otherObject)
    {
        return (this.name.equalsIgnoreCase (otherObject.name)) &&
            (this.population == otherObject.population) &&
            (this.growthRate == otherObject.growthRate);
    }
}
```

이전에 짠 것이 없으면 강의자료실에서 Species.java 파일 다운로드

Lab: define equal method

- Make a main class to test equal method

```
public class SpeciesEqualsDemo {
   public static void main (String [] args)   {
      [redacted]
      if (s1 == s2)
         System.out.println ("Match with ==.");
      else
         System.out.println ("Do Not match with ==.");
      if (s1.equals (s2))
         System.out.println ("Match with the method equals.");
      else
         System.out.println ("Do Not match with the method equals.");
      System.out.println ("Now change one Klingon ox.");
      s2.setSpecies ("klingon ox", 10, 15); //Use lowercase
      if [redacted]
         System.out.println ("Match with the method equals.");
      else
         System.out.println ("Do Not match with the method equals.");
   }
}
```

1) 두개 종족의 객체 만들기
2) 각 객체에 동일한 객체 이름, 종족수, 증가율 입력

==로 각 객체 비교한 결과 확인

아까 짠 equal 함수로 비교한 결과 확인

객체입력값을 바꾸고 Equal 한지 재 확인

LISTING 5.3 **A Species Class Definition—First Attempt** *(part 1 of 2)*

We will give a better version of this class later in this chapter.

Later in this chapter you will see that the modifier `public` *for instance variables should be replaced with* `private`.

```
import java.util.Scanner;
public class SpeciesFirstTry
{
    public String name;
    public int population;
    public double growthRate;

    public void readInput()
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("What is the species' name?");
        name = keyboard.nextLine();
        System.out.println("What is the population of the " +
                           "species?");
        population = keyboard.nextInt();

        System.out.println("Enter growth rate " +
                           "(% increase per year):");
        growthRate = keyboard.nextDouble();
    }
    public void writeOutput()
    {
        System.out.println("Name = " + name);
        System.out.println("Population = " + population);
        System.out.println("Growth rate = " + growthRate + "%");
    }
    public int getPopulationIn10()
    {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0))
        {
            populationAmount = populationAmount +
                               (growthRate / 100) *
                               populationAmount;
            count--;
        }
        if (populationAmount > 0)
                result = (int)populationAmount;
        return result;
    }
}
```

LISTING 5.6 함수 추가

```
public void readInput ()    {
    Scanner keyboard = new Scanner (System.in);
    System.out.println ("What is the species' name?");
    name = keyboard.nextLine ();
    System.out.println ("What is the population of the species?");
    population = keyboard.nextInt ();
    System.out.println ("Enter growth rate (% increase per
year):");
    growthRate = keyboard.nextDouble ();
  }
  public void writeOutput () {
    System.out.println ("Name = " + name);
    System.out.println ("Population = " + population);
    System.out.println ("Growth rate = " + growthRate + "%");
  }
  public int predictPopulation (int years)  {
    int result = 0;
    double populationAmount = population;
    int count = years;
    while ((count > 0) && (populationAmount > 0))
    {
      populationAmount = (populationAmount +
          (growthRate / 100) * populationAmount);
      count - - ;
    }
    if (populationAmount > 0)
      result = (int) populationAmount;
    return result;
  }
```

LISTING 5.11 A Class with Accessor and Mutator Methods

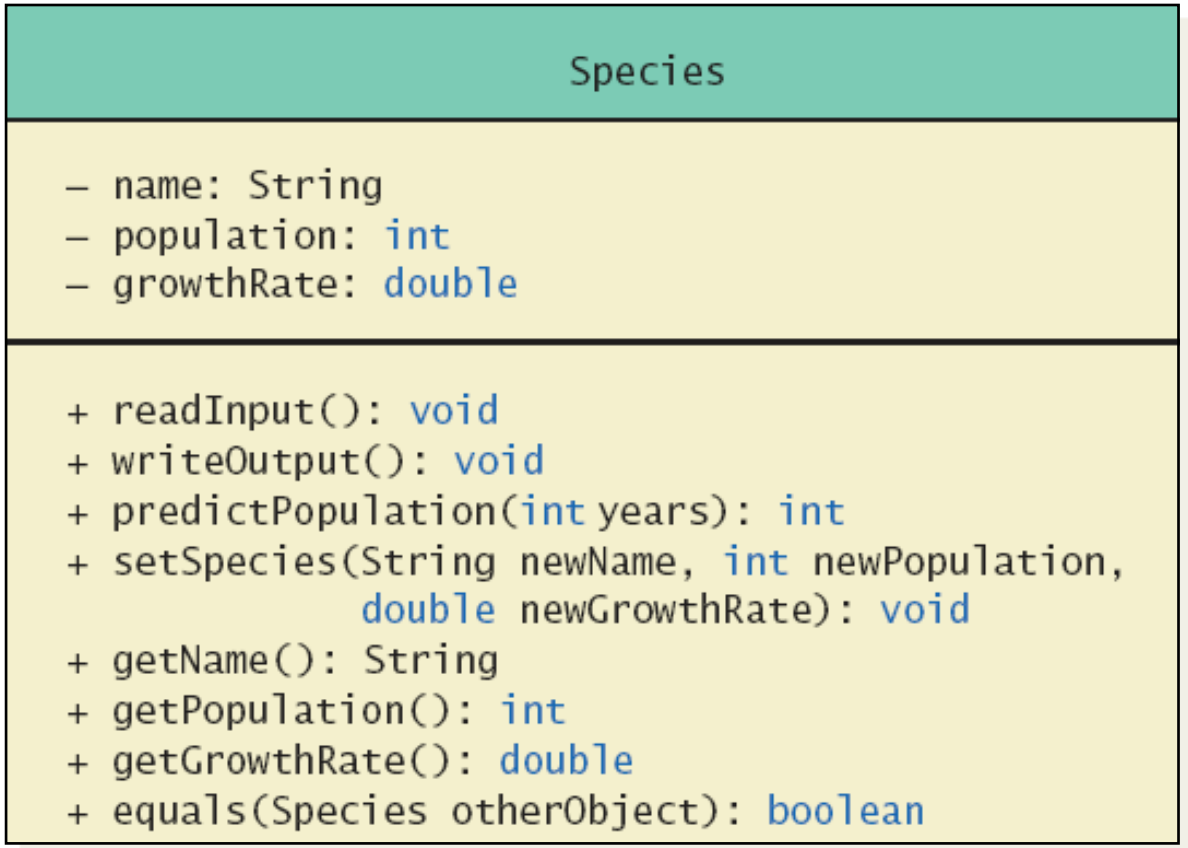
```
import java.util.Scanner;
public class SpeciesFourthTry
{
    private String name;
    private int population;
    private double growthRate;

    <The definitions of the methods readInput, writeOutput, and
     predictPopulation go here. They are the same as in Listing
     5.3 and Listing 5.6.>
    public void setSpecies(String newName, int newPopulation,
                           double newGrowthRate)
    {
        name = newName;
        if (newPopulation >= 0)
            population = newPopulation;
        else
        {
                System.out.println(
                          "ERROR: using a negative population.");
                System.exit(0);
        }
        growthRate = newGrowthRate;
    }
    public String getName()
    {
        return name;
    }
    public int getPopulation()
    {
        return population;
    }
    public double getGrowthRate()
    {
        return growthRate;
    }
}
```

Yes, we will define an even better version of this class later.

A mutator method can check to make sure that instance variables are set to proper values.

Lab: define equal method

- Class Diagram for the class **`Species`** in listing 5.17



Parameters of a Class Type

- When assignment operator used with objects of class type
  - Only **memory address is copied**
- Similar to use of **parameter of class type**
  - **Memory address of actual parameter passed** to formal parameter
  - Formal parameter may access public elements of the class

Parameters of a Primitive Type

```
public void increaseNum(int num)
{
    num++; // num is changed
}
public void doStuff()
{
    int x = 5;
    increaseNum(x);
    System.out.println(x);
}
```

- Prints  Why?
- num is local to increaseNum method; does not change x

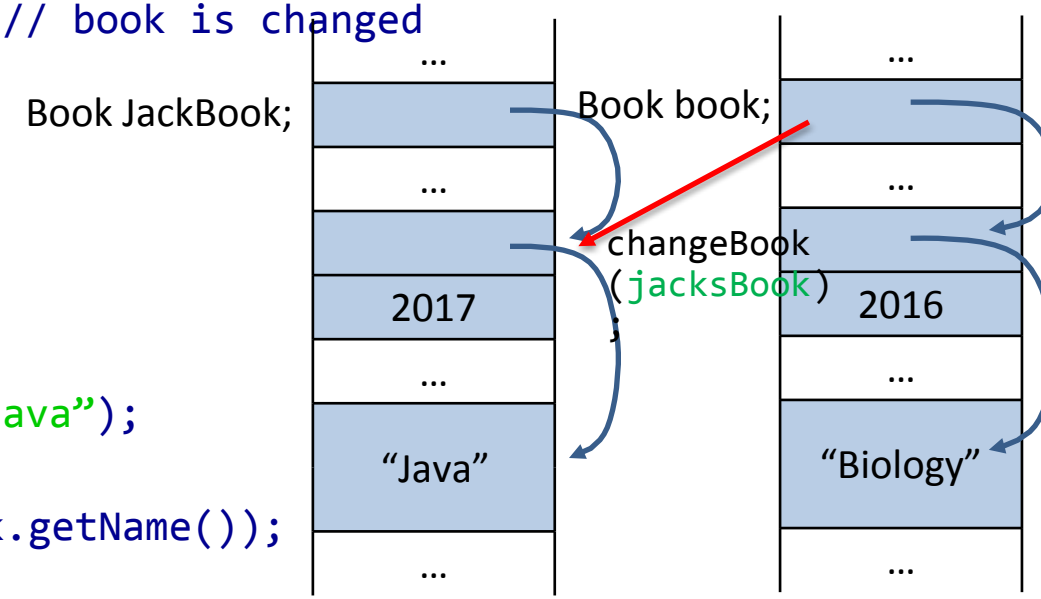
Call by value in C!

```
void swap(int first, int second) {
  int tmp = first;
  first = second;
  second = tmp;
}
void main() {
   int x = 10, y = 20;
   swap(x, y);
   printf("x = %d, y = %d\n", x, y);
}
```

Parameters of a Class Type

```
public Book(String name) { this.name = name; }
public void changeBook(Book book){
    book = new Book("Biology"); // book is changed
    /*
     book = new Book();
     book.setName("Biology");
    */
}
public void doStuff() {
    Book jacksBook = new Book("Java");
    changeBook(jacksBook);
    System.out.println(jacksBook.getName());
}
```

Call by reference in C!

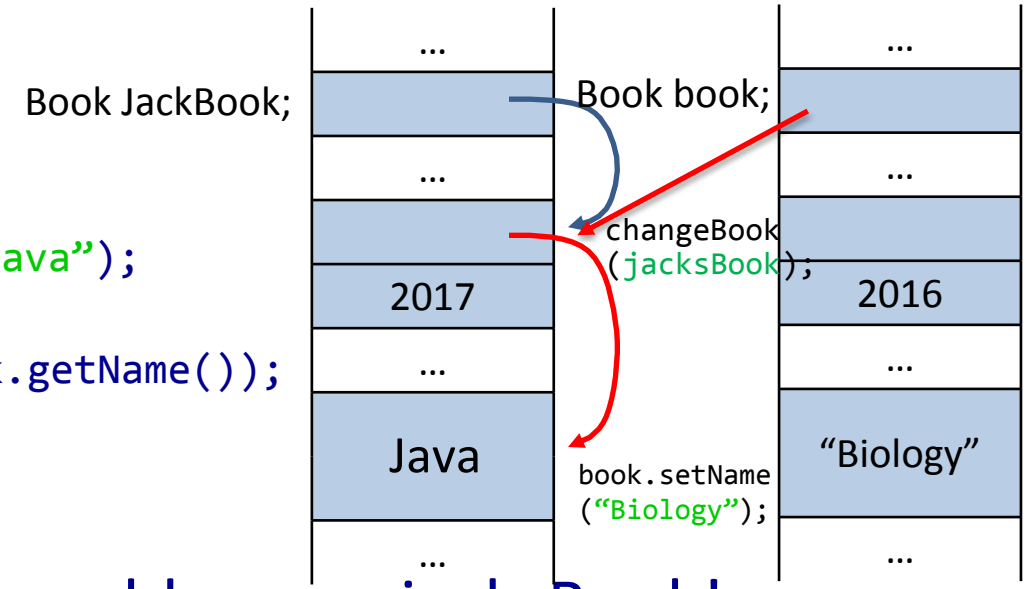


- Prints ? . Why?
- book is local to changeBook, does not change jacksBook

Parameters of a Class Type

Call by reference in C!

```
public void changeBook(Book book)
{
    book.setName("Biology");
}
public void doStuff()
{
     Book jacksBook = new Book("Java");
     changeBook(jacksBook);
     System.out.println(jacksBook.getName());
}
```



- Prints ? . Why?
- book contains the same address as jacksBook!
- Pay attention: **the value of *book*** is not changed!

Practice 5

- EX5_3. Implement a class PersonAddress that represents an entry in an address book.
  - Attributes
    - The first & last name of the person (성, 이름)
    - The e-mail address of the person
    - The telephone number of the person
  - Methods
    - Access each attribute ➔ getter method 3개
    - Change the e-mail address ➔ update email method 1개
    - Change the telephone number ➔ change number method 1개
    - Test whether two instances are equal based solely on name ➔ equal method 1개

Practice 5

- EX5_3. Implement the class

```
public class PersonAddress {

    // instances 선언

    // method 구현
    public void initialize(String first, String
last, String email, String phone) {    }
    public String getFirstName(){    }
    public String getLastName(){    }
    public String getEmailAddress(){   }
    public String getPhoneNumber(){   }
    public void updateEmail(String newEmail){   }
    public void updatePhone(String newPhone){   }
    public boolean equal(PersonAddress other){   }
}
```

```
public class PersonAddressTest {

    // Main 함수 만들기
    public static void main(String[] args) {

      // person address 객체 2개생성
      // 초기화
      // person 1 정보 불러오기
      // person 2 정보 불러오기
      // person 1 과 2 비교
      // person 1의 이메일 업데이트
      // person 2의 전화번호 업데이트

}
}
```

Practice 5

- EX5_3. Write a method and its precondition and postconditions.
  - public String getFirstName()
    - Precondition: none.
    - Postcondition: The first name was returned.
  - public String getLastName()
  - public String getEmailAddress()
  - public String getPhoneNumber()
  - public void updateEmail(String newEmail)
    - Precondition: none.
    - Postcondition: The email address was changed to newEmail.
  - public void updatePhone(String newPhone)
  - public boolean equal(PersonAddress otherPerson)
    - Precondition: otherPerson is not null.
    - Postcondition: True was returned if the first and last names match.

Summary

- Classes have
  - Instance variables to store data
  - Method definitions to perform actions
- Instance variables should be private
- Class needs accessor, mutator methods
- Methods may be
  - Value returning methods
  - Void methods that do not return a value