Recap

- I/O Streams
- Readers and writers
- Reading text and numbers from **`System.in`**

Reading from Disk Files

- Use a **`FileReader`**

```
BufferedReader in =
  new BufferedReader(new FileReader("f.txt"));
String s;
int total = 0;
while ((s = in.readLine()) != null)
  { total += Integer.parseInt(s);
  }
in.close();
System.out.println("Total = " + total);
```

Writing to Disk Files

- Use a **`FileWriter`**
- Methods to write strings and characters

Going Further

- Much more to I/O in Java
  - Binary data files
  - Random access files
  - Reading and writing objects (*serialization*)
  - Network I/O
- Even database access (JDBC)

Chapter 11
More Classes and Objects

Important Object-Oriented Concepts

- Inheritance
- Method overriding
- Polymorphism

Inheritance

- Often classes (or objects) have a lot in common
  - e.g. bank accounts

Account

```
balance
accountNumber
interestRate
```

```
deposit, withdraw
```

Inheritance (cont.)

- Other types of accounts will be similar
  - e.g. cheque (current) account

ChequeAccount

```
balance, accountNumber
interestRate
overdraftLimit,
overdraftIntRate
```

```
deposit, withdraw
```

Problems

- Duplication of code and data
- Difficult maintenance
  - e.g. account number format changes
- More types of accounts
  - even more duplication

Inheritance — The Solution

- Classes can *inherit* properties from other classes
  - Like genetic inheritance

- In Java:

```
public class X extends Y
  {
  } // class X
```

Inheritance (cont.)

- Diagrammatically:

Inheritance (cont.)

- Cheque account class:

```
public class ChequeAccount extends Account
  { public double overdraftLimit;
    public double overdraftIntRate;
  } // class ChequeAccount
```

- This class *inherits*:
  - balance, account number and interest rate
  - withdraw and deposit methods

Some Terminology

- Superclass (parent class)
- Subclass (child class)
- Subclassing (inheriting)

- In Java: only one superclass
  - *single inheritance*

Another Example

Overriding Methods

- Sometimes a subclass needs to do things a little differently to the superclass
  - e.g. withdrawals from a cheque account
- *Method overriding*
  - replaces inherited method
  - it is *not* the same as overloading

Overriding Methods (cont.)

```
public class ChequeAccount extends Account
  { public double overdraftLimit;
    public double overdraftIntRate;

    public void withdraw (double amount)
      { ... // Allow for overdraft
      } // withdraw

  } // class ChequeAccount
```