Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

- Nesting Classes and Arrays

Locations

- Identifying a specific location in the real world sometimes requires multiple pieces of information.
- For example, assume we wanted to identify which city Farmingdale State College is located in.
- What pieces of information make up Farmingdale's location?

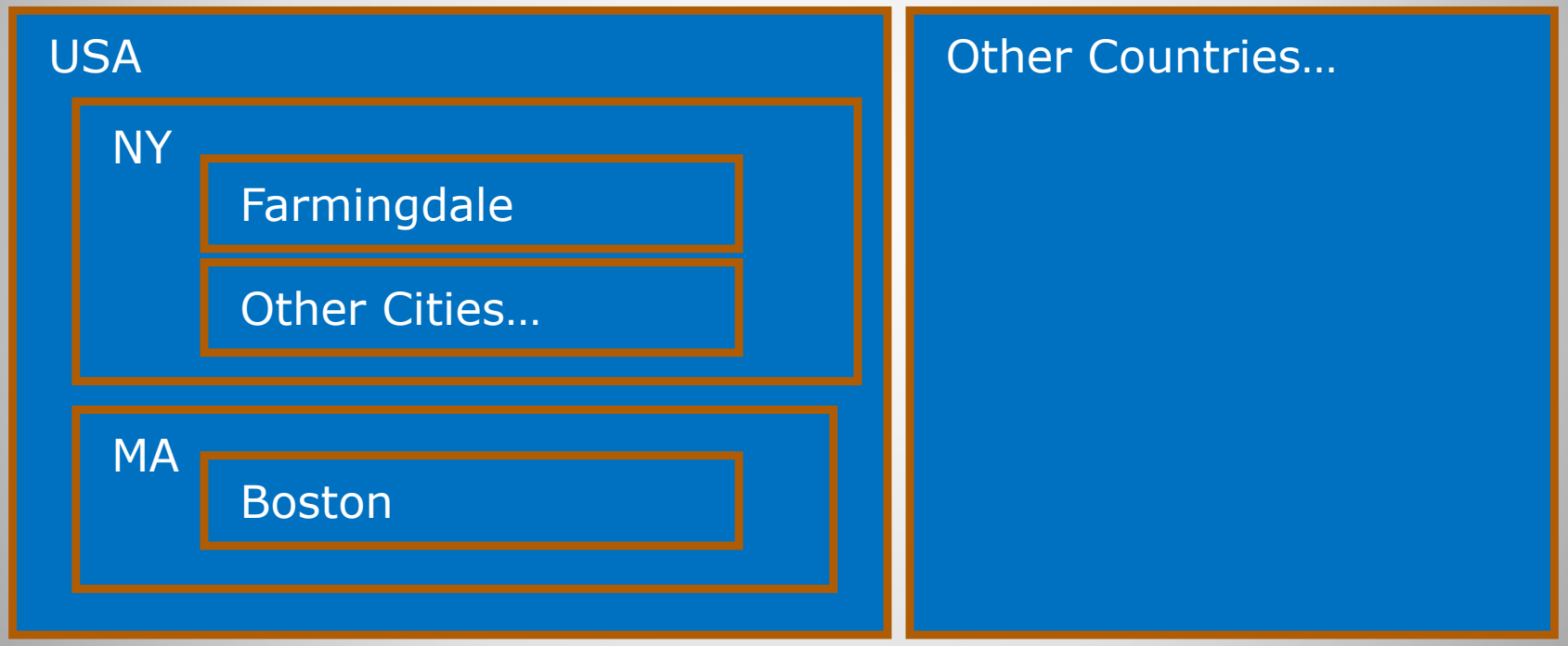
Locations

- We could use the following (from largest to smallest): Country, State, City.

- If we use a . to separate the information we would have the following:

USA.NY.Farmingdale

USA.MA.Boston

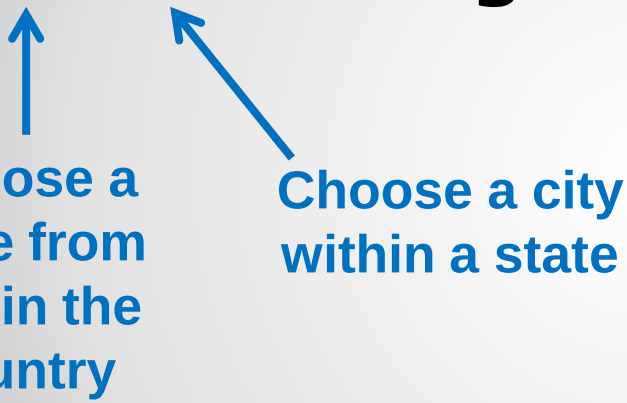


Locations

Locations

- The . is used to choose places within a hierarchy.

USA.NY.Farmingdale



Examples and Public/Private

- In the most of the examples that follow we will not worry about using public/private members and data hiding.
- We will just assume that all classes are in the same package so we can access all member variables.
- The last few examples will have public methods incorporated in them.

Nested Classes

- A class is a user-defined data type.
- You can define members of a class that have other classes as data types.
- For example:

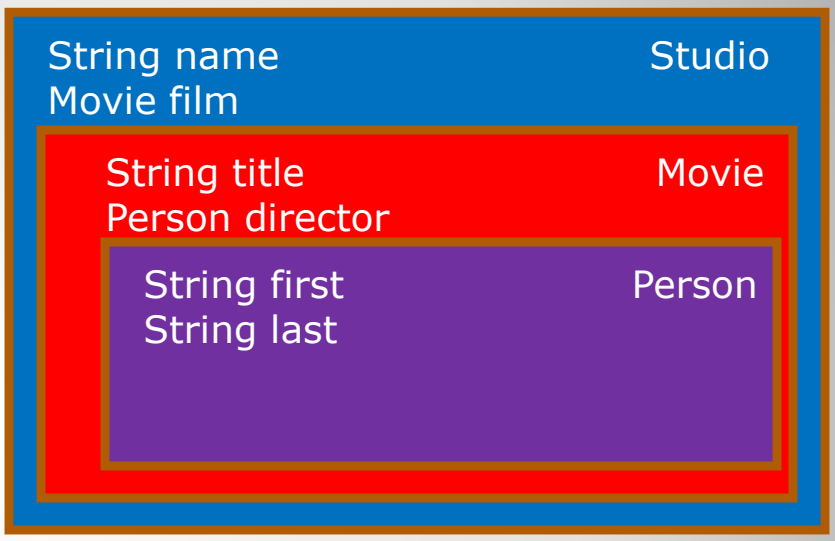
```
class Person {
   String first;
   String last;
}

class Movie {
   String title;
   Person director = new Person();
}
```

director has a data type of Person

Nested Classes

- We can nest as many times as we want. For example:

```
class Person {
   String first;
   String last;
}

class Movie {
   String title;
   Person director = new Person();
}

class Studio {
   String name;
   Movie film = new Movie();
}

Studio s = new Studio(); // Instance of Studio
```



Using Nested Class Members

- Use . to separate the pieces hierarchically.
- For example:

```
class Person {
   String first;
   String last;
}

class Movie {
   String title;
   Person director = new Person();
}

class Studio {
   String name;
   Movie film = new Movie();
}
```

```
Person p = new Person();
p.first = "Rose";
```

Set first on a Person

Use . to pick member

```
Movie m = new Movie();
m.director.first = "Rose";
```

Set first on the director within the Movie

Use . to pick member

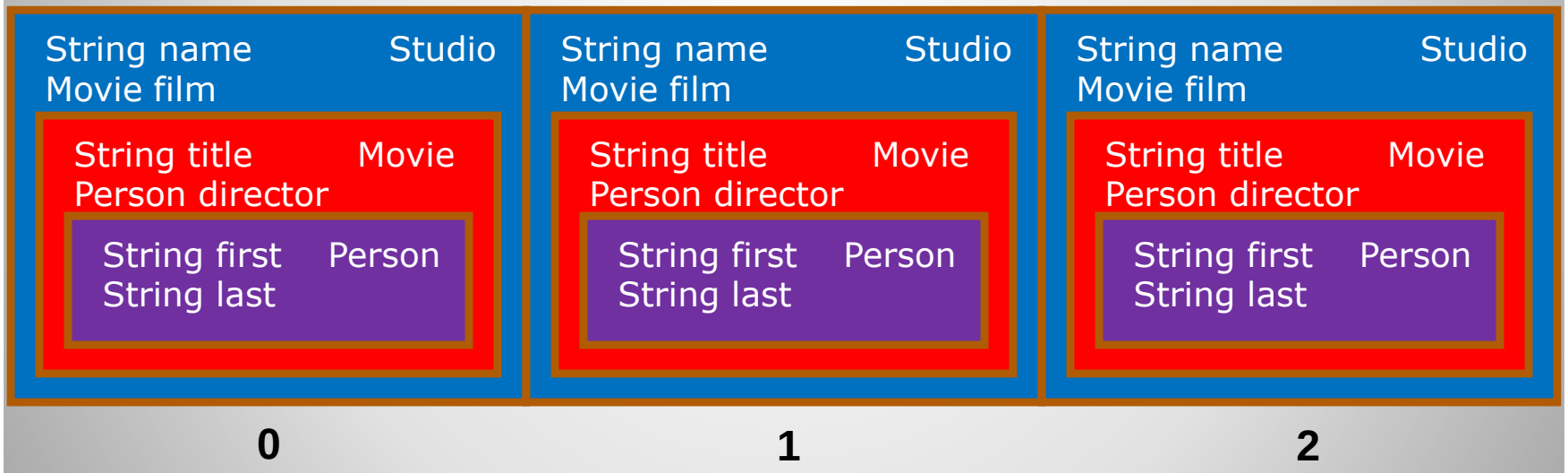
```
Studio s = new Studio();
s.film.director.first = "Rose";
```

Set first on the director within the Movie of the Studio

Use . to pick member

- We can declare an array of Studio.

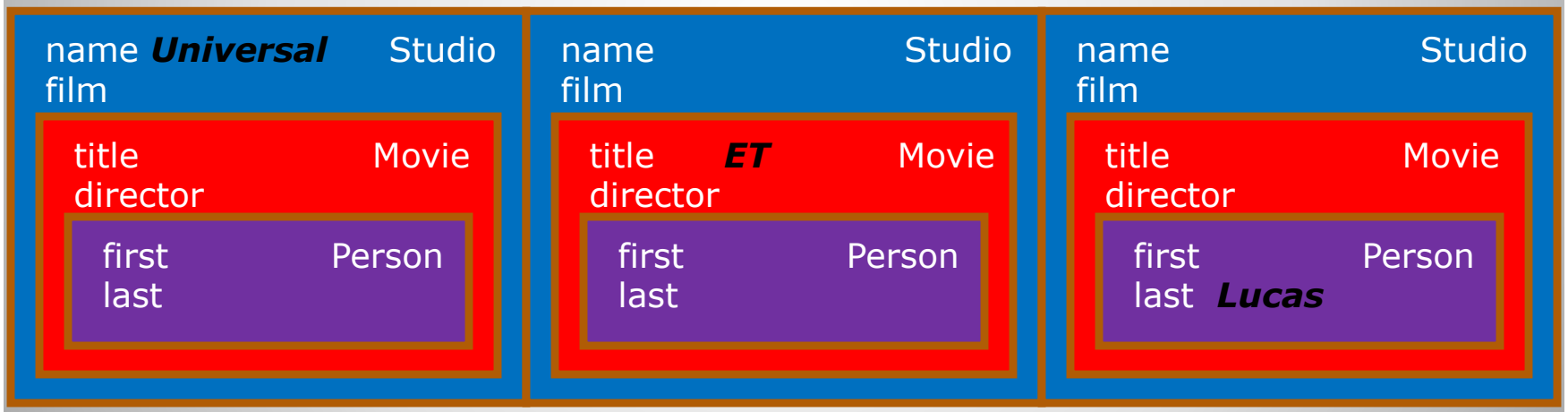
```
Studio ar[3] = new Studio[3];
ar[0] = new Studio();
ar[1] = new Studio();
ar[2] = new Studio();
```



Array of Nested Classes

- Using the array of nested classes

```
Studio ar[3] = new Studio[3];
ar[0] = new Studio();
ar[1] = new Studio();
ar[2] = new Studio();
ar[0].name = "Universal";              // 1st item studio name
ar[1].film.title = "ET";               // 2nd item movie title
ar[2].film.director.last = "Lucas";    // 3rd item movie director last name
```



Array of Nested Classes

- Now on to arrays of classes.
- The following examples use methods and private variables…

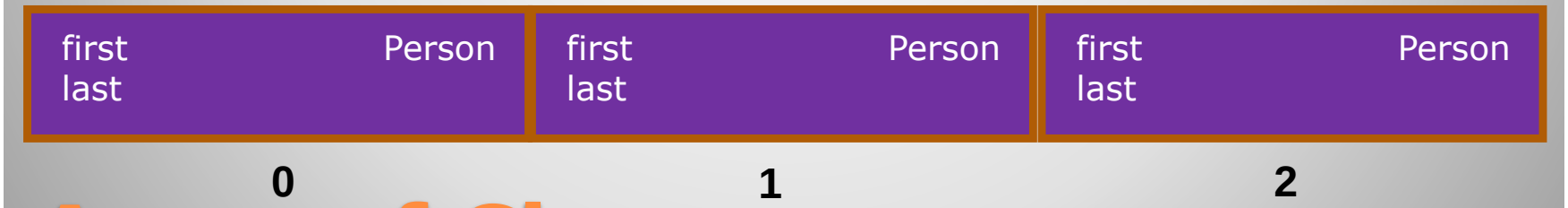
Array of Classes

Array of Classes

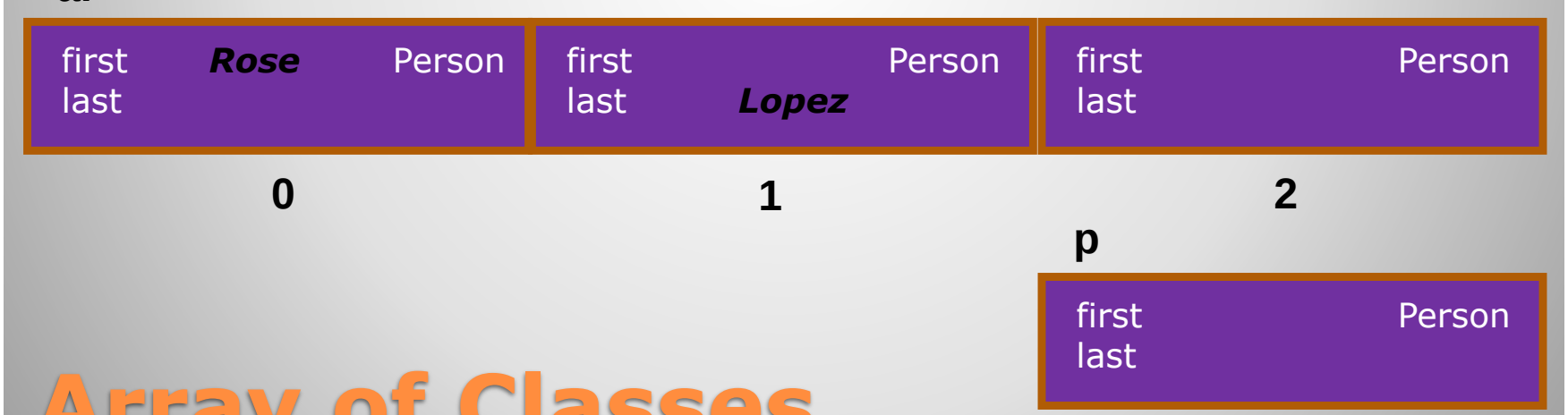
- Using an array of a class

```
class Person {
   private String first;
   private String last;

   // Assume get/set functions are defined here…
}

Person ar[3] = new Person[3]; // Declare an array of Person
ar[0] = new Person();
ar[1] = new Person();
ar[2] = new Person();
```

ar



```
Person ar[3];
// Code to call new for array and all elements goes here…
ar[0].setFirst("Rose"); // Call set function on 1st Person element
ar[1].setLast("Lopez");
System.out.println( ar[1].getLast() ); // Print last of 2nd element
Person p;
p = ar[1]; // Copy second element into another Person variable
ar[2] = p; // Copy Person variable to third element of array
p = ar[1].getLast(); // Incorrect. Different data types (last is a string)
ar[0] = "Rose";  // Incorrect. Different data types (ar[0] is a Person)
```



Array of Classes

- The Movie class has a Person instance as a member variable

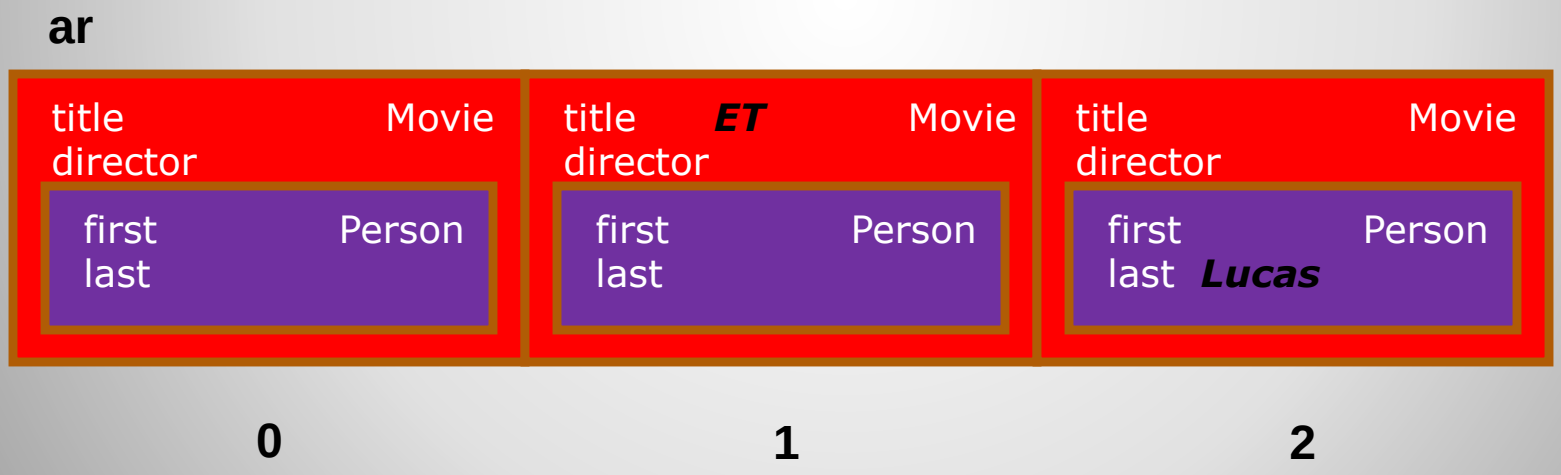
```
class Person {
   private String first;
   private String last;

   // Assume get/set functions are defined here…
};

class Movie {
private:
   private String title;
   private Person director = new Person();

   // Assume get/set functions are defined here…
};
```

Nested Classes

Array of Nested Classes

- Using an array of nested classes

```
Movie ar[3];
// Code to call new for array and all elements goes here…

System.out.println( ar[1].getTitle() );                  // Prints ET
System.out.println( ar[2].getDirector().getLast() ); // Prints Lucas
```



- End of Slides.

End of Slides