MIS 768: Advanced Software Concepts
Spring 2024

ArrayList and Wrapper Classes

Purpose

- Apply ArrayList in aggregating objects
- Learn the usage of dialog boxes in Java programs
- Get familiarize with the wrapper classes and String class for text processing

1. Preparation

(1) Launch Eclipse, and set the workspace to your personal directory.

(2) Create a **package** to hold our source file. Select the folder **src** from the package navigator. Right click on the folder, and then select **New \ Package** from the popup menu.

Name the package as **edu.unlv.mis768.labwork7.**

(3) Download **07_lab_files.zip** from WebCampus. Extract the zip file and then import the .java files into **edu.unlv.mis768.labwork7.**

2. Dialog boxes

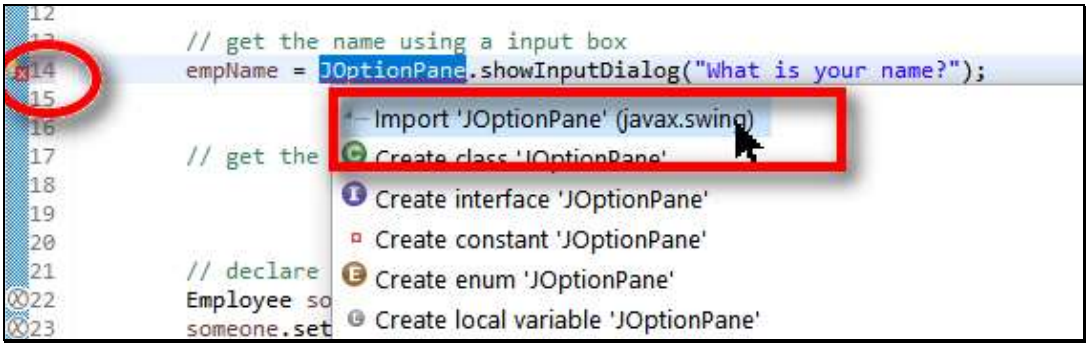
(4) Create a class named **PayrollDialog**.

(5) This program demonstrates using dialogs with **JOptionPane** with **Input Dialog** and **Message Dialog**.

(6) Please enter the following code that uses an **Input Dialog** to get the name String.

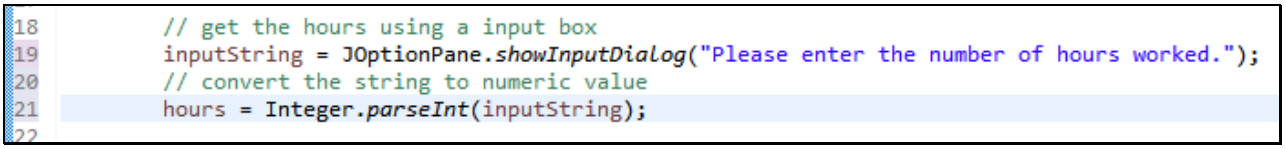
```
4  public class PayrollDialog {
5
6      public static void main(String[] args) {
7          // declare variables
8          String inputString; // for reading input
9          String empName; // the employee's name
10         int hours; // the employee's working hours
11         double grossPay; // the grossPay to be calculated
12
13         // get the name using a input box
14         empName = JOptionPane.showInputDialog("What is your name?");
15
```

(7) When you see errors around **JOptionPane**, you can click the error sign on the side bar of the editor and select "**Import 'JOptionPane' (javax.swing)**" to resolve the problem. Eclipse will add the import statement at the beginning of the code for you.



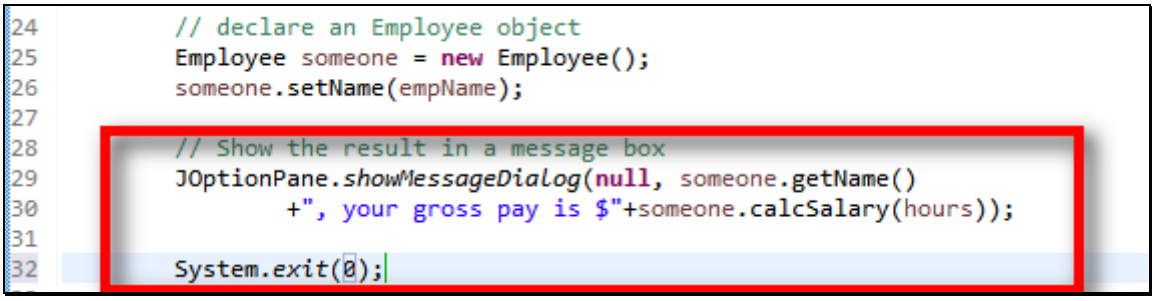
(8) Please also enter the following lines of code.

Note that the **JOptionPane's showInputDialog** method always returns the user's input as a String. We need to convert it to a number before we can use it.



```
// get the hours using a input box
inputString = JOptionPane.showInputDialog("Please enter the number of hours worked.");
// convert the string to numeric value
hours = Integer.parseInt(inputString);
```

(9) Finally, complete the following code to use a Message Dialog to show the result, and **System.exit(0)** to end the program.



```
// declare an Employee object
Employee someone = new Employee();
someone.setName(empName);

// Show the result in a message box
JOptionPane.showMessageDialog(null, someone.getName()
        +", your gross pay is $"+someone.calcSalary(hours));

System.exit(0);
```

(10) You can now run and test the program.

3. Testing and processing character data

(11) Open the partially completed program **CustomerNumber.java**

In this program, the user will enter a customer number, and we'd like to verify whether the input follows the required format.

We can test whether a character is a letter by using `Character.isLetter()` method, and test whether a character is a digit by using `Character.isDigit()` method.

(12) Please complete the program

```
/**
 * The isValid method determines whether a String is a valid customer number.
 * If so, it returns true.
 * @param custNumber The String to test.
 * @return true if valid, otherwise false.
 */
private static boolean isValid(String custNumber) {
    boolean goodSoFar = true;  // Flag
    int i = 0;                 // Control variable indexing the char

    // Test the length.
    if (custNumber.length() != 7)
        // set the flag to false if the length isn't right
        goodSoFar = false;

    // if it passes the length test
    if (goodSoFar){
        // Test the first three characters for letters.
        for (i=0; i<3; i++){
            // get one char
            char d =custNumber.charAt(i);
            // if the char is not a letter
            if (!Character.isLetter(d))
                // set the flag to false
                goodSoFar = false;
        }

        // Test the last four characters for digits.
        for (                        ){
            // get one char

            // if the char is not a digit

                // set the flag to false

        }
    }

    return goodSoFar;
}
```

4. Searching Strings

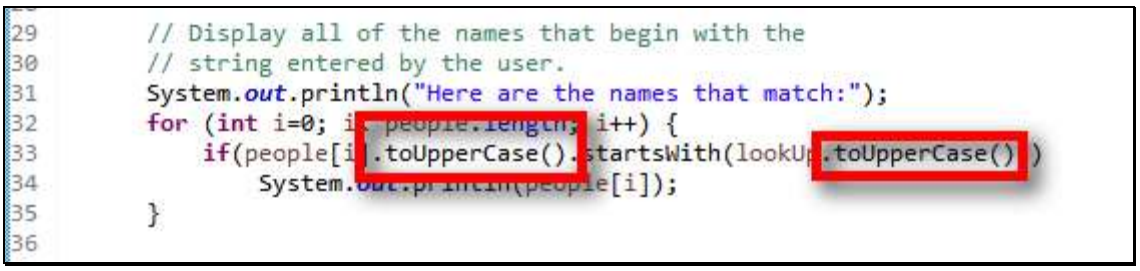
(13) Open the partially completed program **PersonSearch.java**.

In this program, we'll ask the user to enter a few characters, and the program will compare the input to the elements in the array.

(14) Try to use a for loop and the `startsWith()` method to complete this program.

```
29        // Display all of the names that begin with the
30        // string entered by the user.
31        System.out.println("Here are the names that match:");
32        for (int i=0; i< people.length; i++) {
33            if(people[i].startsWith(lookUp))
34                System.out.println(people[i]);
35        }
36
```

(15) This comparison is case-sensitive. If we would like to make it case-insensitive, we can convert the strings to uppercase before making the comparison:

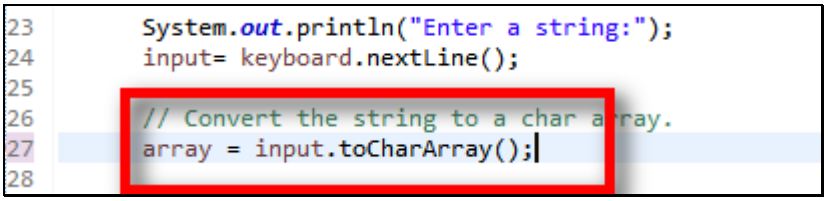


5. Extracting Characters to Arrays

(16) Open the partially completed program **StringAnalyzer.java**.

In this program, the user will enter a string. The program will count the letters, digits and spaces in the string.

(17) Use the `toCharArray` method to convert the string into a char array.



(18) Then in a **`for`** loop, please use *`isLetter, isDigit, isWhitespace`* to count the numbers of letters, digits and spaces in that char array.