COS 126 N-body Assignment Overview
TYNTK for Nbody

- You will need to use a virtual terminal.
- A *shell* is a command-line interpreter in a terminal.
- "-introcs" enables our libraries in commands.
- Follow checklist, study assignment.
- Incremental debugging is a path to success.
- You need to critically evaluate advice.
- Overwhelmed? Avoid *rabbit holes*.
- Bored? Jump right in!

**Note.** You will not be tested on this content. ➡️ Except to the extent that ignoring it slows you down.
Virtual Terminals and Shells
For this assignment you will need a (virtual) terminal

You already own two virtual terminals

- An app on your computer.
- A pane in your IntelliJ window.

Note. Easiest to use IntelliJ pane for assignments, but realize that Terminal is generally useful.
A shell is a command-line interpreter for a terminal

- Main purpose: *Control your computer with a keyboard*
- Developed in 1970s in UNIX operating system.
- Many, many variations in use.
- Recent convergence to **bash** "Bourne-again shell".
- Still heavily used by most programmers.

Shell commands that you need to know now

- Manipulate files and folders on your computer.
- Invoke Java compiler and runtime.
- Use our Std* libraries
- Redirect input and output.

Review slides 15–18 in Lecture 4
Shell commands to manipulate files and folders

**Terminology.** A *folder* in the browser is a collection of files and folders.

**Convention.** When using the shell, we refer to a folder as a *directory*.

- `pwd`  
  print working directory name

- `ls`  
  list contents of working directory

- `cd X`  
  change to directory X

- `more`  
  show contents of file X

- `rm X`  
  delete file X

- `mkdir X`  
  create new directory, name it X

- `rmdir X`  
  remove directory X
Typical shell command sequence to manipulate files and folders

% pwd
/Users/rs

% cd Desktop

% ls
etc/
hello/

% mkdir COS126

% ls
COS126/
etc/
hello/

% mv hello COS126

% cd COS126

% ls hello
COS 126.iml
HelloWorld.java
readme.txt
logo.png
Shell commands to invoke Java compiler and runtime

```
javac X.java  compile Java program X.java
               creates file X.class

java X          invoke Java runtime for X.class
                input from std input, output to std output

java X < data    invoke Java runtime for X.class
                input from file data, output to std output

java X > result  invoke Java runtime for X.class
                input from std input, output to file result

java X < data > result  invoke Java runtime for X.class
                        input from file data, output to file result
```
Our software libraries implement simple abstractions for I/O

- For Nbody, you need StdIn, StdOut, StdDraw, and StdAudio.
- Review Lecture 4 and Section 1.5 in the textbook.
- Standard practice is to download the code, then tell Java where to find it.
- Our `introcs` shell commands save you the trouble of doing so.

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdIn</td>
<td>read numbers and text</td>
</tr>
<tr>
<td>StdOut</td>
<td>write numbers and text</td>
</tr>
<tr>
<td>StdDraw</td>
<td>draw geometric shapes</td>
</tr>
<tr>
<td>StdAudio</td>
<td>play and manipulate sound</td>
</tr>
</tbody>
</table>

Use these!

```
javac-introcs X.java
java-introcs X
```
Typical shell command sequence to invoke Java compiler and runtime

```plaintext
% javac-introcs Hello.java
% java-introcs Hello Bob
How are you ← invisible ctrl-D (ctrl-Z in Windows)
Hello Bob How are you

% java-introcs Hello Bob > example
how are things
%
% more example
Hello Bob how are things

% java-introcs Hello Bob < example
Hello Bob Hello Bob how are things

% java Hello Bob
Exception in thread "main"
java.lang.NoClassDefFoundError: StdIn
    at Hello.main(Hello.java:5)
    ...
```

---

**Hello.java**

```java
class Hello {
    public static void main(String[] args) {
        String s = StdIn.readLine();
        StdOut.print("Hello ");
        StdOut.print(args[0]);
        StdOut.println(" " + s);
    }
}
```
A rabbit hole for the shell

**bash is a full programming language**
- Manipulate files and folders. ✔
- Compile, run and interpret programs. ✔
- Search and sort within files.
- Manage users and access control
- Process management.
- Provide system information.
- if, for, while ...
- Type `man X` to learn about command X.

**Explore bash when you have the time**
- Used heavily in later CS courses.
- Used widely by programmers for decades.
- Can be fun! ← Try `man say` on Mac OS X
Another rabbit hole for the shell

**jshell is a Java interpreter**

- Commands are snippets of Java code.
- Extensive language support.
- Worthwhile for answering simple questions about language features.

```
% jshell
| Welcome to JShell -- Version 11.0.2 |
| For an introduction type: /help intro |

jshell> 2+2
$1 ==> 4

jshell> 1.0/2.0*3
$2 ==> 1.5

jshell> 1/2/3
$4 ==> 0

jshell> int sum = 0;
sum ==> 0

jshell> for (int i = 0; i < 10; i++)
   sum += i;

jshell> sum
sum ==> 45
```

Bottom line. Try it!  
also can use `jshell-introcs` with standard libraries
Controlling your computer with a keyboard is useful (and fun)

- Shell in Terminal app: *Control your computer with a keyboard*.
- Many argue that this is the *most efficient* way to program.
- Essential features in widespread use for over 50 years.
- Full capabilities built into modern IDEs.

What you need to know (this week)

- `ls`, `more`, `cd`, `mkdir`, `pwd`
- `java`, `javac`, `jshell`
- `-introcs` versions
- redirection for file I/O
N-body Assignment
A "data-driven" program that

- Reads a description of a universe from standard input
- Simulates motion as per command-line arguments.
- Prints final positions on standard output

```sh
> more planets.txt
5
2.50e+11
 1.4960e+11 0.0000e+00 0.0000e+00 2.9800e+04 5.9740e+24 earth.gif
 2.2790e+11 0.0000e+00 0.0000e+00 2.4100e+04 6.4190e+23 mars.gif
 5.7900e+10 0.0000e+00 0.0000e+00 4.7900e+04 3.3020e+23 mercury.gif
 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 1.9890e+30 sun.gif
 1.0820e+11 0.0000e+00 0.0000e+00 3.5000e+04 4.8690e+24 venus.gif

>java -introcs NBody 31557600.0 25000.0 < planets.txt
5
2.50e+11
 1.4959e+11 -1.6531e+09 3.2949e+02 2.9798e+04 5.9740e+24 earth.gif
 -2.2153e+11 -4.9263e+10 5.1805e+03 -2.3640e+04 6.4190e+23 mars.gif
 3.4771e+10 4.5752e+10 -3.8269e+04 2.9415e+04 3.3020e+23 mercury.gif
 5.9426e+05 6.2357e+06 -5.8569e-02 1.6285e-01 1.9890e+30 sun.gif
 -7.3731e+10 -7.9391e+10 2.5433e+04 -2.3973e+04 4.8690e+24 venus.gif
```

**Note.** This program is challenging (but we provide help).
General advice for assignments

- Get an early start
- Follow checklists
- Study assignment
- Use incremental approach
- DO NOT BINGE
Program incrementally

**Decompose the problem into a step-by-step process**

- Parse command-line arguments.
- Read universe from standard input.
- Draw universe on standard drawing.
- Simulate motion of bodies in the universe.
- Print universe on standard output.

**Do the steps in order of difficulty, debugging along the way**

- Start with comments (no code).
- Parse arguments, read universe, print universe.
- Submit and "Check Submitted Files"
- Draw universe (and play music)
- Simulate motion of bodies in universe
Start with comments and no code

**NBody.java**

```java
public class NBody {
    public static void main(String[] args) {
        // Step 1. Parse command-line arguments.
        // Step 2. Read universe from standard input.
        // Step 3. Initialize standard drawing.
        // Step 5. Simulate the universe.
        // Step 6. Print universe to standard output.
    }
}
```

**Next.** Follow the checklist, one step at a time, referring as needed to the assignment.
What could go wrong?

% java-introcs NBody 157788000.0 25000.0 > planets.txt
<ctrl-C>

% java-introcs NBody 157788000.0 25000.0 < planets.txt
Exception in thread "main" java.util.NoSuchElementException
  at java.util.Scanner.throwFor(Scanner.java:907)
  at java.util.Scanner.next(Scanner.java:1530)
  at java.util.Scanner.nextInt(Scanner.java:2160)
  at java.util.Scanner.nextInt(Scanner.java:2119)
  at StdIn.readInt(StdIn.java:319)
  at NBody.main(NBody.java:54)

% more planets.txt

It's empty—you overwrote it with an empty file

Ooops, meant to go the other way
What could go wrong?

no motion

no gravity
What could go wrong?

- no double buffering
- gravity sign error
What could go wrong?

- wrong force loop
- cut-and-paste error (x vs. y)
Opportunities for the bored

planetsparty.txt
(created by Mary Fan)

twinbinaries.txt
(David Costanzo)
Opportunities for the bored II

chaosblossum.txt
(Erik Keselica)

galaxy.txt
(Matt Tilghman)
Too much information

Sources of information for this assignment

- Lecture.
- Textbook.
- Assignment.
- Checklist.
- This meeting.
- Sample programs from precept.
- Advice from lab, preceptors, peers.
- Web search.
- Personal experiments/experience.
- Help tab on course website

General goal. Develop an ability to critically evaluate which information will be helpful to you.
COS 126 Programming Exam

• Overview
• Example programming exam
Programming exams

COS 126 Programming exams.

- **February 27** and April 23.
- Mini in-class assignments, written by RS.
- “Have you been participating in precept?”
- “Can you write a short program?”
- Prepare by practicing with old exams and exercises in book/booksite.
- Practice programming exam on Feb 25

February 2020

<table>
<thead>
<tr>
<th>Su</th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td><strong>8</strong></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td><strong>19</strong></td>
<td>20</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td><strong>25</strong></td>
<td>26</td>
<td><strong>27</strong></td>
<td>28</td>
<td>29</td>
</tr>
</tbody>
</table>

Policies (official details in email from RS).

- Covers Lectures 1 through 6
- Open course materials.
- No other web access.
- No outside communication.
Programming Exam Logistics

Writing a short program in 50 minutes can be a challenge for anyone.

• You will use your own computer.
• You will download and edit a template.
• You will submit your solution in the same way as you do for assignments.

You don't all fit in this room.

• Pay attention to RS email.
• Know where to go and arrive early.
• Make sure your computer is charged.

Advice: Practice, practice, practice.

• Write some short programs on your own.
• Attend the practice programming exam.
• Try a past programming exam (untimed).
• Try another one (timed).

warning: some old ones are more difficult than yours will be
Q. Can you write a simple program on your own?

Example (Fall 2015).

Part 1. Write programs that find the number of distinct values among the integers on standard input, assuming that the input is nonempty and in sorted order.

Your task. Add code to the file Count1.java to print the number of integers on standard input and the number of distinct values among those integers.

Details. Write a single loop that uses StdIn.readInt() to read each integer one at a time, but do not save them in an array. To compute the number of distinct values, add code to the loop to update distinct if the new value just read differs from the value read just before it.
Your task. Add code to print the number of integers on standard input and the number of distinct values among those integers.

```java
public class Count1 {
    public static void main(String[] args) {
        int count = 1;  // number of integers
        int distinct = 1;  // number of different ones
        int val = StdIn.readInt();
        while (!StdIn.isEmpty()) {
            int newVal = StdIn.readInt();
            count++;
            // count the integers
            if (newVal != val) {
                distinct++;
                // count the different ones
                val = newVal;
            }
        }
        StdOut.println(distinct + " distinct values among " + count + " integers");
    }
}
```

% more testCount1tiny.txt
1 1 1 1 2 2 2 2 4 4 4 4 4 5 5 6 6 9 9
% java Count1 < testCount1tiny.txt
1 distinct values among 18 integers

% java Count1 < testCount1tiny.txt
6 distinct values among 18 integers