

Chapter 4: Reading an Input File

Reading an Input File

Now for the last topic in today's lesson! We have a file on disk with the team's data in it. If we want to print a roster from it or do anything else with the information, how do we read it and use it? We won't have to change the Player and Team classes this time. But we will need a new TeamDriver program.

Once again, let me show you the class, then I will explain what it does:

[Print this code](#)

```
import java.io.*;

import java.util.Scanner;


public class TeamDriver
{

    public static void main(String[] args)

    {

        Team t = new Team();

        String name;

        char position;

        double avgPoints;

        double avgRebounds;

        double avgAssists;


        try

        {

            Scanner scan = new Scanner(new File("TeamFile.txt"));

            while (scan.hasNext())

            {

                name = scan.next() + " " + scan.next();

                position = scan.next().charAt(0);

                avgPoints = scan.nextDouble();

                avgRebounds = scan.nextDouble();

                avgAssists = scan.nextDouble();

                t.addPlayer(new Player(name, position, avgPoints, avgRebounds, avgAssists));

            }

            scan.close();

        }

        catch(IOException e)

        {

            System.out.println("*** I/O Error ***\n" + e);

        }

        System.out.println(t.toString());

    }

}
```

Since we'll be creating a team of players from values we get out of a file, we'll need local variables to hold those values. So the first thing we need to do is declare our local variables. In this example, I used the name *t* for our Team and the names *name*, *position*, *avgPoints*, *avgRebounds*, and *avgAssists* for the information we need to create a Player. As you will see, we won't actually need a name for our Player objects.

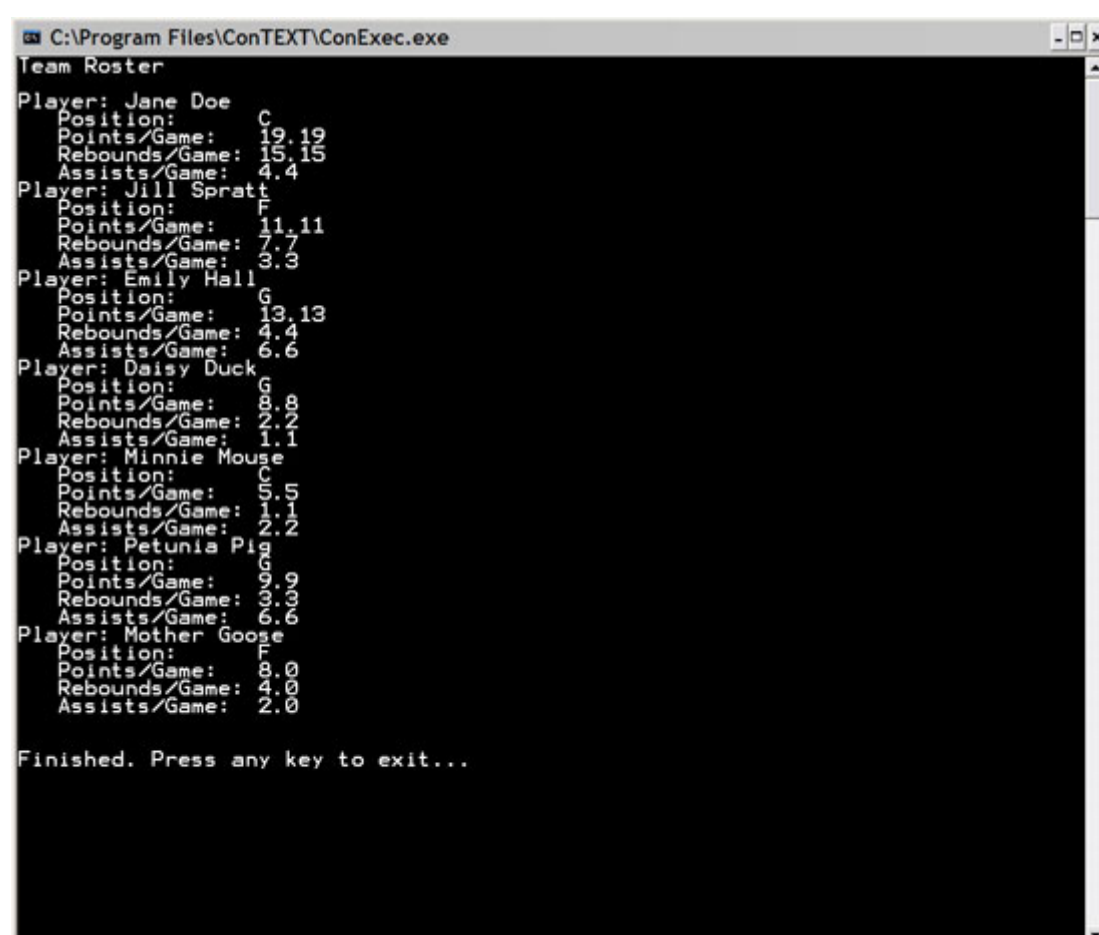
Since we're using file operations, we need to have try and catch blocks again in case there are input errors. Inside the try block, you can see that we're able to use the same Scanner class we've used for keyboard input to process input files as well. Instead of creating our Scanner with an argument of `System.in`, we create it with a File object as its argument. Since we don't need to use the File object outside of the Scanner class, we can create it without a name at the same time we pass it to the Scanner. So the argument for the Scanner is `new File("TeamFile.txt")`, where the argument we pass to the File constructor is the name of the disk file we want to read. In this case, we're assuming that the file is in the same disk directory as our program.

Once we've created and connected the Scanner and File objects, we can read data from the file just as if it were being entered from the keyboard. We have a loop set up that keeps reading as long as there is more data to read. Each time through the loop, we read one line from the input file with information about one player.

As we read each player's information from the file, we create a new Player object to hold the data, and add the object to our team in the Team object *t*.

Once we finish processing the input file, we close the Scanner object to release the file properly.

Our last action in this program is to display the team roster the same way we did before using *t*'s `toString()` method and a call to `println()`. If reading the file worked as it should, you should see the following output on your screen:



```
C:\Program Files\ConTEXT\ConExec.exe
Team Roster
Player: Jane Doe
  Position: C
  Points/Game: 19.19
  Rebounds/Game: 15.15
  Assists/Game: 4.4
Player: Jill Spratt
  Position: F
  Points/Game: 11.11
  Rebounds/Game: 7.7
  Assists/Game: 3.3
Player: Emily Hall
  Position: G
  Points/Game: 13.13
  Rebounds/Game: 4.4
  Assists/Game: 6.6
Player: Daisy Duck
  Position: G
  Points/Game: 8.8
  Rebounds/Game: 2.2
  Assists/Game: 1.1
Player: Minnie Mouse
  Position: C
  Points/Game: 5.5
  Rebounds/Game: 1.1
  Assists/Game: 2.2
Player: Petunia Pig
  Position: G
  Points/Game: 9.9
  Rebounds/Game: 3.3
  Assists/Game: 6.6
Player: Mother Goose
  Position: F
  Points/Game: 8.0
  Rebounds/Game: 4.0
  Assists/Game: 2.0
Finished. Press any key to exit...
```

Note that this is exactly the same roster we saw when we created the file in the driver program. Since we wrote that information to a file, then read it back in to create a roster of players again, we should expect to see the same results.