# Accessing Records Randomly

lec(11)

# Writing Records to a Random Access File

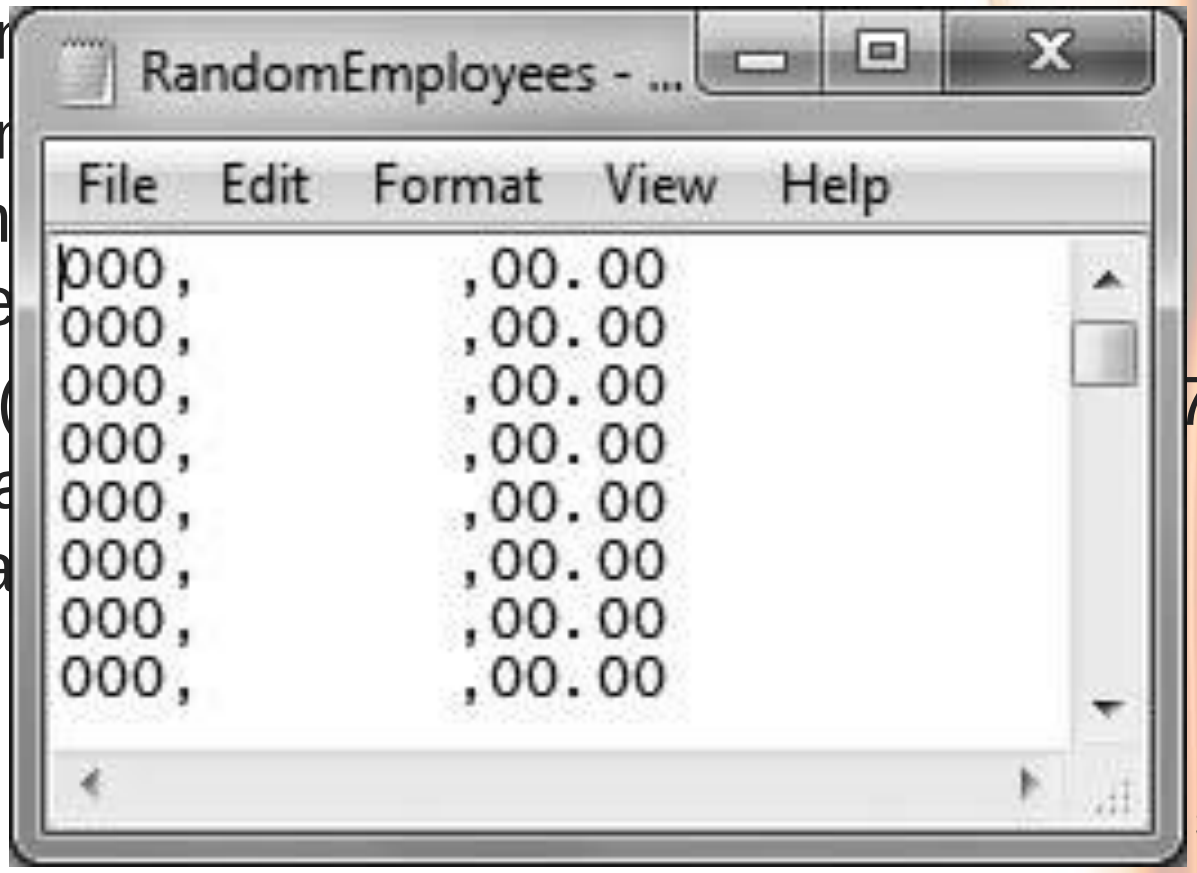
- Access a particular record

```
fc.position((n-1) * r-size);
```

- Place records into file based on key field
- Key field
  - Field that makes record unique from all others

# Writing Records to Employees.txt Randomly

- The first step create a file that holds default records for example, using zeroes for the ID numbers and pay rates, ar
- For this exar number is th than 1,000 e
- Record size( blanks ) + pa separator va



```
import java.nio.file.*;
import java.io.*;
import java.nio.ByteBuffer;
import static java.nio.file.StandardOpenOption.*;
public class CreateEmptyEmployeesFile
{
   public static void main(String[] args)
   {
      Path file =
         Paths.get("C:\\Java\\Chapter.13\\RandomEmployees.txt");
      String s = "000,        ,00.00" +
         System.getProperty("line.separator");
      byte[] data = s.getBytes();
      ByteBuffer buffer = ByteBuffer.wrap(data);
      FileChannel fc = null;
      final int NUMRECS = 1000;
      try
      {
         OutputStream output = new
            BufferedOutputStream(file.newOutputStream(CREATE));
         BufferedWriter writer = new
            BufferedWriter(new OutputStreamWriter(output));
         for(int count = 0; count < NUMRECS; ++count)
             writer.write(s, 0, s.length());
         writer.close();
      }
      catch(Exception e)
      {
          System.out.println("Error message: " + e);
      }
   }
}
```

Figure 13-30 The CreateEmptyEmployeesFile class

# Writing Records to Employees.txt Randomly (Cont.)

- Now you can replace any of records with data for an actual employee.
- locate the correct position for the new record by performing arithmetic with the record's key field.

```
fc.position((n-1) * R.length());
```

```
import java.nio.file.*;
import java.io.*;
import java.nio.channels.FileChannel;
import java.nio.ByteBuffer;
import static java.nio.file.StandardOpenOption.*;
import java.util.Scanner;
public class CreateEmployeesRandomFile
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      Path file =
         Paths.get("C:\\Java\\Chapter.13\\RandomEmployees.txt");
      String s = "000,       ,00.00" +
         System.getProperty("line.separator");
      final int RECSIZE = s.length();
      FileChannel fc = null;
      String delimiter = ",";
      String idString;
      int id;
      String name;
      String payRate;
      final String QUIT = "999";
      try
      {
         fc = (FileChannel)file.newByteChannel(READ, WRITE);
         System.out.print("Enter employee ID number >> ");
         idString = input.nextLine();
         while(!(idString.equals(QUIT)))
         {
            id = Integer.parseInt(idString);
            System.out.print("Enter name for employee #" +
               id + " >> ");
            name = input.nextLine();
            System.out.print("Enter pay rate >> ");
            payRate = input.nextLine();
            s = idString + delimiter + name + delimiter +
               payRate + System.getProperty("line.separator");
            byte[] data = s.getBytes();
            ByteBuffer buffer = ByteBuffer.wrap(data);
            fc.position(id * RECSIZE);
            fc.write(buffer);
            System.out.print("Enter next ID number or " +
               QUIT + " to quit >> ");
            idString = input.nextLine();
         }
         fc.close();
      }
```

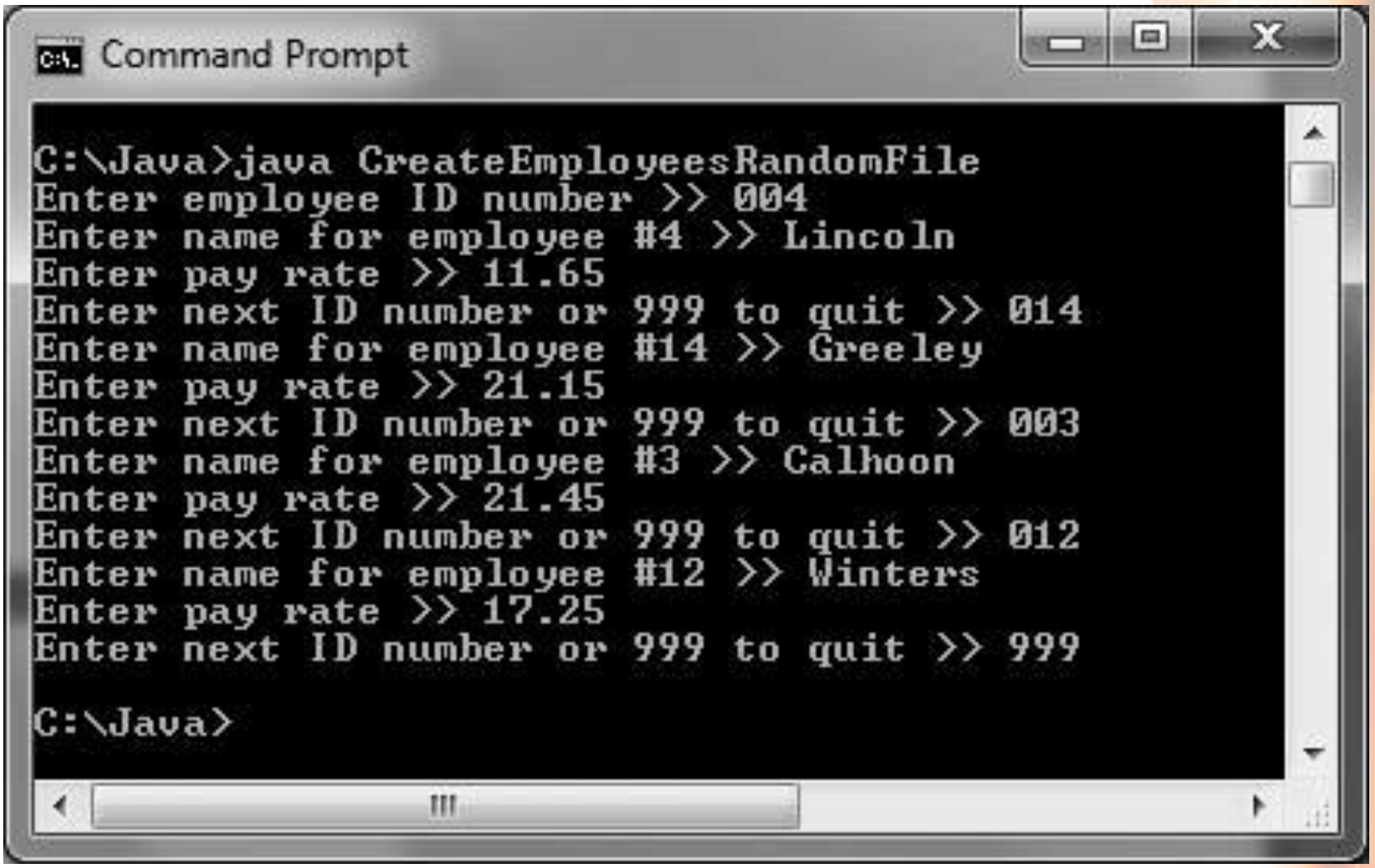
**Figure 13-34** The `CreateEmployeesRandomFile` class *(continues)*

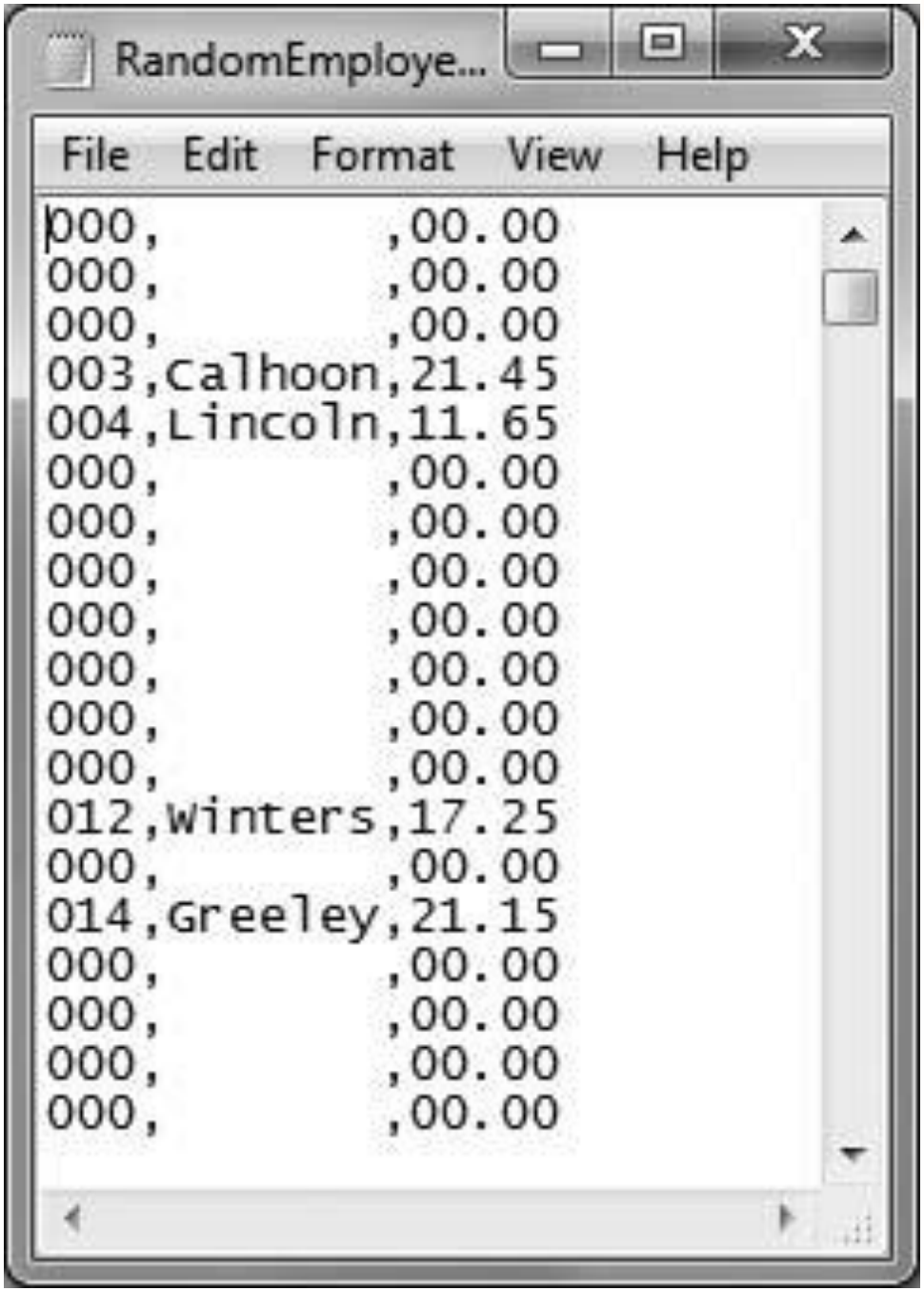
# Writing Records to a Random Access File (cont'd.)

*(continued)*

```
        catch (Exception e)
        {
            System.out.println("Error message: " + e);
        }
    }
}
```

**Figure 13-34** The CreateEmployeesRandomFile class


Command Prompt
C:\Java>java CreateEmployeesRandomFile
Enter employee ID number >> 004
Enter name for employee #4 >> Lincoln
Enter pay rate >> 11.65
Enter next ID number or 999 to quit >> 014
Enter name for employee #14 >> Greeley
Enter pay rate >> 21.15
Enter next ID number or 999 to quit >> 003
Enter name for employee #3 >> Calhoon
Enter pay rate >> 21.45
Enter next ID number or 999 to quit >> 012
Enter name for employee #12 >> Winters
Enter pay rate >> 17.25
Enter next ID number or 999 to quit >> 999
C:\Java>

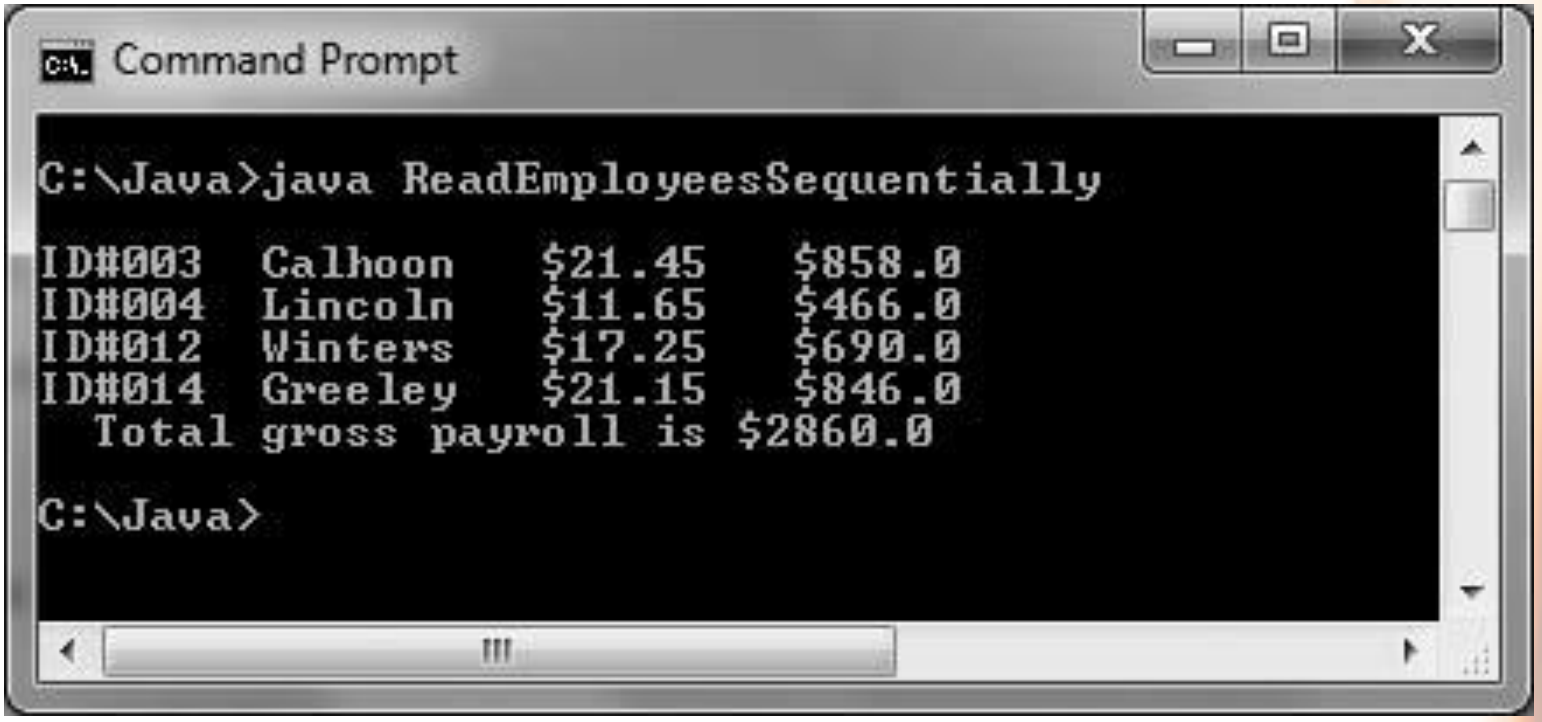

RandomEmploye...
File Edit Format View Help
000, ,00.00
000, ,00.00
000, ,00.00
003,Calhoon,21.45
004,Lincoln,11.65
000, ,00.00
000, ,00.00
000, ,00.00
000, ,00.00
000, ,00.00
000, ,00.00
000, ,00.00
012,Winters,17.25
000, ,00.00
014,Greeley,21.15
000, ,00.00
000, ,00.00
000, ,00.00
000, ,00.00

# Reading Records from a Random Access File

- A random access file can be accessed
  - Sequentially
  - Randomly

# Accessing a Random Access File Sequentially

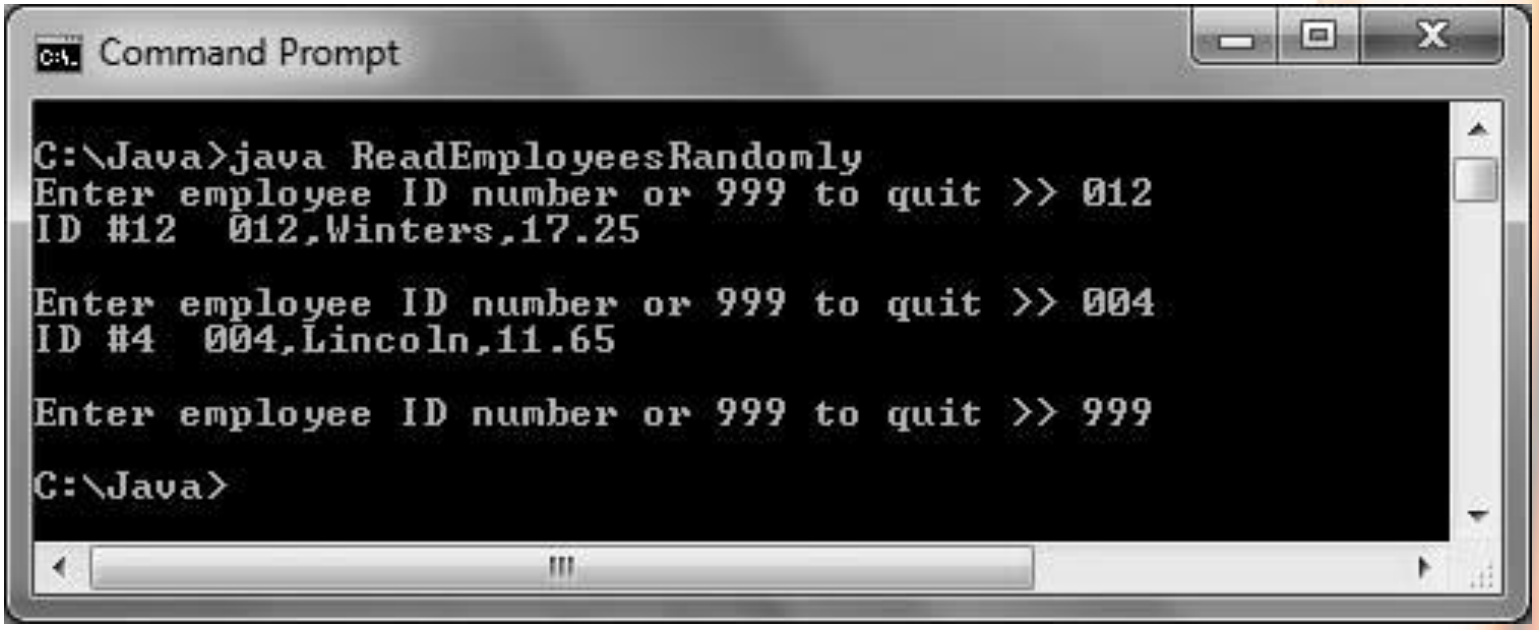
- `ReadEmployeesSequentially` application
  - Reads through 1000-record RandomEmployees.txt file sequentially in a for loop (shaded)
  - When ID number value is 0
    - No user-entered record stored at that point
    - Application does not bother to print it
- Figure 13-37 The ReadEmployeesSequentially class


Command Prompt
C:\Java>java ReadEmployeesSequentially
ID#003 Calhoon $21.45 $858.0
ID#004 Lincoln $11.65 $466.0
ID#012 Winters $17.25 $690.0
ID#014 Greeley $21.15 $846.0
Total gross payroll is $2860.0
C:\Java>

# Reading Records from a Random Access File

- Benefit of using random access file
  - Retrieve specific record from file directly Without reading through other records

- Figure 13-39 The ReadEmployeesRandomly class


Command Prompt
C:\Java>java ReadEmployeesRandomly
Enter employee ID number or 999 to quit >> 012
ID #12  012,Winters,17.25
Enter employee ID number or 999 to quit >> 004
ID #4  004,Lincoln,11.65
Enter employee ID number or 999 to quit >> 999
C:\Java>