Objective

Advanced Input/output

- Reader and Writers
- Binary input/output
- Random Access
- Object input and output streams

Review: 11.1 and 11.2
Study Chapter 21 (web only) of BiG Java: Early Object, 6th Edition.

Review

Reading Text Files

- Simplest way to read text: use `Scanner` class
- To read from a disk file, construct a `FileReader`
- Then, use the `FileReader` to construct a `Scanner` object

```
FileReader reader = new FileReader("input.txt");
Scanner in = new Scanner(reader);
```

Use the `Scanner` methods to read data from file

- `next`, `nextLine`, `nextInt`, and `nextDouble`

Review

Writing Text Files

- To write to a file, construct a `PrintWriter` object

```
PrintWriter out = new PrintWriter("output.txt");
```

- If file already exists, it is emptied before the new data are written into it
- If file doesn't exist, an empty file is created

Continued…

Review

Writing Text Files

- Use `print` and `println` to write into a `PrintWriter`:

```
out.println(29.95);
out.println(new Rectangle(5, 10, 15, 25));
out.println("Hello, World!");
```

- You must close a file when you are done processing it:

```
out.close();
```

- Otherwise, not all of the output may be written to the disk file

Review

Text Files

- When reading text file use Scanner
- When writing text use PrintWriter class.
- You must close all files when you are done processing them.

Review

A Sample Program

- Reads all lines of a file and sends them to the output file, preceded by line numbers
- Sample input file:

Check Example 1

```
Mary had a little lamb
Whose fleece was white as snow.
And everywhere that Mary went,
The lamb was sure to go!
```

- Program produces the output file:

```
/* 1 */ Mary had a little lamb
/* 2 */ Whose fleece was white as snow.
/* 3 */ And everywhere that Mary went,
/* 4 */ The lamb was sure to go!
```

Review

Self Check

- What happens when you supply the same name for the input and output files to the `LineNumberer` program?
- What happens when you supply the name of a nonexistent input file to the `LineNumberer` program?

Review

Answers

- When the `PrintWriter` object is created, the output file is emptied. Sadly, that is the same file as the input file. The input file is now empty and the while loop exits immediately.
- The program catches a `FileNotFoundException`, prints an error message, and terminates.

Text and Binary Formats

- Two ways to store data:
  - Text format
  - Binary format

Text Format

- Human-readable form
- Sequence of characters
  - Integer 12,345 stored as characters '1' '2' '3' '4' '5'
- Use **`Reader`** and **`Writer`** and their subclasses to process input and output
- To read:

```
FileReader reader = new FileReader("input.txt");
```

- To write

```
FileWriter writer = new FileWriter("output.txt");
```

Text Format

The reader class has a method, read, to read a single character at a time.

At the end of input, read returns -1.

```
FileReader reader = new FileReader("input.txt");
int next = reader.read();
char c;
if (next != -1)
  c = (char) next;
```

Binary Format

- Data items are represented in *bytes*
- Integer 12,345 stored as a sequence of four bytes `0 0 48 57`
- Use **`InputStream`** and **`OutputStream`** and their subclasses
- More compact and more efficient

Continued…

Binary Format

- To read:

```
FileInputStream inputStream
      = new FileInputStream("input.bin");
```

```
InputStream reader = ...
int next = reader.read();
byte b;
if (next != -1)
  b = (byte) next;
```

- To write   Be careful of typecasting Check Example negative

```
FileOutputStream outputStream
      = new FileOutputStream("output.bin");
```

Text and Binary Format

- Job of `FileInputStream`: interact with files and get bytes
- To read numbers, strings, or other objects, combine class with other classes

  example:

```
DataInputStream  fs= new DataInputStream(
                                        new FileInputStream("data.dat"));
```

Self Check

- Suppose you need to read an image file that contains color values for each pixel in the image. Will you use a `Reader` or an `InputStream`?
- Why do the read methods of the `Reader` and `InputStream` classes return an `int` and not a `char` or `byte`?

Answers

- Image data is stored in a binary format–try loading an image file into a text editor, and you won't see much text. Therefore, you should use an `InputStream`.
- They return a special value of -1 to indicate that no more input is available. If the return type had been `char` or `byte`, no special value would have been available that is distinguished from a legal data value.

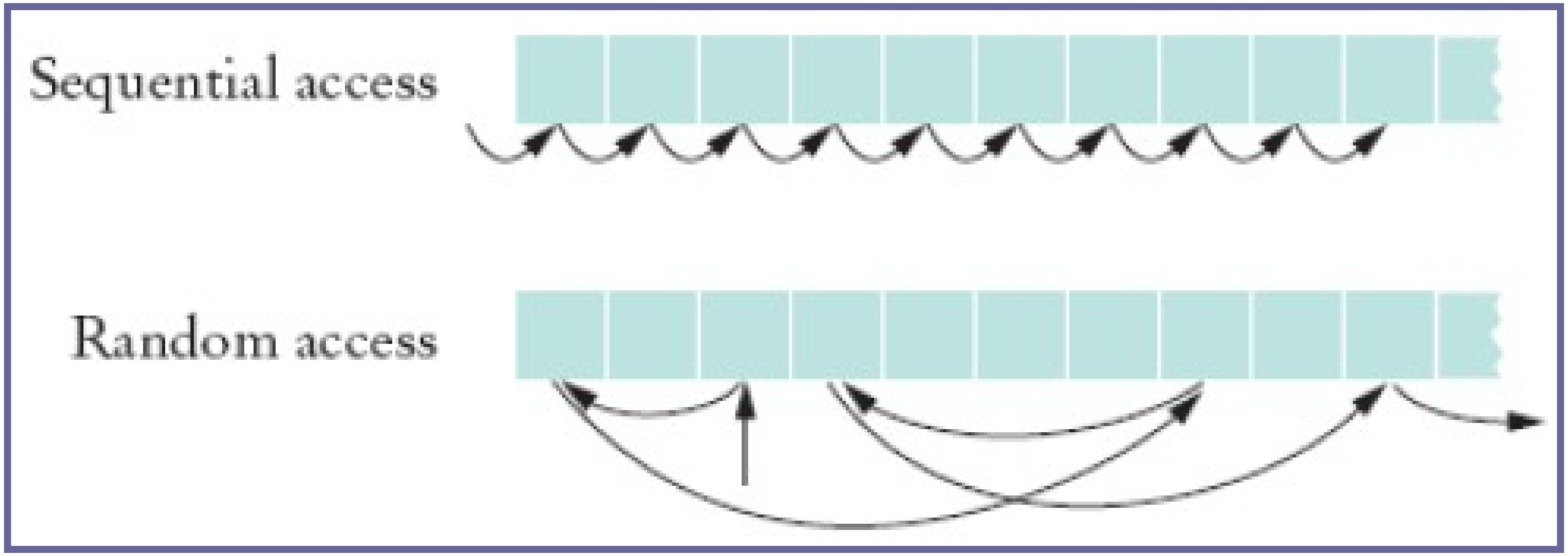
Random Access vs. Sequential Access

- Sequential access
  - A file is processed a byte at a time
  - It can be inefficient
- Random access
  - Allows access at arbitrary locations in the file
  - Only disk files support random access
    - `System.in` and `System.out` do not
  - Each disk file has a special file pointer position
    - You can read or write at the position where the pointer is

Continued…

Random Access vs. Sequential Access

- Each disk file has a special file pointer position
  - You can read or write at the position where the pointer is



Random and Sequential Access

RandomAccessFile

- You can open a file either for
  - Reading only ("**r**")
  - Reading and writing ("**rw**")

```
RandomAccessFile f = new RandomAcessFile("bank.dat","rw");
```

- To move the file pointer to a specific byte

```
f.seek(n);
```

Continued…

RandomAccessFile

- To get the current position of the file pointer.

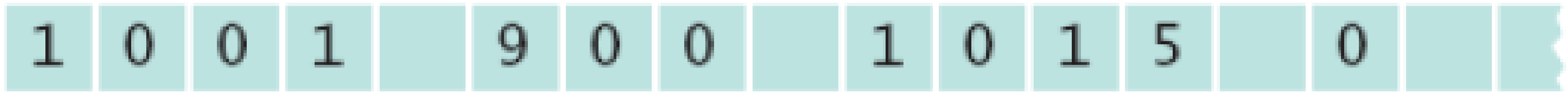
```
long n = f.getFilePointer();
      // of type "long" because files can be very large
```

- To find the number of bytes in a file long

```
fileLength = f.length();
```

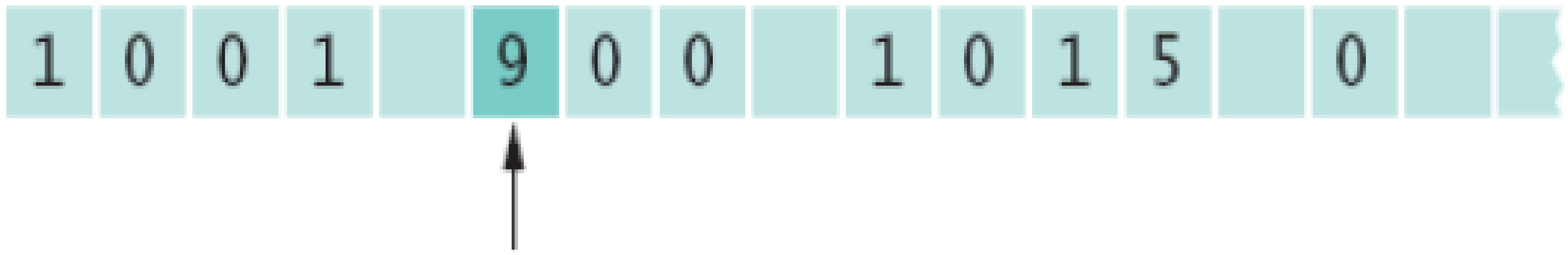
A Sample Program

- Use a random access file to store a set of bank accounts
- Program lets you pick an account and deposit money into it
- To manipulate a data set in a file, pay special attention to data formatting
  - Suppose we store the data as text
    Say account 1001 has a balance of $900, and account 1015 has a balance of 0

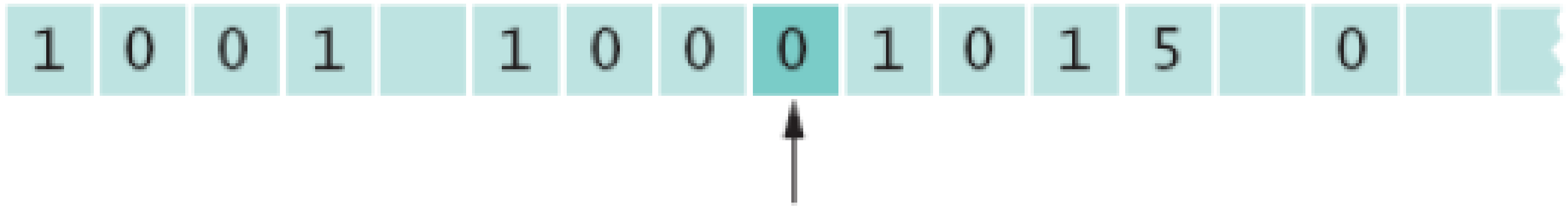


A Sample Program

We want to deposit $1000 into account 1001



If we now simply write out the new value, the result is



A Sample Program

- Better way to manipulate a data set in a file:
  - Give each value a fixed size that is sufficiently large
  - Every record has the same size
  - Easy to skip quickly to a given record
  - To store numbers, it is easier to store them in binary format

A Sample Program

- `RandomAccessFile` class stores binary data
- `readInt` and `writeInt` read/write integers as four-byte quantities
- `readDouble` and `writeDouble` use 8 bytes

```
double x = f.readDouble();
f.writeDouble(x);
```

Continued…

A Sample Program

- To find out how many bank accounts are in the file

```
public int size() throws IOException
{
   return (int) (file.length() / RECORD_SIZE);
         // RECORD_SIZE is 12 bytes:
         // 4 bytes for the account number and
         // 8 bytes for the balance }
```

A Sample Program

- To read the nth account in the file

```
public BankAccount read(int n)
      throws IOException
{
   file.seek((n-1) * RECORD_SIZE);
   int accountNumber = file.readInt();
   double balance = file.readDouble();
   return new BankAccount(accountNumber, balance);
}
```

A Sample Program

- To write the nth account in the file

```
public void write(int n, BankAccount account)
      throws IOException
{
   file.seek((n-1) * RECORD_SIZE);
   file.writeInt(account.getAccountNumber());
   file.writeDouble(account.getBalance());
}
```

Example

Example 2

Output:

```
Account number: 1001
Amount to deposit: 100
adding new account
Done? (Y/N) N
Account number: 1018
Amount to deposit: 200
adding new account
Done? (Y/N) N
Account number: 1001
Amount to deposit: 1000
new balance=1100.0
Done? (Y/N) Y
```

Self Check

- Why doesn't `System.out` support random access?
- What is the advantage of the binary format for storing numbers? What is the disadvantage?

Answers

- Suppose you print something, and then you call `seek(0)`, and print again to the same location. It would be difficult to reflect that behavior in the console window.
- Advantage: The numbers use a fixed amount of storage space, making it possible to change their values without affecting surrounding data. Disadvantage: You cannot read a binary file with a text editor.

Object Streams

- **`ObjectOutputStream`** class can save a

  entire objects to disk
- **`ObjectInputStream`** class can read objects back in from disk
- Objects are saved in binary format; hence, you use streams

Writing a `BankAccount` Object to a File

- The object output stream saves all instance variables

```
BankAccount b = . . .;
ObjectOutputStream out = new ObjectOutputStream(
      new FileOutputStream("bank.dat"));
out.writeObject(b);
```

Reading a `BankAccount` Object From a File

- **`readObject`** returns an `Object` reference
- Need to remember the types of the objects that you saved and use a cast

```
ObjectInputStream in = new ObjectInputStream(
      new FileInputStream("bank.dat"));
BankAccount b = (BankAccount) in.readObject();
```

Continued…

Reading a `BankAccount` Object From a File

- `readObject` method can throw a `ClassNotFoundException`
- It is a checked exception
- You must catch or declare it

Write and Read an `ArrayList` to a File

- Write

```
ArrayList<BankAccount> a = new ArrayList<BankAccount>;
// Now add many BankAccount objects into a
out.writeObject(a);
```

- Read

```
ArrayList<BankAccount> a = (ArrayList<BankAccount>)
      in.readObject();
```

Serializable

- Objects that are written to an object stream

  must belong to a class that implements the **`Serializable`** interface.

```
class BankAccount implements Serializable
{
   . . .
}
```

- `Serializable` interface has no methods.

Continued…

Serializable

- Serialization: process of saving objects to a stream
  - Each object is assigned a serial number on the stream
  - If the same object is saved twice, only serial number is written out the second time
  - When reading, duplicate serial numbers are restored as references to the same object

Example 3

First Program Run

```
1001:20100.0
1015:10000.0
```

Second Program Run

```
1001:20200.0
1015:10000.0
```

Check Example serial

Self Check

- What do you have to do to the `Coin` class so that its objects can be saved in an `ObjectOutputStream`?

Answers

- Add implements `Serializable` to the class definition.

Transient

You can create your own Serializable method

Declare instance variables as:

```
      private transient double x;
      private transient double y;
```

Over write methods:

```
 private void writeObject(ObjectOutputStream out)
                                          throws IOException{
        out.defaultWriteObject();
        out.writeDouble(x);
        out.writeDouble(y);
}
```

Continued…

Transient

Over right methods:

```
private void readObject(ObjectInputStream in) throws
        IOException, ClassNotFoundException{
                in.defaultReadObject();
                this.x = in.readDouble();
                this.y = in.readDouble();
}
```

Check Example transiet

Summary

- Select a data format
- Use readers and writers if your processing text
- Use streams if your processing byte
- If you use object stream, make your classes implements Serializable
- Use Object streams if you are processing objects