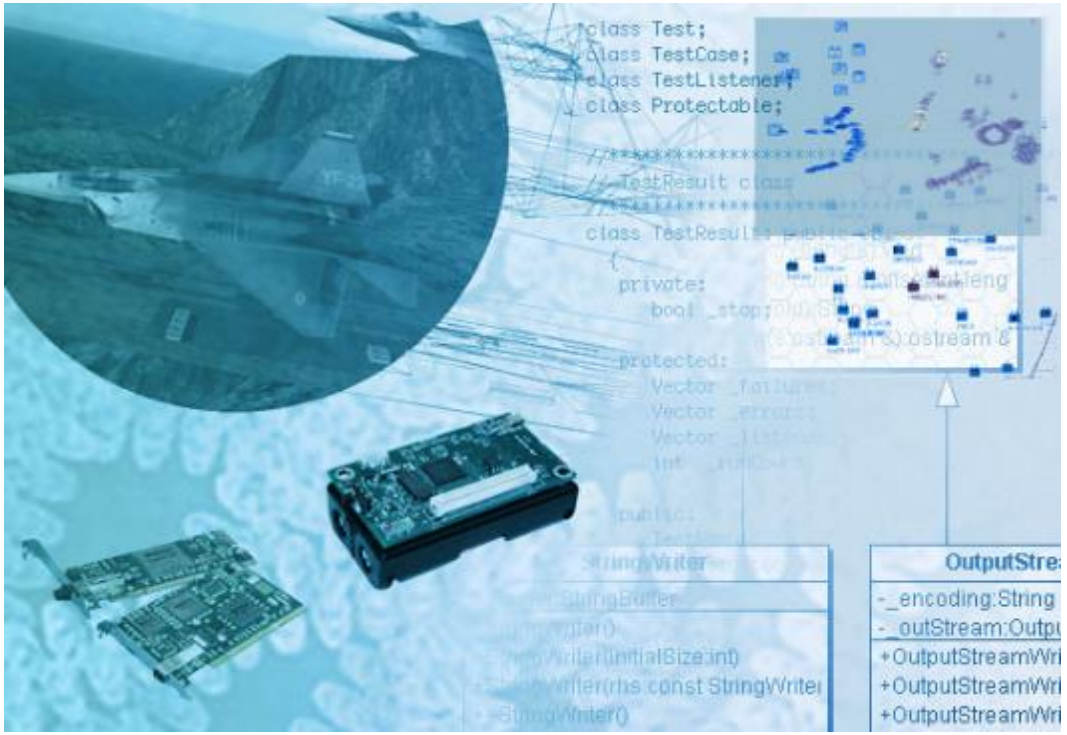
CSYE 6200

CONCEPTS OF OBJECT-ORIENTED DESIGN

SESSION 7

MARK G. MUNSON



ADMINISTRATION

- **Session 7 – Mar 8th:** Assignment #4 due
  - Mid-Term Exam this week (Ch. 1 – 10, UML, Coding)
- **Session 8 – Mar 15th**
- **Session 9 – Mar 22nd:** Assign 5a simulation due (not collected)
- **Session 10 – Mar 29th:** Assign 5b – diagram review
- **Session 11 – Apr 5th:** Quiz 3
- **Session 12 – Apr 12th:** Extra
  - Assign 5abc due this week
- **Session 13 – Apr 19th:** Full Review
- **Final Exam – Apr 26th** (Online)

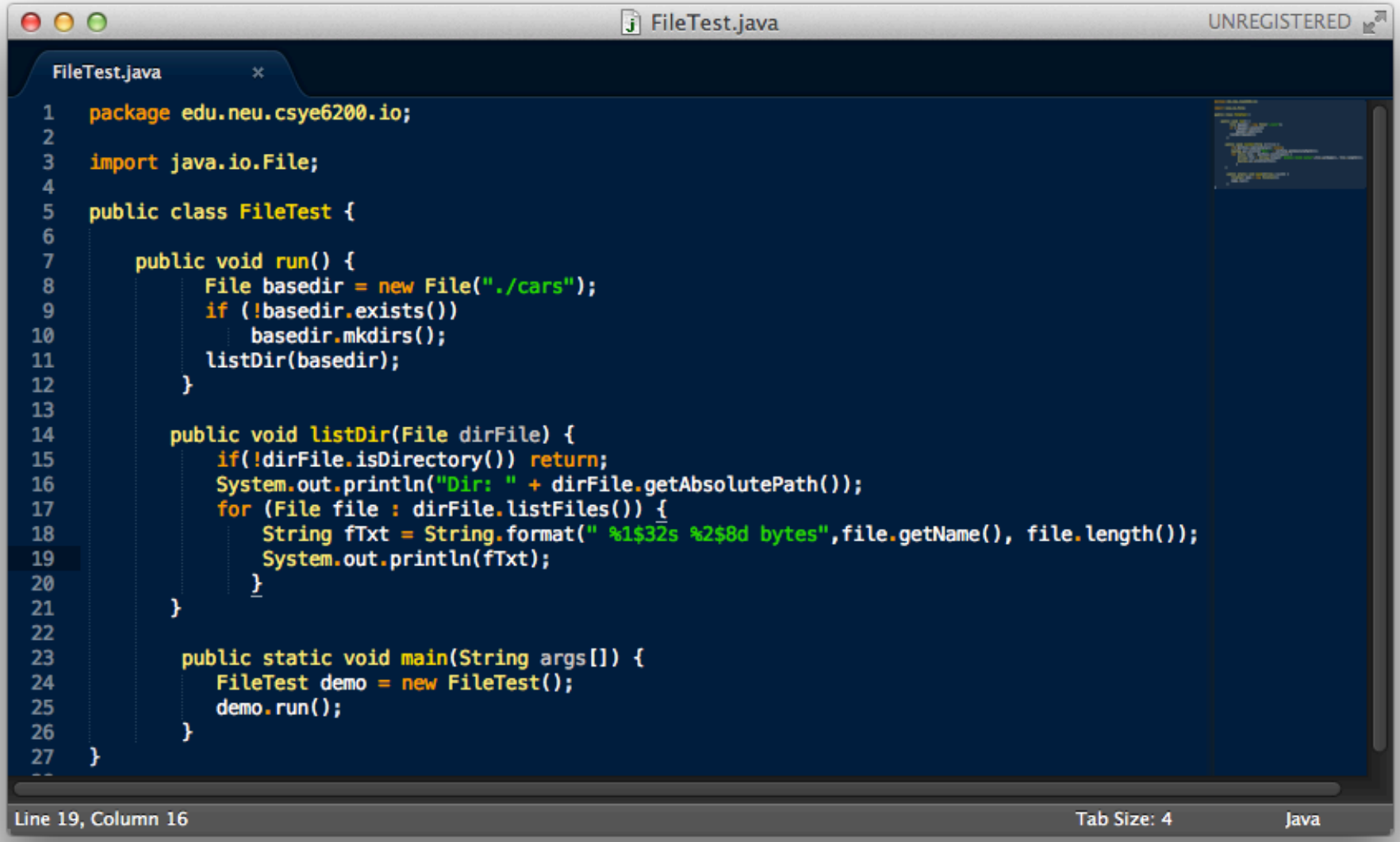
THE LECTURE

- **Recap**
- **Streaming**
  - Binary Input/Output
  - Reader/Writers
  - File IO – Serial Objects
- **Collections – sorting**
- **Generics < >**
- **Assignment 5 Introduction**
- ***GUI Swing***
  - *Short intro to JFrame, JPanel, JButton, Layout Managers*

RECAP

FILE I/O
FILE CLASS

FILE TEST



~50 MINUTES

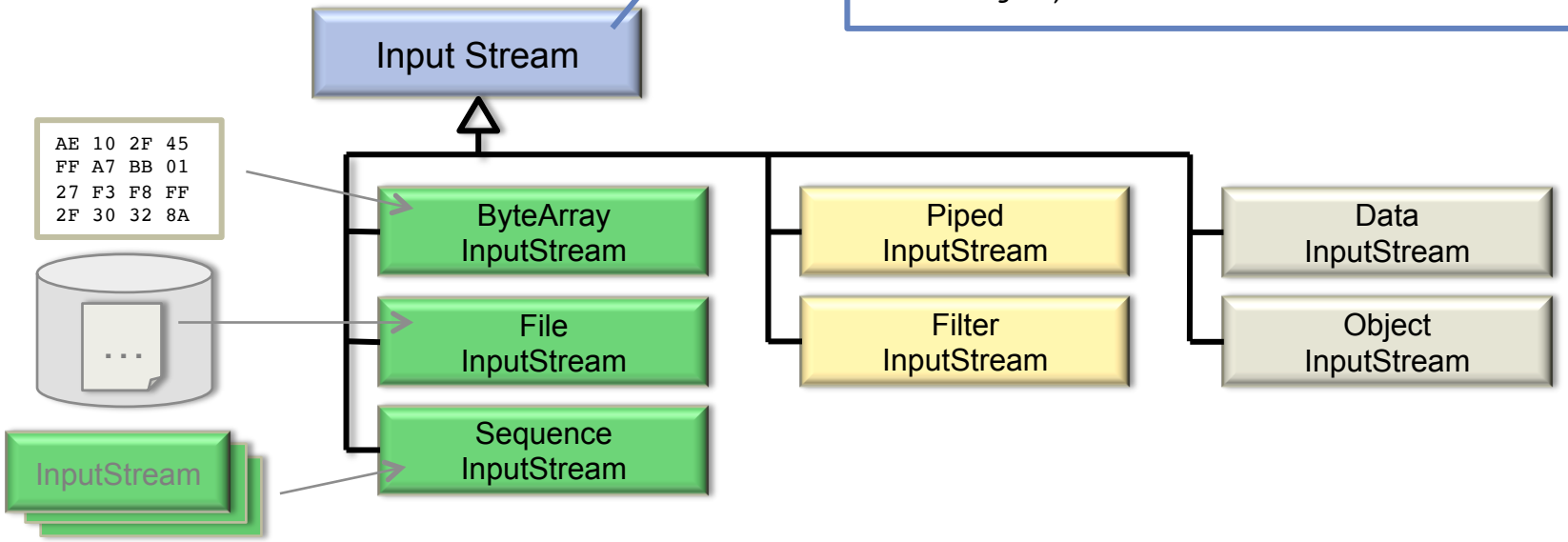
EXAM – END:

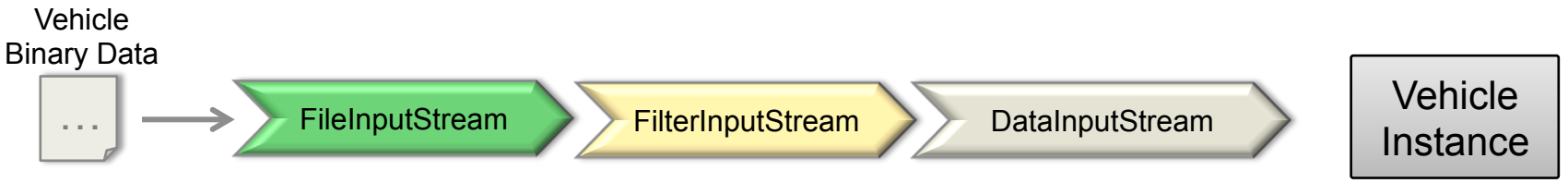
REMAINING: X MIN

I/O – STREAMS

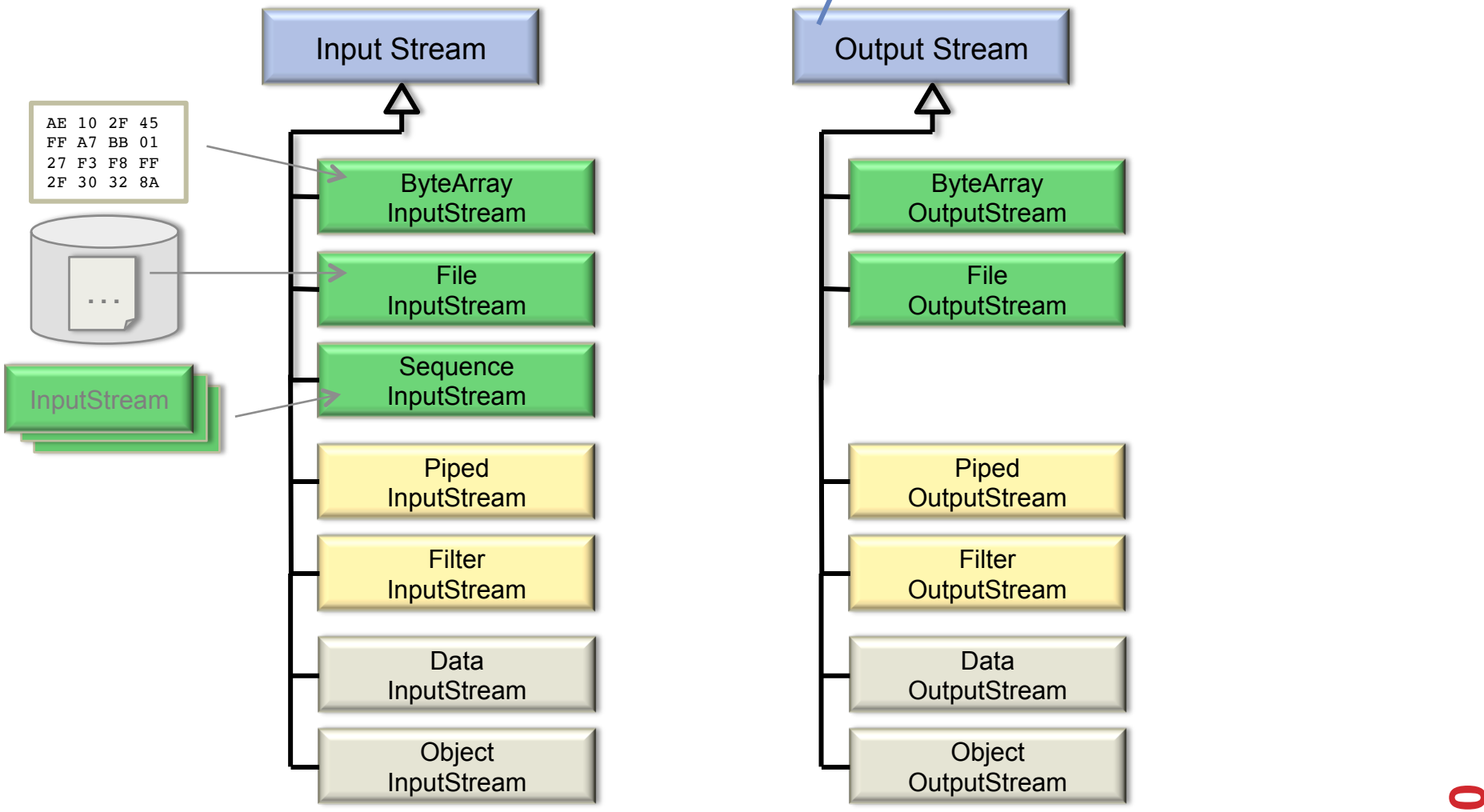
READING STREAMS

```
public int read()
public int read(byte barray[])
public int read(byte barray[], int offset,
int length)
```





INPUT AND OUTPUT STREAMS

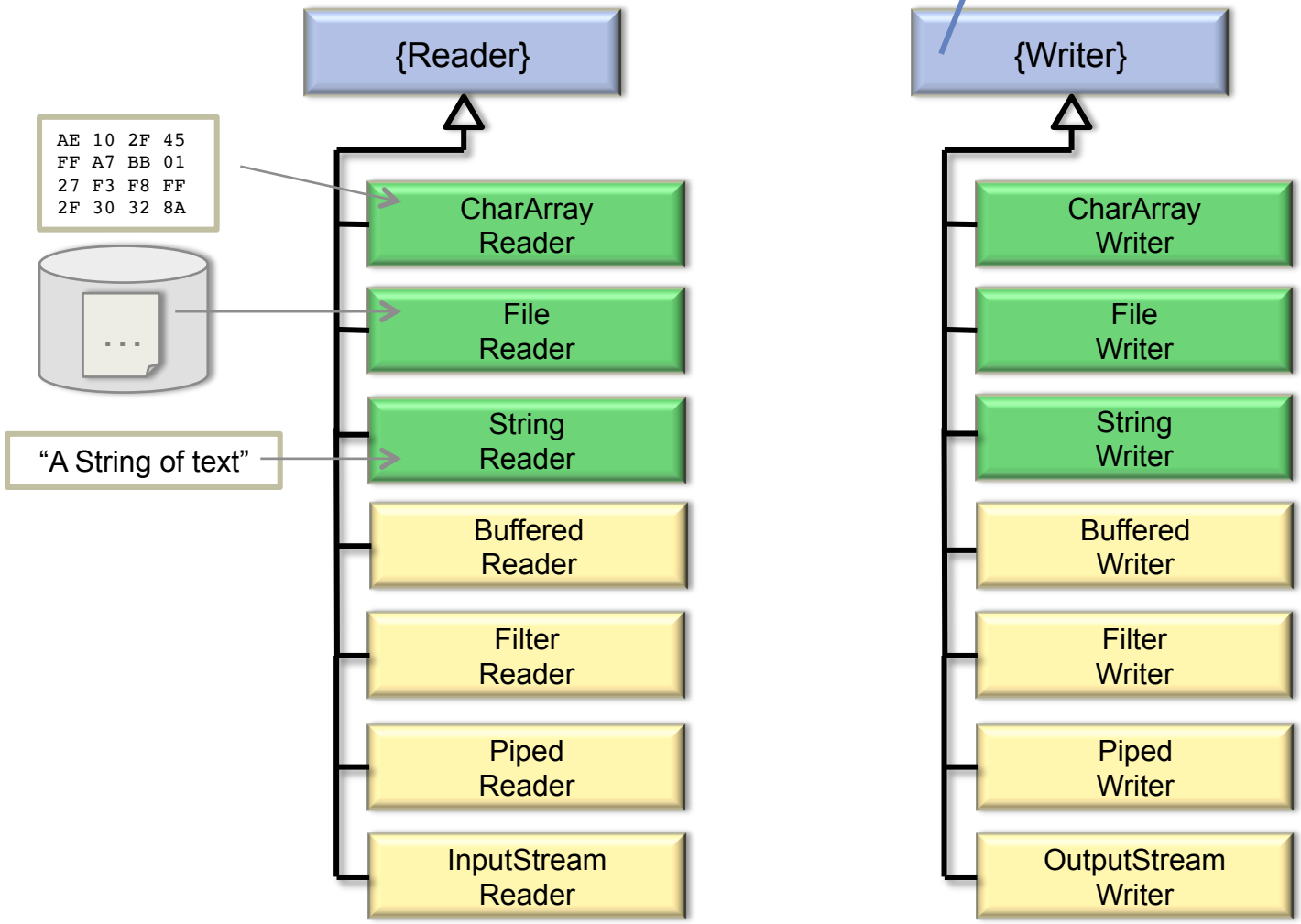


READER AND WRITER

- The standard IO classes work fine for binary data, but what about Unicode characters?
- Java implemented some special character classes, but eventually they created a whole new IO stack, dedicated to reading and writing character data
- To support this, they generated two abstract classes called Reader and Writer (java.io)

READING AND WRITING STREAMS

```
public void write(char cval)
public int write(char carray[])
public int write(char carray[],
int offset, int length)
public int write(String str)
public int write(String str, int
offset, int length)
```



FILE I/O – READERS/WRITERS

FILEREADER / FILEWRITER

FileWriterTest.java — UNREGISTERED

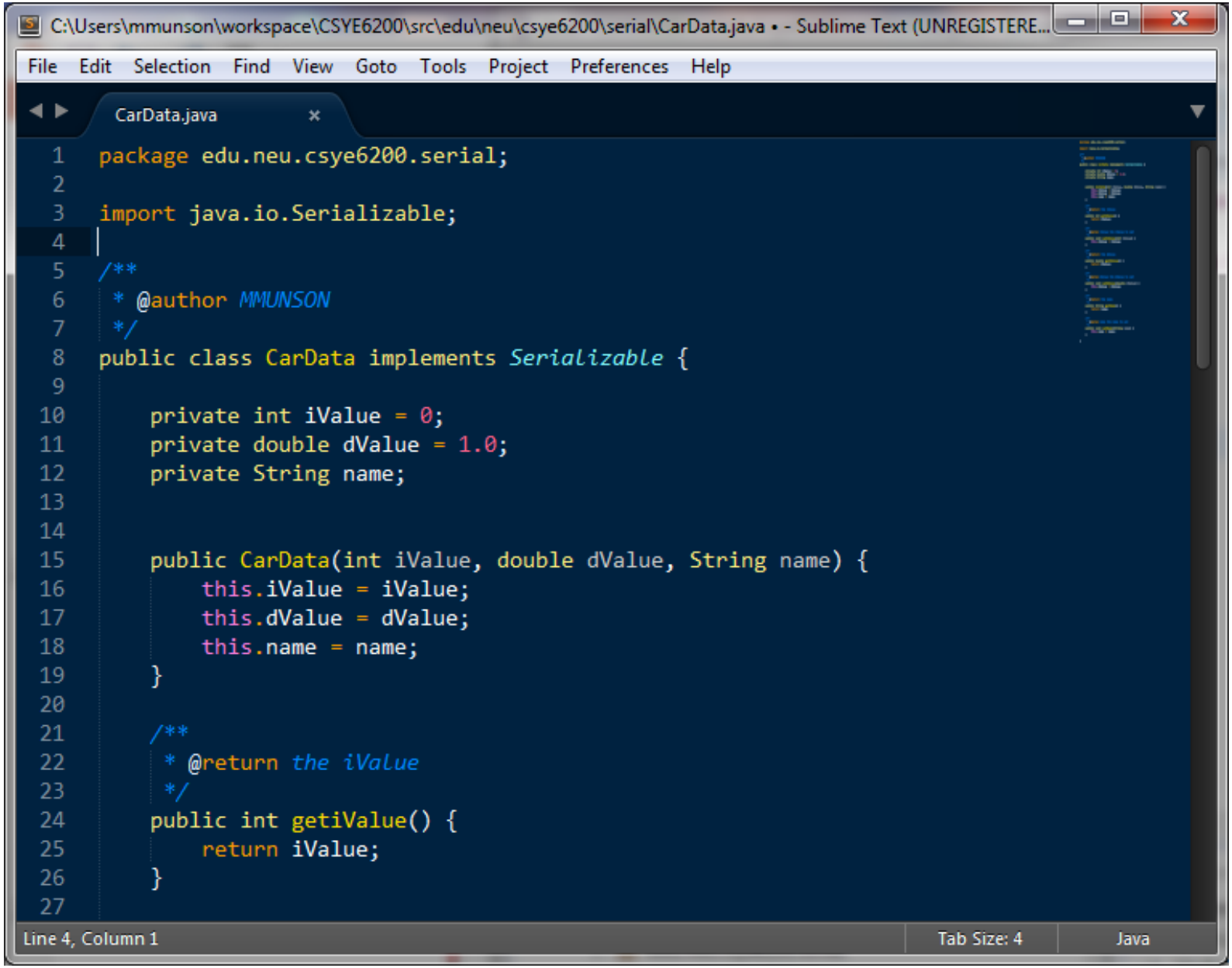
FileWriterTest.java

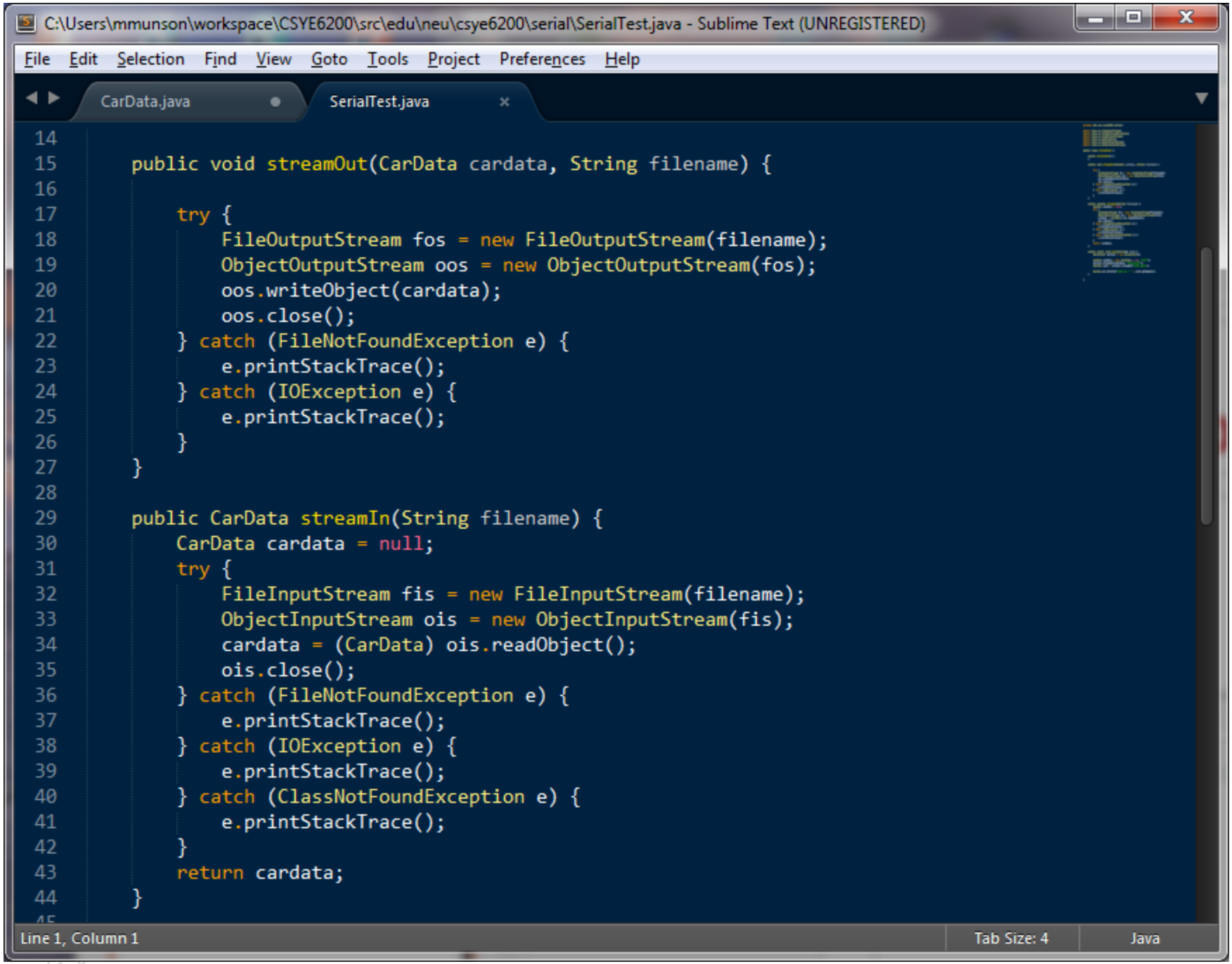
```
1   package edu.neu.csye6200.io;
2
3   import java.io.FileReader;
4   import java.io.FileWriter;
5   import java.io.IOException;
6
7   public class FileWriterTest {
8
9       public void run() {
10          String base = "src/edu/neu/csye6200/io/";
11          try {
12              FileReader reader = new FileReader(base+"FileWriterTest.java");
13              FileWriter writer = new FileWriter(base+"FWTDuplicate.txt");
14
15              int inVal = reader.read();
16              while (inVal >= 0) {
17                  writer.write(inVal);
18                  inVal = reader.read();
19              }
20              reader.close();
21              writer.close();
22          } catch (IOException e) {
23              e.printStackTrace();
24          }
25      }
26
27      public static void main(String[] args) {
28              FileWriterTest demo = new FileWriterTest();
29              demo.run();
30      }
31  }
```

Line 17, Column 37 — Tab Size: 4 — Java

READING AND WRITING SERIAL OBJECT

FILE I/O
SERIAL OBJECTS




C:\Users\mmunson\workspace\CSYE6200\src\edu\neu\csye6200\serial\SerialTest.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
CarData.java
SerialTest.java
14
15 public void streamOut(CarData cardata, String filename) {
16
17 try {
18 FileOutputStream fos = new FileOutputStream(filename);
19 ObjectOutputStream oos = new ObjectOutputStream(fos);
20 oos.writeObject(cardata);
21 oos.close();
22 } catch (FileNotFoundException e) {
23 e.printStackTrace();
24 } catch (IOException e) {
25 e.printStackTrace();
26 }
27 }
28
29 public CarData streamIn(String filename) {
30 CarData cardata = null;
31 try {
32 FileInputStream fis = new FileInputStream(filename);
33 ObjectInputStream ois = new ObjectInputStream(fis);
34 cardata = (CarData) ois.readObject();
35 ois.close();
36 } catch (FileNotFoundException e) {
37 e.printStackTrace();
38 } catch (IOException e) {
39 e.printStackTrace();
40 } catch (ClassNotFoundException e) {
41 e.printStackTrace();
42 }
43 return cardata;
44 }
Line 1, Column 1
Tab Size: 4
Java

COLLECTIONS

COMPARABLE INTERFACE

C:\Users\mmunson\workspace\CSYE6200\src\edu\csye6200\sort\Vehicle.java • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

CarData.java | SerialTest.java | Vehicle.java

```java
package edu.neu.csye6200.sort;

public class Vehicle implements Comparable<Vehicle>{

    private final String license;

    public Vehicle(String license) {
        this.license = license;
    }

    @Override
    public int compareTo(Vehicle veh) {
        return license.compareTo(veh.license);
    }


    public String toString() {
        return ("Vehicle: " + license);
    }
}

```

Line 1, Column 17 | Tab Size: 4 | Java

COLLECTION SORTING

C:\Users\mmunson\workspace\CSYE6200\src\edu\csye6200\sort\ColSortExample.java • - Sublime Text (UNREGISTERE...

File Edit Selection Find View Goto Tools Project Preferences Help

ColSortExample.java

```
1  package edu.neu.csye6200.sort;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5
6  public class ColSortExample {
7
8      private ArrayList<Vehicle> vehList = null;
9
10     public ColSortExample() {
11         vehList = new ArrayList<Vehicle>();
12
13         vehList.add(new Vehicle("XYZ-789"));
14         vehList.add(new Vehicle("GEF-674"));
15         vehList.add(new Vehicle("ART-730"));
16         vehList.add(new Vehicle("PNN-003"));
17     }
18
19     public void sort() {
20         Collections.sort(vehList);
21     }
22
```

Line 13, Column 45 | Tab Size: 4 | Java

GENERICS < T >

GENERICS

- **In our original list example we always assumed the use of base Objects:**

```
ArrayList alist = new ArrayList();
ArrayList<Object> vlist = new ArrayList<Object>();
```

  - So the add and get methods would be:

```
public int add(Object obj) { … }
public Object get(int index) { … }
```

- **But with Generics, we can specify the assumed type**

```
ArrayList<Vehicle> vlist = new ArrayList<Vehicle>();
```

  - So the add and get methods would be:

```
public int add(Vehicle obj) { … }
public Vehicle get(int index) { … }
```

GENERICS

Classes with Generics are specified with the form:

class *name*<*type param-list*> { …

Example:

```
Public class Registry<T> {
  T val;
  public void setVal(T t) { val = t; }
  public T getVal() { return val; }
}
```

GENERICS

The form for creating a class instance with Generics is:

class-name<*type-arg-list*> var-name =

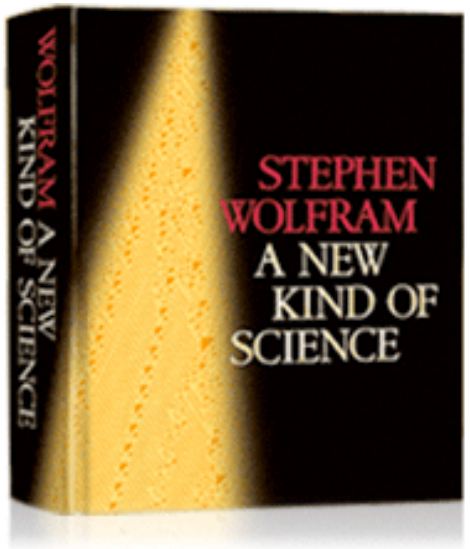
new *class-name*<*type-arg-list*>(*cons-arg-list*);

Example:

```
ArrayList<Vehicle> vlist = new ArrayList<Vehicle>();
…
Vehicle v0 = vlist.get(0);
Vehicle v1 = vlist.get(1);
```

COMPUTATIONAL SCIENCES

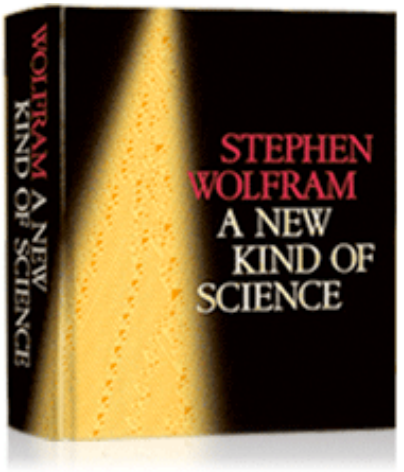
COMPUTATIONAL SCIENCES



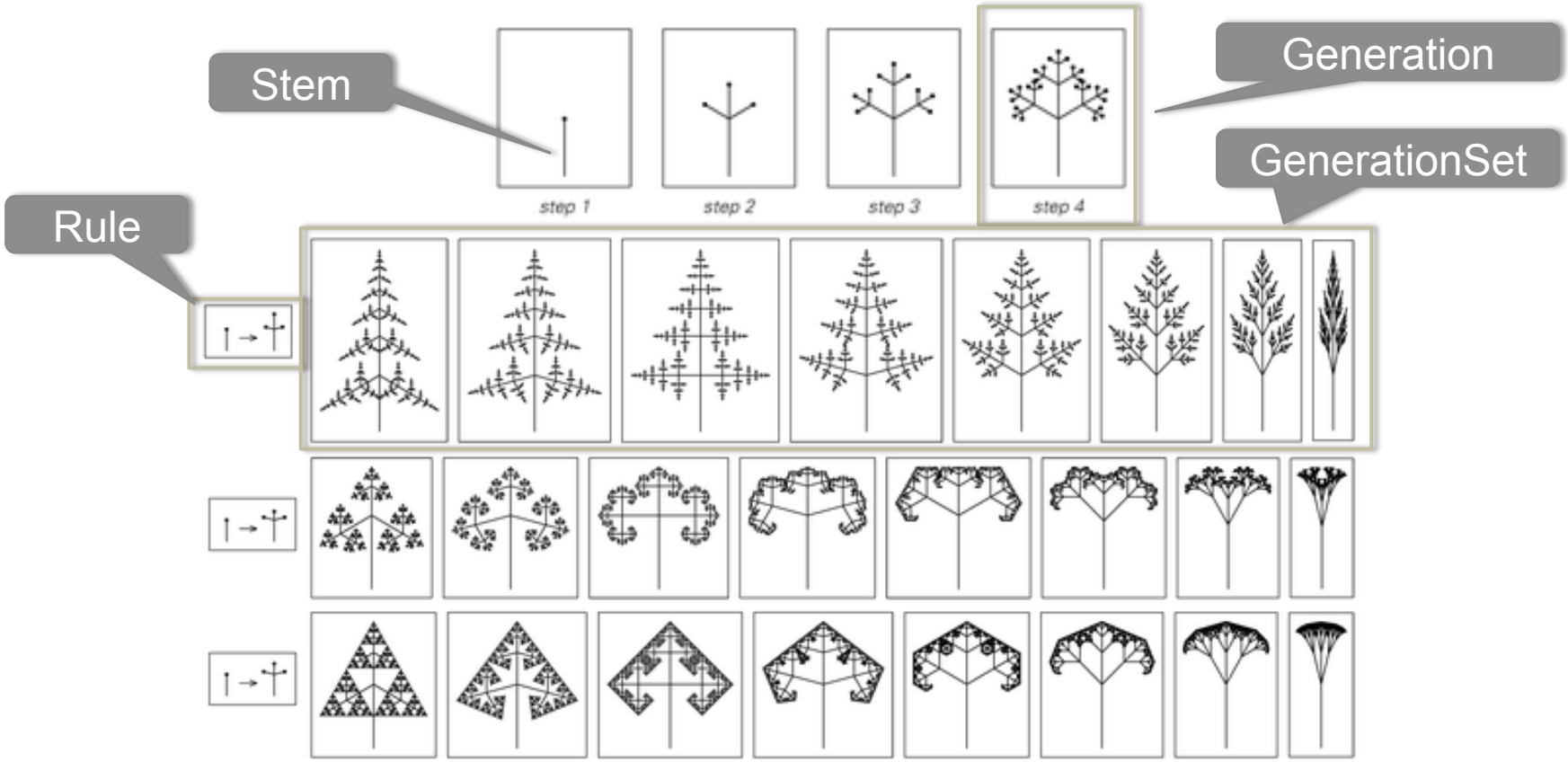
- **Cellular Automaton programming was popularized by Stephen Wolfram's *A New Kind of Science***
  - Available free online
  - http://www.wolframscience.com/nksonline/toc.html
  - iPad downloadable version $10

- **Chapter 8: Fundamental Issues in Biology pp. 383 – 422**

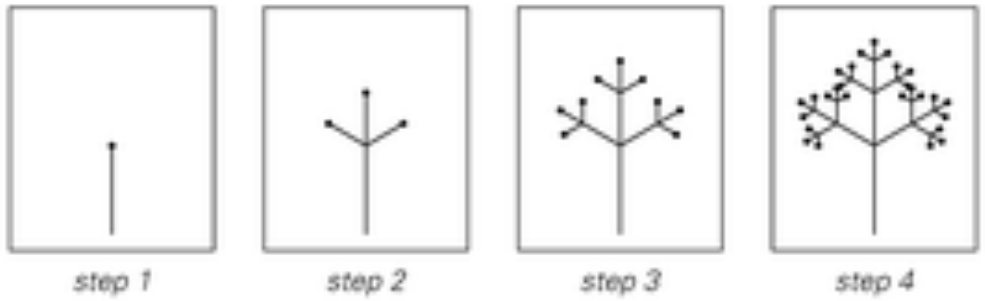
COMPUTATIONAL SCIENCES



- Chapter 8: Implications for Everyday Systems
  - Subsection: Growth of Plants and Animals



STEMS AND RULES

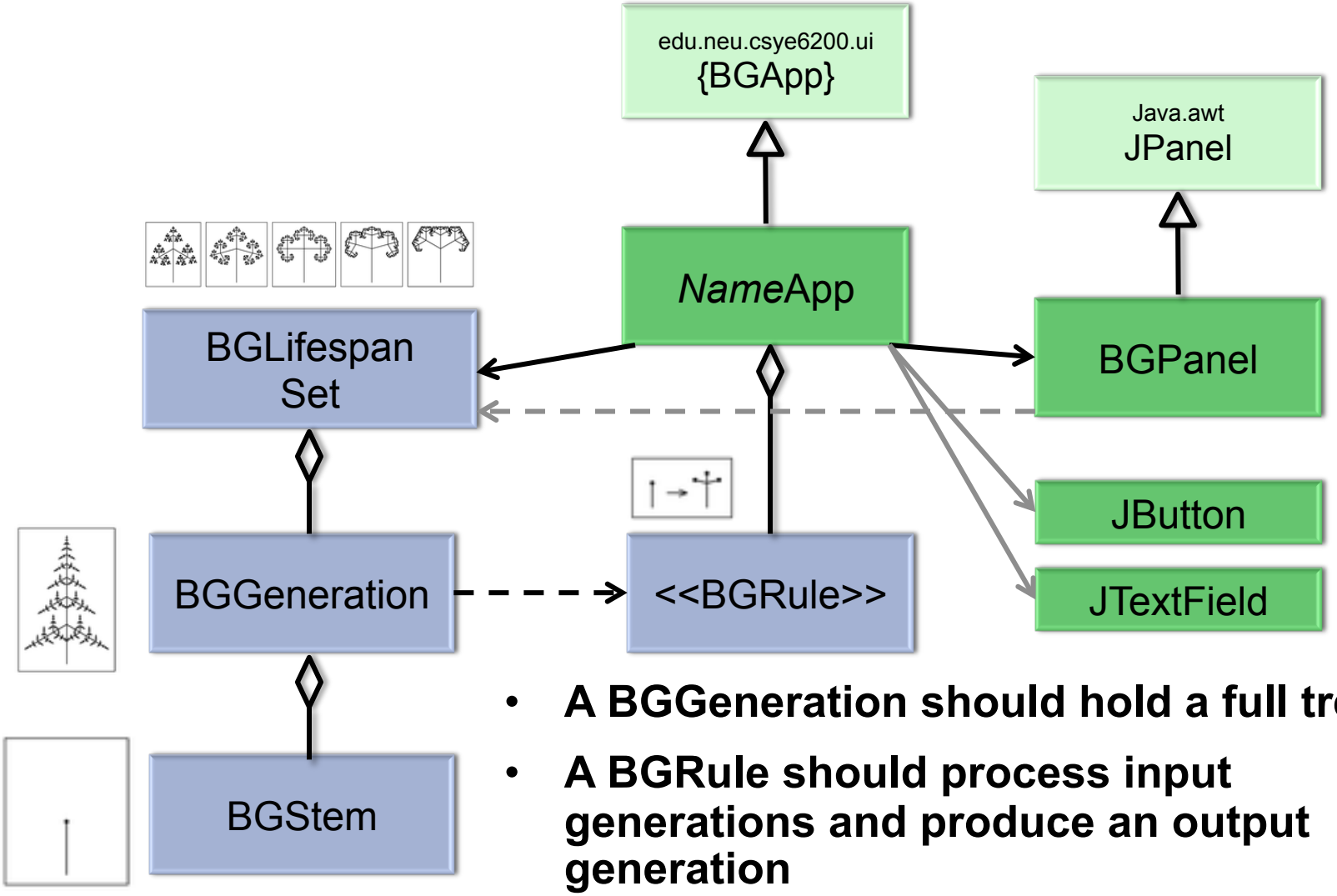


- **Stem**
  - A stem has an initial position and length
  - It may have child stems that originate at its tip
  - A stem may grow in length and/or width over time
  - Each child is another stem, which obeys the same rules
- **Rule**
  - Converts a stem into a ‘stem with child stems’
  - Can control the number and relative angles of each child stem
  - Can govern stem growth in terms of length and width
  - May create a new generation from an existing generation

BIOLOGICAL GROWTH (BG) DESIGN

- **Algorithm**
  - Stem (parent stem, start point, length, direction child stems)
  - Generation (a list of attached stems)
  - LifespanSet (a list of generations)
  - Rule (Biological Growth from one stage to the next)
- **User Interface**
  - UI Generation and/or GenerationSet Visualization
  - UI Application

BG APP STATIC CLASS DIAGRAM



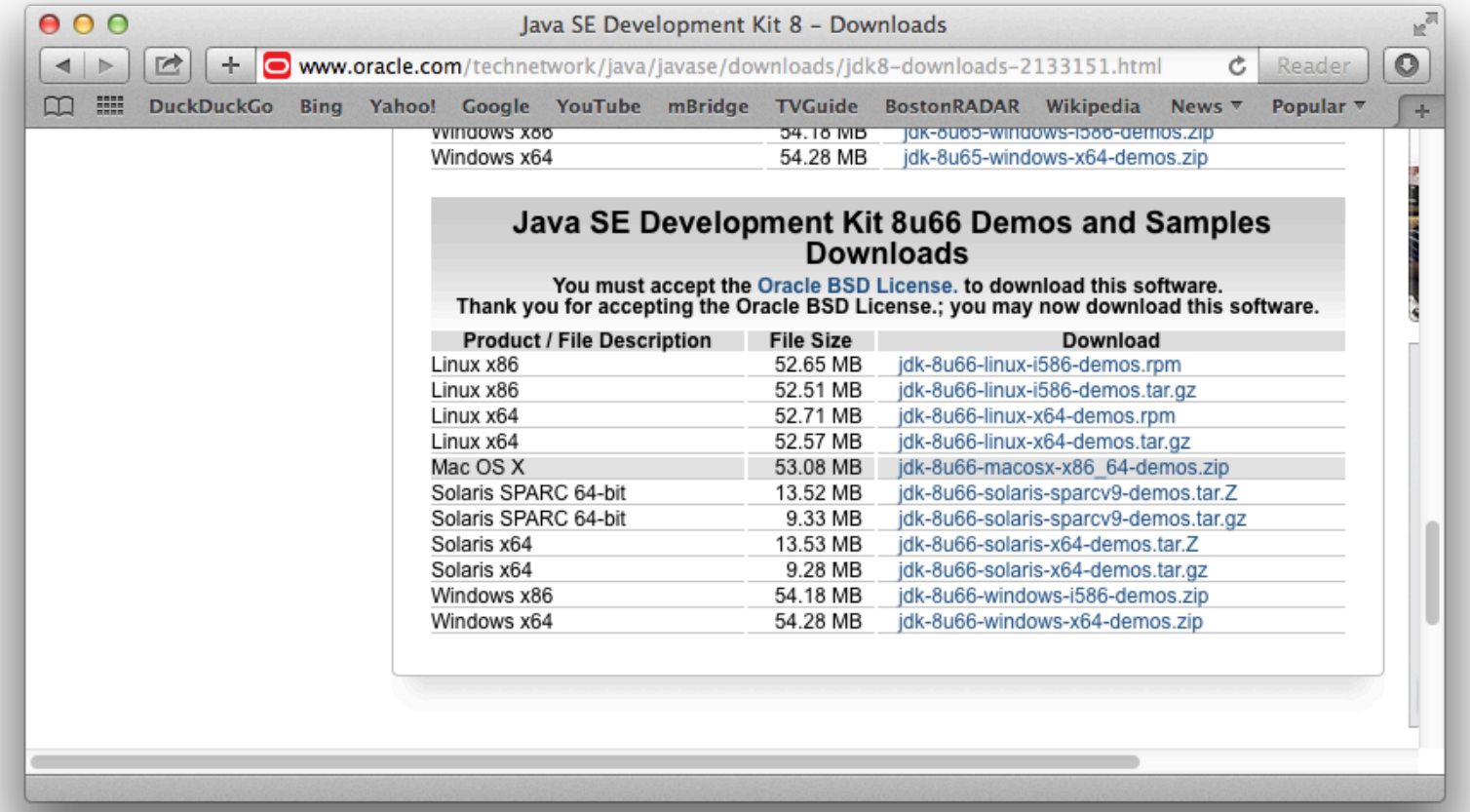
- **A BGGeneration should hold a full tree/leaf**
- **A BGRule should process input generations and produce an output generation**

GUI - SWING

SWING DEMO

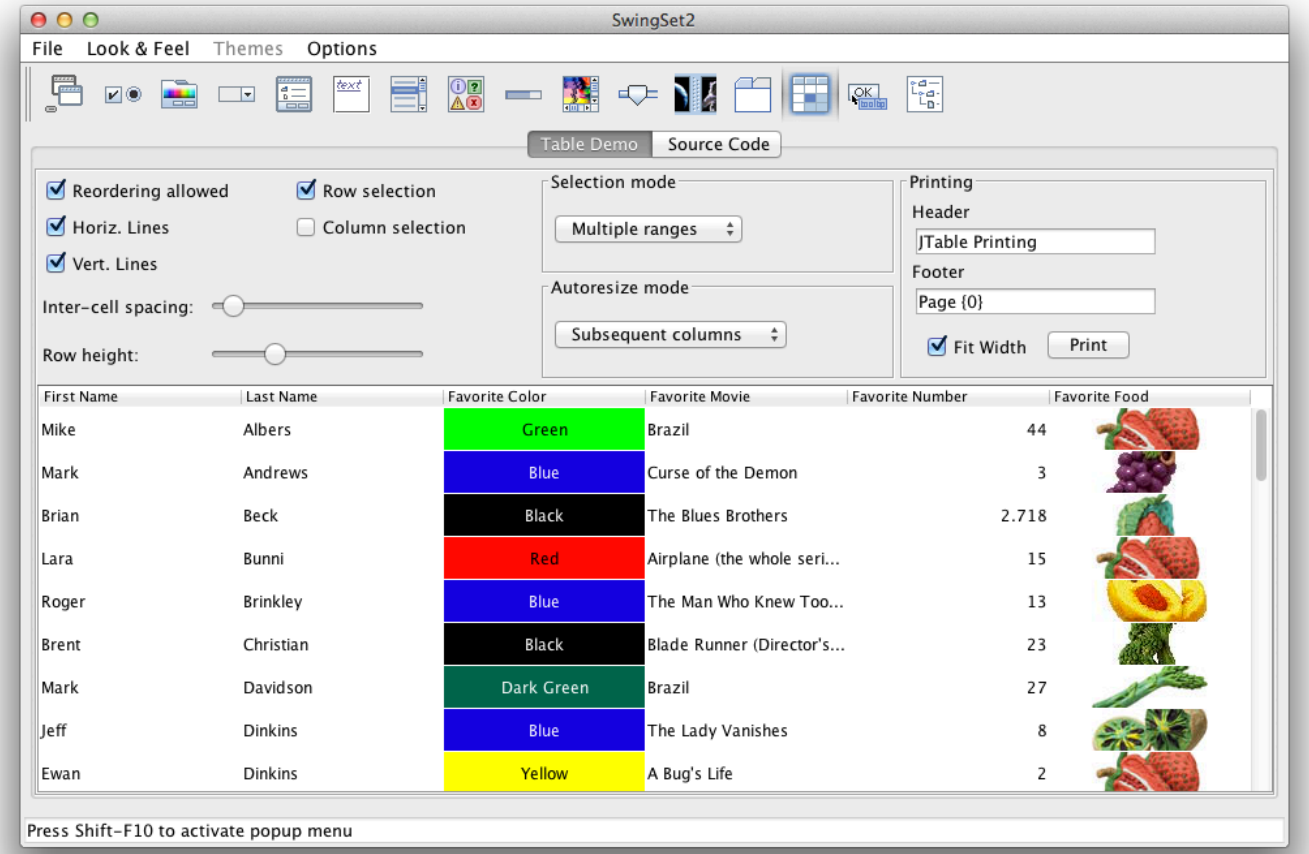
JAVA SAMPLE CODE

Oracle SE Download area - fetch the Demos and Samples



JAVA SWINGSET

In ./jdk1.8.0_112/demo/jfc/SwingSet2, run the SwingSet2.jar program (`> java –jar SwingSet2.jar`)



SWING COMPONENTS

- **All Java control widgets inherit from JComponent**
  - Provides a base class for the Swing architecture
  - Supports the pluggable look-and-feel at runtime
  - Supports ‘tool tips’, a pop-up display of a short description

- **Each component may be placed into a top-level container**
  - JFrame
    - Interfaces with AWT Frame resources
    - Handles the default look-and-feel
    - Handles the displayed window icon
  - JPanel – a generic lightweight container
  - JDialog – provides a modal pop-up window

LAYOUT MANAGERS

- **Swing has multiple layout managers that govern the placement of items within a container pane**

Manager	Description
FlowLayout	Simple left-to-right placement
BorderLayout	Default layout (north, south, east, west, center)
GridLayout	Grid based
GridBagLayout	A flexible grid layout
BoxLayout	Vertical or Horizontal layout
SpringLayout	Constrint based layout

- **For quality results, other open source layout managers should be used**

MORE SWING COMPONENTS

- **JLabel**

```
JLabel nameLbl = new JLabel("name:");
```

- **JTextField**

```
JTextField nameTF = new JTextField();
    nameTF.setText("Lastname");
```

- **JComboBox**

```
JComboBox<String> townCBox = new JComboBox<String>();
    townCBox.addItem("Boston");
    townCBox.addItem("Worcester");
```

MORE SWING COMPONENTS

- **JTextArea – a simple text display**

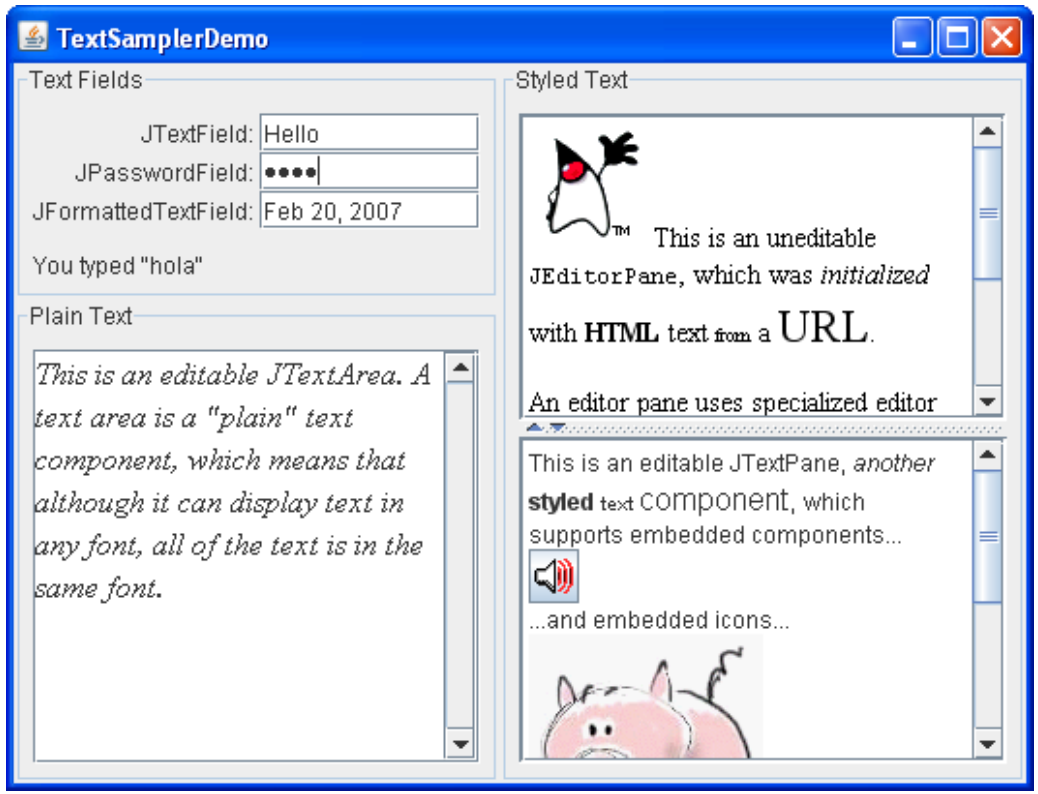
```
JTextArea descTA = new JTextArea(5, 80);
    descTA.setEditable(true);
    descTA.setLineWrap(true);
    descTA.setWrapStyleWord(true);
```

- **JScrollPane**

```
JScrollPane descSP = new JScrollPane(descTA);
    descSP.getVeiwport().add(descTA);
```

SWING TEXT EDITORS

Examples can be found at: https://docs.oracle.com/javase/tutorial/uiswing/components/editorpane.html



NEXT WEEK / ASSIGNMENT #5

JABG: Read
- Ch. 11 Threads
- Ch. 16 Java Swing/Events

Design Pattern: Observer

- **Assignment #5 Biological Growth [multi-part]**
  - Part A – Biological Growth (Part A Due Mar. 22nd) *Not Collected*
    - Create a package called edu.neu.csye6200.bg
    - Create supporting classes to perform Biological Growth calculations
    - Recommended tasks:
      - Create a Stem class which contains an array of child Stem instances plus length, direction, age, etc.
      - Create a BGRule class which can extend an existing Stem, and possibly create new generation of stems based on a Stem growth
      - Create a BGGenerationSet that holds multiple BG generations and can call the BGRule class repeatedly to "grow" successive generations.
        Question: will you keep copies of prior generations?
      - Add interfaces to collect growth data and generate statistics
      - Demonstrate valid growth