CSE 215: Programming Language II Lab

Lab – 10 **Lab Officer: Tanzina Tazreen**

File I/O

File IO
In Java, File I/O is quite an essential task. We constantly have the requirement to save data to some place and read from it. This can be done if the data can be saved to file. Once it's saved somewhere, it can later be "read" from the file.

Java File Handling
The File class from the **java.io** package, allows us to work with files.
To use the File class, create an object of the class, and specify the filename or directory name:

```
import java.io.File;  // Import the File class

File myObj = new File("filename.txt"); // Specify the filename
```

The File class has many useful methods for creating and getting information about files. For example:

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

You will learn how to create, write, read and delete files.
In Java, File can be read and written to in a number of ways. The following table outlines the names of the classes used to do so:

Read	Write
`Scanner`	`FileWriter`
`BufferedReader`	`BufferedWriter`

API Documentation

Class	Constructor/Methods	Functionality
File	`File(String pathToFile)`	Creates a file object with the file in the provided path
	`boolean exists()`	Returns whether the file exists in the given path
Scanner ★	`Scanner(File fileObject)`	Creates a Scanner object to read from the given file
	`boolean hasNext()`	Returns whether the file has more content to be read
BufferedReader	`BufferedReader(FileReader fileReaderObj)`	Creates a BufferedReader object
	`String readLine()`	Reads a line from the file
FileWriter ★	`FileWriter(File fiileToWriteToObj, boolean append)`	Creates a FileWriter object. If append is true, file is not overwritten, rather it is appended to.
	`void write(String line)`	Writes a line to the file
BufferedWriter	`BufferedWriter(FileWriter fileWriterObj)`	Creates a BufferedWriter object.
	`void write(String line)`	Writes a line to the file.
	`void newLine()`	Writes "\n" to the current position in the file.

Create a File

```
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle
errors

public class CreateFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
      } else {
        System.out.println("File already exists.");
      }
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

We would have to create/open a file inside a ***try*** block otherwise it throws an IOException if an error occurs (if the file cannot be created for some reason)

To create a file in a specific directory, specify the path of the file and use double backslashes to escape the "\" character (for Windows). On Mac and Linux you can just write the path, like: /Users/name/filename.txt

```
File myObj = new File("C:\\Users\\MyName\\filename.txt");
```

Write To a File

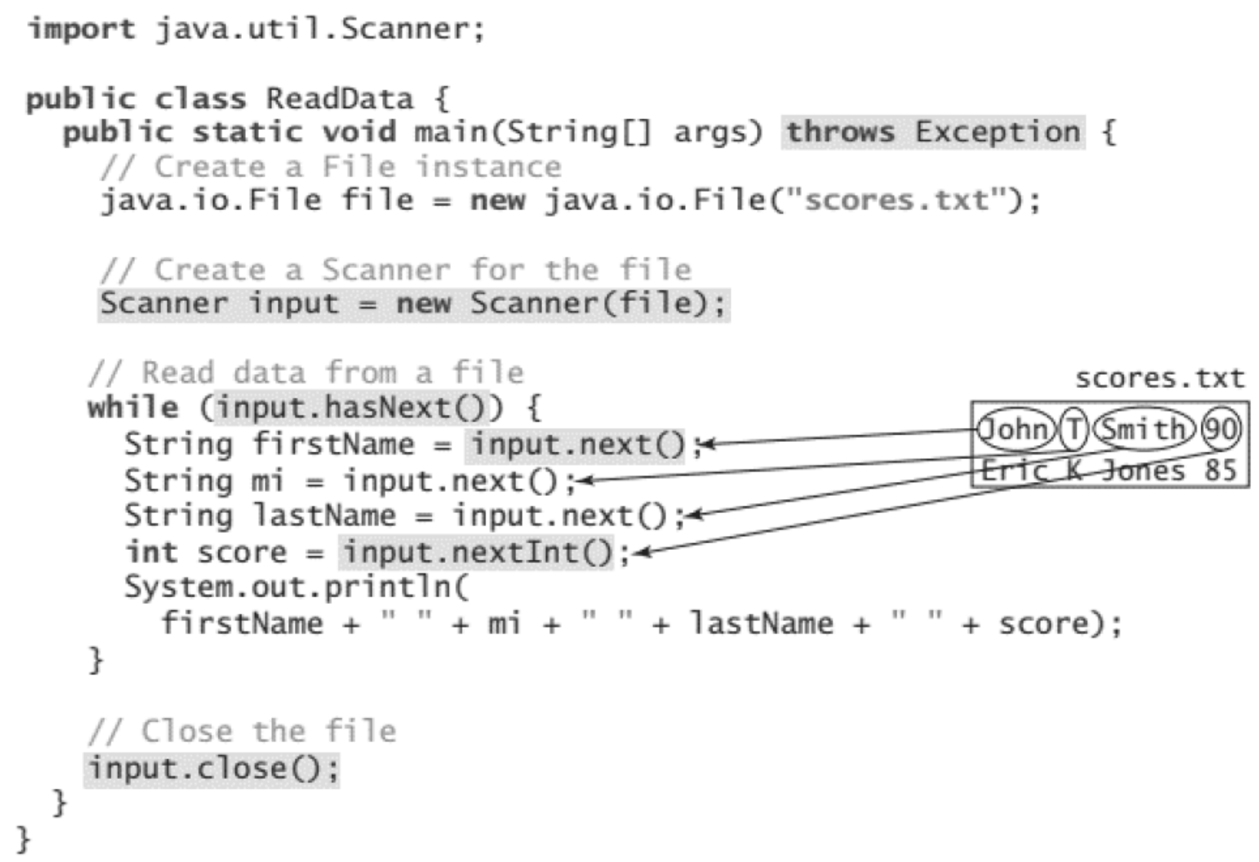
we use the FileWriter class together with its write() method to write some text to the file we created in the example above. Note that when you are done writing to the file, you should close it with the close() method.

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle
errors

public class WriteToFile {
  public static void main(String[] args) {
    try {
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Files in Java might be tricky, but it is fun 
enough!");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

Read a File

we use the Scanner class to read the contents of the text file we just created

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle 
errors
import java.util.Scanner; // Import the Scanner class to read text 
files

public class ReadFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```



```
import java.util.Scanner;

public class ReadData {
  public static void main(String[] args) throws Exception {
    // Create a File instance
    java.io.File file = new java.io.File("scores.txt");

    // Create a Scanner for the file
    Scanner input = new Scanner(file);

    // Read data from a file
    while (input.hasNext()) {
      String firstName = input.next();
      String mi = input.next();
      String lastName = input.next();
      int score = input.nextInt();
      System.out.println(
        firstName + " " + mi + " " + lastName + " " + score);
    }

    // Close the file
    input.close();
  }
}
```

Delete a File

To delete a file in Java, use the delete() method.

```
import java.io.File;  // Import the File class

public class DeleteFile {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.delete()) {
      System.out.println("Deleted the file: " + myObj.getName());
    } else {
      System.out.println("Failed to delete the file.");
    }
  }
}
```

Task:

1. Write a program that takes integers from user and writes them into a file until user inputs a negative number. The program should then read the file and print sum and average of the numbers.

2. Create a Quiz class with id and mark. Now write a program that reads a file containing records of Quiz objects and initialize an array. The program should then print all the objects in the Quiz array and print the id of the student who obtained the highest mark.

Sample File:
113098 20
115089 15
345678 12
234566 18 Program
Output: ID:113098
mark:20 ID:115089
mark:15 ID:345678
mark:12 ID:234566
mark:18
Highest mark obtained by ID:113098