

# MIS 768: Advanced Software Concepts

## Spring 2024

### Loop and File

#### Purpose

- Learn the syntax of the loop structure
- Practice file input and output

#### 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.labwork4**.
- (3) Download **04\_lab\_files.zip** from WebCampus. Extract the zip file and then import the .java files into **edu.unlv.mis768.labwork4**.

### PART A: Loop Structure

#### 2. while Loop

- (4) Open **SoccerTeams.java**.
- (5) This program calculates the number of soccer teams that a youth league may create from the number of available players. Input validation is demonstrated with while loops.

The program will ask the user to enter the number of players per team (between 9 and 15), and the available number of players. The program will return how many teams the players can form and how many players are left.

- (6) Use a while loop to validate user input. The **teamSize** is greater than 15 or less than 9, ask for input again until the user enters the correct number.

Thus, the while condition here should be the *invalid* conditions that needs re-entry.

```
20 // Get the number of players per team.
21 System.out.print("Enter the number of players per team.");
22 teamSize = kb.nextInt();
23
24 // Validate the number entered.
25 while (teamSize < MIN_PLAYERS || teamSize > MAX_PLAYERS) {
26     // if the value is not valid, ask the user to enter again
27     System.out.print("The number must be at least " + MIN_PLAYERS +
28                     " and no more than " + MAX_PLAYERS +
29                     ".\n Enter the number of players.");
30     teamSize = kb.nextInt();
31 }
32
```

- (7) Please complete the following code to get the number of players, and validate whether the number is greater than zero. If the number is not greater than zero, ask the user to input again.

```
33 // Get the number of available players.
34 System.out.print("Enter the available number of players.");
35 players = kb.nextInt();
36
37 // Validate the number entered.
38 while ( ) {
39
40 }
41
```

- (8) Finally, calculate the result and show the results to the user.

```
42 // Calculate the number of teams.
43 teams = players / teamSize;
44
45 // Calculate the number of leftover players.
46 leftOver = players % teamSize;
47
48 // Display the results.
49 System.out.print("There will be " + teams +
50                 " teams with " + leftOver +
51                 " players left over.");
52
```

### 3. do-while Loop

- (9) Can we use a do-while Loop instead of while Loop in this program? If so, how? If not, why?

### 4. Sentinel Values

- (10) Open **SoccerPoints.java**.

- (11) This program calculates the total number of points a soccer team has earned over a series of games.

The user enters a series of point values, then -1 when finished.

```
Enter the number of points your team
has earned for each game this season.
Enter -1 when finished.

Enter game points or -1 to end: 3
Enter game points or -1 to end: 9
Enter game points or -1 to end: 10
Enter game points or -1 to end: -1
The total points are 22
```

- (12) The while condition here should represent the reverse of the stopping condition.

Please complete the following code:

```
16 // Display general instructions.
17 System.out.println("Enter the number of points your team");
18 System.out.println("has earned for each game this season.");
19 System.out.println("Enter -1 when finished.");
20 System.out.println();
21
22 // Get the first number of points.
23 System.out.print("Enter game points or -1 to end: ");
24 points = keyboard.nextInt();
25
26 // Accumulate the points until -1 is entered.
27 while (points != -1) {
28     // Add points to totalPoints.
29     totalPoints += points;
30
31     // Get the next number of points.
32     System.out.print("Enter game points or -1 to end: ");
33     points = keyboard.nextInt();
34
35 }
36
37 // Display the total number of points.
38 System.out.println("The total points are " + totalPoints);
39
```

## 5. for Loop

- (13) Please open **Squares.java**. This program asks the user to enter an integer and then prints out the numbers and their squared numbers.

```
Please enter an integer for printing the squares table: 6
Number    Number Squared
-----
1          1
2          4
3          9
4         16
5         25
6         36
```

(14) Please complete the following code:

```
17 // print the table header
18 System.out.println("Number    Number Squared");
19 System.out.println("-----");
20
21 // use a loop to print from 1 to the number entered
22 for(int i=1; i<=number; i++) {
23     System.out.println(i + "\t\t" + i * i);
24 }
```

## PART B: File

### 6. Write data to a file

(15) Please open **WriteFileDemo.java**.

The purpose of this program is to save a list of names to a file. The file name is designated by the user.

The program prompts the user to (1) enter the number of names, (2) give a file name, and then (3) use a **for** loop to get the names from the user, and write the names to a file.

```
How many friends do you have? 3
Enter the filename: D:\temp\friends.txt
Enter the name of friend number 1: alan
Enter the name of friend number 2: bob
Enter the name of friend number 3: calvin
Data written to the file.
```

(16) After the user enters an integer, there is a newline character remained in the Scanner object. Thus, we need to consume it before getting the next String.

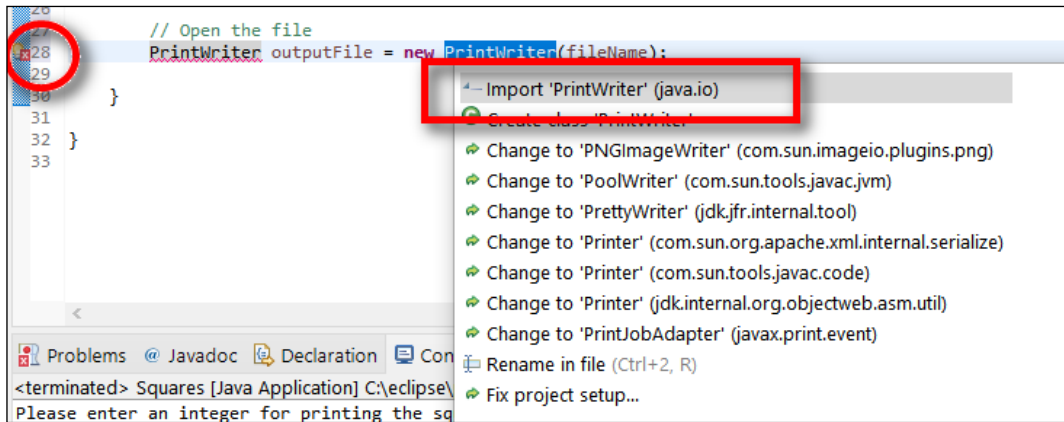
Please enter the following code for **WriteFileDemo** class

```
5 public class WriteFileDemo {
6
7     public static void main(String[] args) {
8         // declare variables
9         String fileName; // name of the file to store the data
10        String friendName; // name of a friend to be written to the file
11        int numOfFriends; // number of friends to be entered by the user
12
13        // Scanner object for keyboard input
14        Scanner kb = new Scanner(System.in);
15
16        // get user input for number of friend
17        System.out.print("How many friends do you have?");
18        numOfFriends = kb.nextInt();
19
20        // Consume the remaining newline character
21        kb.nextLine();
22
23        // get the filename
24        System.out.print("Please enter the file name where you would like to store the data: ");
25        fileName = kb.nextLine();
26    }
```

- (17) Next, use the **filename** to open a new file for saving files.

```
23 // get the filename
24 System.out.print("Please enter the file name where you would like to store the data: ");
25 fileName = kb.nextLine();
26
27 // Open the file
28 PrintWriter outputFile = new PrintWriter(fileName);
29
```

- (18) After entering the above line of code, you will see errors. Click the error symbol on the side bar. Then select import "PrintWriter" to include the import statement.



The screenshot shows an IDE with a Java file. Line 28 has a red squiggly line under `PrintWriter`. A red circle highlights the error icon in the left margin. A context menu is open, showing 'Import "PrintWriter" (java.io)' as the first option, which is highlighted with a red rectangle. Other options include 'Create class "PrintWriter"', 'Change to "PNGImageWriter" (com.sun.imageio.plugins.png)', 'Change to "PoolWriter" (com.sun.tools.javac.jvm)', 'Change to "PrettyWriter" (jdk.jfr.internal.tool)', 'Change to "Printer" (com.sun.org.apache.xml.internal.serialize)', 'Change to "Printer" (com.sun.tools.javac.code)', 'Change to "Printer" (jdk.internal.org.objectweb.asm.util)', 'Change to "PrintJobAdapter" (javax.print.event)', 'Rename in file (Ctrl+2, R)', and 'Fix project setup...'. The bottom status bar shows '<terminated> Squares [Java Application] C:\eclipse\'. The text 'Please enter an integer for printing the sq' is visible at the bottom.

- (19) You will then see the import statement be added to the program.

```
1 package edu.unlv.mis768.labwork4;
2
3 import java.io.PrintWriter;
4 import java.util.Scanner;
5
6 public class WriteFileDemo {
7
8     public static void main(String[] args) {
9         // declare variables
10        String fileName; // name of the file to store the data
11        String friendName; // name of a friend to be written to the file
12        int numOffriends; // number of friends to be entered by the user
13
14        // Scanner object for keyboard input
15        Scanner kb = new Scanner(System.in);
16
17        // get user input for number of friend
18        System.out.print("How many friends do you have?");
19        numOffriends = kb.nextInt();
20
21        // Consume the remaining newline character
22        kb.nextLine();
23
24        // get the filename
25        System.out.print("Please enter the file name where you would like to store the data: ");
26        fileName = kb.nextLine();
27
28        // Open the file
29        PrintWriter outputFile = new PrintWriter(fileName);
30
31    }
32
33 }
```

- (20) However, you will see an error at `PrintWriter`. Please click on the error at the side bar again, and select **Add throws declaration**.

```
26 // get the filename
27 System.out.print("Please enter the file name where you would
28 fileName = kb.nextLine();
29 // Open the file
30 PrintWriter outputFile = new PrintWriter(fileName);
31 }
32 }
33 }
34 }
35 }
36 }
```

- (21) Eclipse will try to add a throws declaration at the header of the method. Since we use the file IO, please select **IO Exception**.

```
8 public class WriteFileDemo {
9 public static void main(String[] args) throws FileNotFoundException {
10 // declare variables
11 String fileName; // name of the file to test
12 String friendName; // name of a friend to test
13 int numOfFriends; // number of friends
14 // Scanner object for keyboard input
15 Scanner kb = new Scanner(System.in);
16 }
```

- (22) Once the file is opened, we can use a for loop to get friend names, and then print the names to the file.

```
30 // Open the file
31 PrintWriter outputFile = new PrintWriter(fileName);
32 // Get input (friend's name) and write it to the file
33 for (int i=0; i<numOfFriends; i++) {
34 // get friend's name
35 System.out.print("Please enter the name of friend " + (i+1));
36 friendName = kb.nextLine();
37 // write the name to file
38 outputFile.println(friendName);
39 }
40 }
41 }
```

- (23) Finally, close the file.

```
33 // Get input (friend's name) and write it to the file
34 for (int i=0; i<numOfFriends; i++) {
35 // get friend's name
36 System.out.print("Please enter the name of friend " + (i+1));
37 friendName = kb.nextLine();
38 // write the name to file
39 outputFile.println(friendName);
40 }
41 // close the file
42 outputFile.close();
43 // Show a confirm message
44 System.out.print("Data written to the file. ");
45 }
46 }
47 }
```

- (24) Now you can run this program to test the results.

**NOTE:** please specify the path of the file as well. If you didn't specify the path, the file will be created under your project folder.

## Exercise:

- (25) Please revise the **WriteFileDemo.java**. When the file already exists, the new names entered will be appended to the file.

## 7. Read data from a file

- (26) Open **ReadFileDemo.java**. Complete the following code:

```
1 package edu.unlv.mis768.labwork4;
2
3 import java.util.Scanner;
4
5 public class ReadFileDemo {
6
7     public static void main(String[] args) {
8         // Create a Scanner object for keyboard input.
9         Scanner keyboard = new Scanner(System.in);
10        String line;
11
12        // Get the file name.
13        System.out.print("Enter the name of a file: ");
14        String fileName = keyboard.nextLine();
15
16        // Open the file.
17        File file = new File(fileName);
18        Scanner inputFile = new Scanner(file);
19
20        // Read the first line from the file.
21        line = inputFile.nextLine();
22    }
```

- (27) We also need to add the import statement for File class. Click on the error symbol at the side bar. Then select **Import 'File'**.

```
8 public static void main(String[] args) {
9     // Create a Scanner object for keyboard input.
10    Scanner keyboard = new Scanner(System.in);
11    String line;
12
13    // Get the file name.
14    System.out.print("Enter the name of a file: ");
15    String fileName = keyboard.nextLine();
16
17    // Open the file.
18    File file = new File(fileName);
19    Scanner inputFile = new Scanner(file);
20
21    // Read the first line from the file.
22    line = inputFile.nextLine();
23
24    // Display the first line of the file.
25    System.out.println(line);
26 }
```

Import 'File' (java.io)

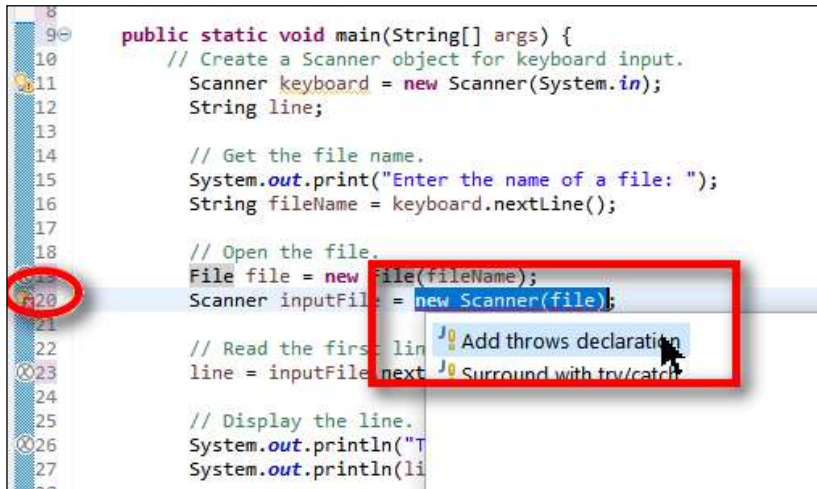
Create class 'File'

Change to 'BitFile' (com.sun.imageio.plugins.common)

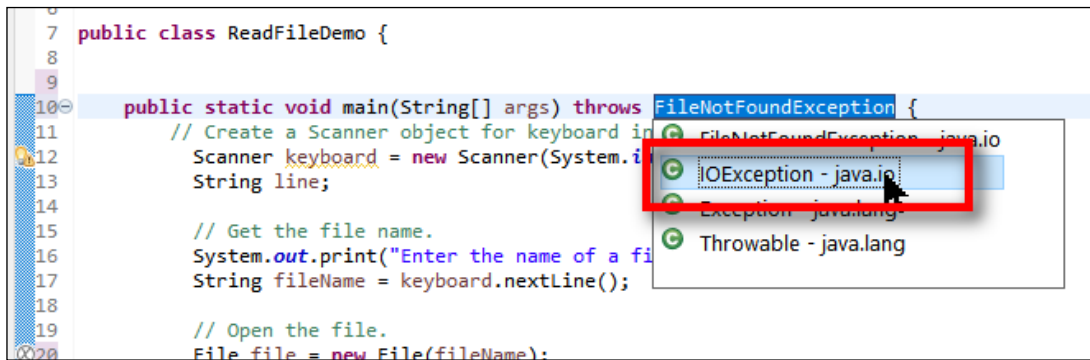
Change to 'ConvFile' (com.sun.tools.sjavac)

(28) Next, the Scanner object is working on file IO, we need to add the throws declaration.

Click on the error symbol at the side bar. Then select **Add throws declaration**.



(29) Please select **throws IOException** to be added to the method header.



(30) You can run and test the program. Please note that this program now only prints the first line of the file content.

## Exercise

(31) Please revise the **ReadFileDemo.java**, so that it can print all the content of the file.