# *Programming II (420-B20-HR)*

# *Lab 7 – Arrays of Objects and Lookups*

Date assigned: Wednesday, March 9, 2016

Date due: **Wednesday, March 9, 2016**

**Learning Objectives**

At the end of this lab, the student will be able to:

1. Declare, create and initialize an array of objects.
2. Use a table lookup to locate a value in an array.

**Commands, Methods and Structures Used:**

***Declaration of an Array of objects:***

*className arrayName***[ ]****= new** *arrayName* **[***maxNoOfElements***];**

***Invoke a method for the nth element of an array:***

*arrayName***[***n-1***].***methodName(actual argument list);*

***Process all the elements in an array:***

**for (int i** **=0; i <** *noOfElements***; ++i)**

**{**

*statements to process array[i]*

**}**

**To Be Handed In:**

1. The ***username*\_B20\_L07\_Project** folder containing the Java source files and the Visio class and sequence diagrams should be zipped and uploaded to **Moodle**.

**To Start:**

1. Download and unzip the **B20\_L07\_Project** folder from **Moodle**. Rename **B20\_L07\_Project** to ***username*\_B20\_L07\_Project**.
2. Start **Eclipse**.
3. Open your **Labs** workspace in your **420-B20** folder.
4. Create a new **Java Project** called ***username*\_B20\_L07\_Project**.

# An Array of Objects

***Purpose:*** Learn to create an array of objects and to invoke methods for array elements.

***To Do:***

## Open the **Student**, **Course** and **School** classes. Run the **School** class. Notice what the output is. It currently reads the data from *grades.txt* and prints it out. Open the grades.txt file – it contains two fields in each record – one for name and one for grade. Look in the **Course** class. Notice that there are two arrays – one for the student names and one for the grades. Look in the **Student** class. Notice that it contains two instance variables that correspond to the two fields from the file. Since each **Student** object contains a name and a grade, we can replace the two arrays in the **Course** class with a single array of **Student** objects.

## In the **Course** class, replace the character array of grades and the **String** array of names with an array of **Student** objects called **student**. The array declaration should be:

private Student student [];

## Replace the creation of the two arrays in the constructors to:

student = new Student[50];

## Change the name of the **readGrades()** method to **readStudents()**. Change the calls in the **Course** constructors as well.

## Change the two array assignment statements in the **readStudents()** method to the following:

**student[numStudents].setStudentName(tokens.nextToken());**

**student[numStudents].setStudentGrade(**

tokens.nextToken().charAt(0)**);**

Notice how a method for a single object of the array of objects is invoked.

## Change the **displayDetailLine()** method to accept a single **Student** parameter instead of the **studentName** and **studentGrade** parameters. Change the **printf()** to call the **getStudentName()** and **getStudentGrade ()** methods for the **Student** object instead of using the **studentName** and **studentGrade** parameters.

## Change the call to **displayDetailLine()** in the **displayGradeList()** method to pass the current student instead of the current name and grade.

## Run the program **School**. What happens?

***Explanation:***

The statement:

student = new Student[50];

creates an array of object references, but does not create the objects themselves. You must still instantiate each object in the array.

## Add the following line at the beginning of the while loop in the **readStudents()** method:

**student[numStudents] = new Student();**

## Run the program again.

# Video Store Case Study

***Purpose:*** Learn to declare, create and initialize an array of objects.

***To Do:***

## The **InventoryReportsFrame** class displays a frame with three report options and an option to display the information about one product. At the moment it only works for the button to display all the products. Run **InventoryReportsFrame** in the **videoStoreSystem** package and click the *List All Products* button. How many products were listed? \_1\_\_\_\_\_\_\_\_\_ Open "inventory.txt". How many products are there? \_\_31\_\_\_\_\_\_\_\_ Which one was printed? \_The last one\_ Why? \_The program doesn’t look until there’s nothing left to print, it just prints the last item.\_\_\_\_\_\_\_\_

We want to be able to load all of the products into memory and then loop through them to produce the report. To do this you are going to create an array of **Product** objects.

***In the Inventory class:***

## Change

private Product product;

to

private Product product[];

## Add an integer instance variable called **numProducts** to count the number of products that are read in.

## In the constructor before calling the **load()** method, initialize **numProducts** to 0 and instantiate the **product** array with 100 **Product** objects:

product = new Product[100];

## In the **load()** method:

### change all references to **product** to **product[numProducts]**

### Increment **numProducts** before the end of the **if (inputLine.countTokens() >= 6)** block.

## In the **getProduct()** method:

### Add an integer parameter called **subscript**.

### If **subscript** is between **0** and **numProducts**, return **product[subscript]**. Otherwise return **null**.

## Add an accessor method for **numProducts** called **getNumProducts()**.

***In the InventoryReportsFrame class:***

## In the **btnListAllProducts\_actionPerformed()** method:

### Loop through all the products and call **displayDetailLine()** for each product. Use **getNumProducts()** to determine how many times to loop and use **getProduct()** to get the next product from the inventory.

### Add a line to the display the total number of products.

## Run **InventoryReportsFrame** again. How many products are listed now? \_31\_\_\_

# Table Lookup

***Purpose:*** Learn to use a table lookup to find a given entry in an array.

***To Do:***

The get and set Category methods in the **Game** and **Movie** classes use an if statement to determine the category name for a given category code. We want to change the methods to look the value up in an array instead.

In the **Movie** class in the **videoStoreSystem** package:

## Add a private array of constants to the **Movie** class:

**private static final String categoryName[] = { "Unknown", "Family",**

**"Action", "Comedy", "Western", "Drama", "Horror", "Sci-Fi" };**

## In the **getCategory ()** method delete the switch statement and use the **categoryCode** instance variable as an index into the **categoryName** array to return the name of the category if **categoryCode** is between 0 and the length of the **categoryName** array. Otherwise return "Unknown".

## In the **setCategoryCode(String)** method replace the nested if statement with the following algorithm:

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| the category name, catName | Declare numeric k   1. k = 0 2. categoryCode = 0 3. loop while k < categoryName.length   if catName = categoryName[k]  categoryCode = k  end if  k = k + 1  end loop | None |

## Run **AddProductFrame** and add a movie. Open the "inventory.txt" file to verify that the **setCategory()** method works properly.

## Run **InventoryReportsFrame** and click on **List all products** to verify that your **getCategory()** method works properly.

## Modify the **setCategoryCode(String)** and **getCategory()** methods in the **Game** class to use table lookup instead of if and switch statements to set and get the category. The categories for a game are:

|  |  |
| --- | --- |
| **Game Category Code** | **Meaning** |
| 0 | Unknown (default) |
| 1 | RPG |
| 2 | Action |
| 3 | Education |

## Run **AddProductFrame** again and add a game. Run **InventoryReports** to verity that your change worked.

## Modify the set and get methods for *format code* in both the **Movie** and **Game** classes to use an array lookup instead of hard-coding the String values in the methods. Note that the default for format is 1, not 0.

## Modify the set and get methods for the **Game** platform code to use an array lookup.

## Test your changes.

## Complete the coding of the report display buttons:

## Code the **btnListMovies\_actionPerformed()** method. It should only display movies. The type parameter on the calls to the **displayHeadings()** and **displayDetailLine()** methods should be 'M'.

## Code the **btnListGames\_actionPerformed()** method. It should only display games. The type parameter on the calls to the **displayHeadings()** and **displayDetailLine()** methods should be 'G'.

## Test your changes.

# Homework

## Complete the **Week 7 Quiz** on Moodle by Mar. 13.