Mandelbrot Set

JonathanCoulton.com (free downloads)

Benoit Mandelbrot: Nov 1924-Oct 2010

As JoCo acknowledges at some of his concerts, the chorus is in fact the procedure for generating a Julia Set for a given C, not the Mandelbrot Set. For the Mandelbrot Set, the lyrics would become, “Take a point called Z in the complex plane, \Let Z1 be Z squared plus Z. \And Z2 is Z1 squared plus Z. \And so on…” .

Streams

- Files are a computer's long term memory
- Need ability for programs to
  - get information from files
  - copy information from program variables to files
- Java *streams* provide connection between variables internal to a program and external files

More generally,

streams manage flows of data between programs and any external entity that can provide or receive data

ex. System.out

Stream Classes

- No class actually named Stream
- Collection of classes that provide mechanism to transfer data between programs and other external entities
- Distinction made between input and output streams
- Distinction between streams that handle text versus other forms of data

Text Streams

Many files contain only simple text

- Game programs remember high scores in a text file
- Web browser keeps track of your bookmarks in a text file

Readers and Writers

- Readers: input streams specialized to handle text
- Writers: output streams specialized to handle text

Streams through which Java provides access to text files called

  - FileReader
  - FileWriter

Creating a Writer

```
FileWriter highScoreTable = new FileWriter( “highscores.txt” );
```

- creates FileWriter stream to place data in file named “highscores”
- file assumed to be in directory or folder current when program is run
- if file does not already exist, new file created.
- if file exists, current contents erased to be replaced by new data

```
// Place some text into notes.txt
 public void saveNoteFile(String[] strArray, int numStrings) {

    try {
      FileWriter aFile= new FileWriter(”highscores.txt");

      for ( int i = 0; i < numStrings; i++ ) {
        aFile.write( strArray[i] );
      }

      aFile.close();
    } catch ( IOException e ) {
      System.out.println(“Can’t save your scores. Bummer. ” + e.getMessage() );
    }
 }
```

FileReader class

- subclass of Reader
- has constructor that takes String file name as parameter

  FileReader bookMarksReader = new FileReader(“bookmarks.html”);
- read method reads 1 character at a time
- constructor, close and read may raise IOExceptions; need to handle with try-catch

BufferedReaders

- constructor accepts FileReader or any other subclass of Reader
- supports method readLine
  - each time invoked, next line from file is read
  - when no lines left, returns null

Most loops to process data from a BufferedReader take the form:

```
String  curLine = someBufferedReader.readLine();

while ( curLine != null ) {
        // Code to process one line from the file
        ...

        curLine = someBufferedReader.readLine();
}
```

Applets and Applications

Applets: Java programs designed to be downloaded through a web browser

Applications: Designed to be installed locally.

For security of user's files, web browsers generally do not allow applets to access files

If we're using hardware (e.g., a Scribbler), we're typically using applications. But BlueJ hides some of this for us.

Writing Applets and Applications

- In every Java program, one class functions as starting point of execution
- Applets
  - class must be a subclass of Applet class or JApplet class
- Applications
  - class must define a static method named main that expects a string array as a parameter and returns no value.

```
public static void main( String arguments[ ] )
```

public static void main(String arguments[ ])

- Not all classes are runnable
- Can pass in arguments from terminal
- BlueJ lets you skip this