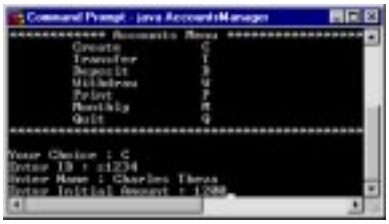
Files

- **Text and Binary Files**
- **Reading and Writing to files**
- **Extending the Accounts manager Program**

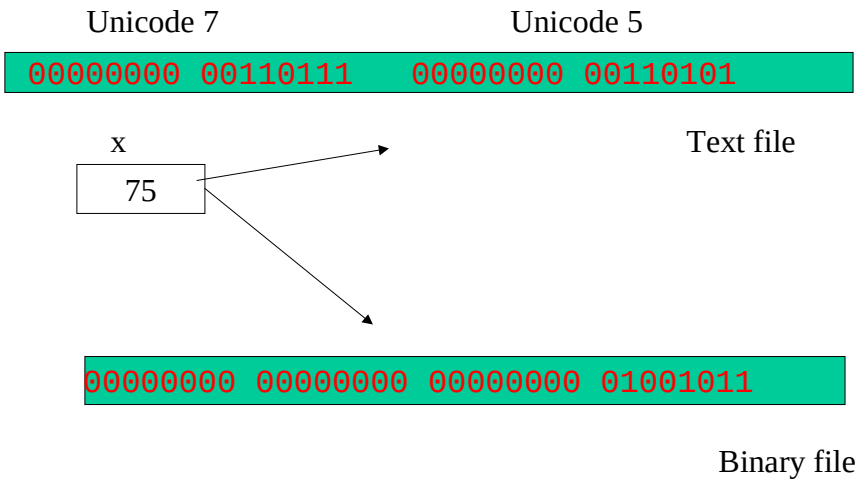
Files

- **In the AccountsManager program we have just finished we had to input all the record details through the keyboard. We need to do this every time we start the program!**
- **Surely there must be a better way ?**
- **Write all the information to a file so that it can be read back.**
- **A sequence of data elements residing in secondary storage (usually disk), is called a *file*.**




```
Account
1234Tom Cruise1200.0
SAccount
1111Mike Cheng800.0
800.0
SAccount
1111Peter Smith800.0
800.0
```

Binary and Text Files



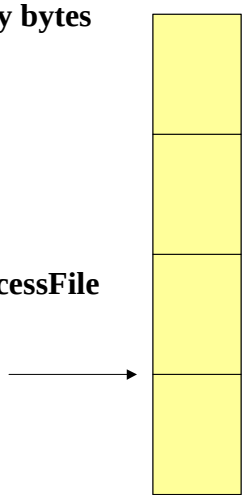
Random Access Files

If each record is 100 bytes how many bytes should I skip to read 4th record ?

To read nth record ?

Use the seek() method of RandonAccessFile specifying the offset.

Why is it useful ?



Reading a Text File - displaying it on the screen

```
import java.io.*;
public class FileToScreen  {
   public static void main(String[] args)
      throws IOException  {

       // create buffered file stream
     BufferedReader bufr = new BufferedReader
                 (new FileReader ("source.txt"));
     String inputLine;

     inputLine = bufr.readLine();

     while (inputLine != null)  {  //  more input
         System.out.println(inputLine);
         inputLine = bufr.readLine();
     }
   }  // main
}
```

File name

FileReader instead of InputStreamReader

source.txt

```
Dickery Dickery Dock. The
mouse ran up the clock.
The clock strucked one and
```

Copying One File to Another

```
import java.io.*;
public class FileToFile2  {
   public static void main(String[] args)
      throws IOException  {

       // create buffered file stream
       BufferedReader bufr = new BufferedReader
                 (new FileReader ("source.txt"));
       // create new buffered output writer

       PrintWriter pw = new PrintWriter (new
          BufferedWriter(new FileWriter ("dest.txt")));


       String inputLine;

      inputLine = bufr.readLine();
      while (inputLine != null)  {  // still more input
         pw.println(inputLine);
         inputLine = bufr.readLine();
      }
      pw.close();  // flushes buffers & frees resources
   }  // main
}
```

Create a FileWriter object chain it to BufferedWriter and then to PrintWriter.

Using println() of PrintWriter.

Numbers and other types of data in text files

```
// writes numbers, booleans to a file as text
import java.io.*;
public class BinaryToTextFile  {
   public static void main(String[] args)
      throws IOException  {

     //  create new buffered output writer
     PrintWriter pw = new PrintWriter(new
       BufferedWriter(new FileWriter ("dest.txt")));

     boolean flag = true;
     int anInt = 17;
     double aDouble = 123.45;

     pw.println(flag);
     pw.println(anInt);
     pw.println(aDouble);

     pw.close();
   }  // main
}
```

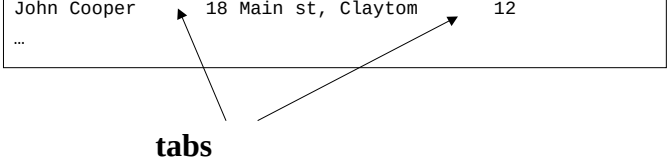
Overloaded println() method

Writing a records with multiple fields

```
import java.io.*;
public class WriteStudentsInfo{
  public static void main(String[] args)
          throws IOException {
    String name, address, fileName;
    int age;
    System.out.println("Enter the name of the file : ");
    ConsoleReader console = new ConsoleReader(System.in);
    fileName = console.readLine();
    PrintWriter pw = new PrintWriter(new BufferedWriter
                                  (new FileWriter(fileName)));
    do {
        System.out.println("Enter name : ");      name = console.readLine();
        System.out.println("Enter address: ");   address = console.readLine();
        System.out.println("Enter age : ");         age = console.readInt();
        pw.println(""+name+"\t"+address+"\t"+age);
        System.out.println("Continue  Y/N ?");
     } while ( console.readLine().charAt(0) == 'Y');
     pw.close();
  }
}
```

Format of the output file and Reading the values back

```
Mark Gossage     12 Betulaav, Mill Park    18
John Cooper      18 Main st, Claytom       12
…
```



tabs

To reading these fields back we use a StringTokenizer class, whose constructor takes a String and a delimeter (“\t”)

```
StringTokenizer inReader =
      new StringTokenizer(line,"\t");
```



```
import java.io.*;   import java.util.*;
public class ReadStudentsInfo
{  public static void main(String[] args)
            throws IOException {
              String line;   String name;  String address;  String fileName;   int age;
              System.out.println("Name of file to read from : ");
              ConsoleReader console = new ConsoleReader(System.in);
              fileName = console.readLine();
              BufferedReader br = new BufferedReader(new FileReader(fileName));
              System.out.println("name \t address \t age");
              while ( (line = br.readLine()) != null) {
                 StringTokenizer inReader = new StringTokenizer(line,"\t");
                 if (inReader.countTokens() != 3)
                          throw new IOException("Invalid Input Format");
                 else {
                    name = inReader.nextToken();
                    address = inReader.nextToken();
                    age = Integer.parseInt(inReader.nextToken());
                 }
                 System.out.println(name+"\t"+address + "\t" + age);
              }
              br.close();
   }
}
```

Extending the Accounts Manager Program

- **As different type of accounts will be saved in the same file, we will write the *class* name to the file, so that the objects can be recreated correctly when read back.**
- **To promote *code reuse* and *easy maintenance* we will create two methods in the `Account` superclass, `writeAttributes()` and `readAttributes()` to write and read the *instance variables*.**
- **These methods will be overridden in the *subclasses* to write and read any *additional instance variables*.**
- **A typical format of a file storing the account objects is given below.**

accounts.dat

```
Account
1234Tom Cruise1200.0
SAccount
1111Mike Cheng800.0
800.0
SAccount
1111Peter Smith800.0
800.0
```


What are the changes ?

- **AccountsManager class has a method `manageWritings()` to write all the objects referenced by the array**
- **This method will call `writeObject()` on all the account objects referenced by the array.**
- **The method `writeObject()` writes the class name and then calls the `writeAttributes()` to write the instance variables.**
- **The AccountsManager*constructor* is modified to read the *class* name from the file, construct the corresponding object using a *default constructor* and then call the `readAttributes()` to read the values.**
- **Once created their references are added to the array of Account references.**