Programming Languages and Techniques (CIS1200)

Lecture 32

Histogram Demo

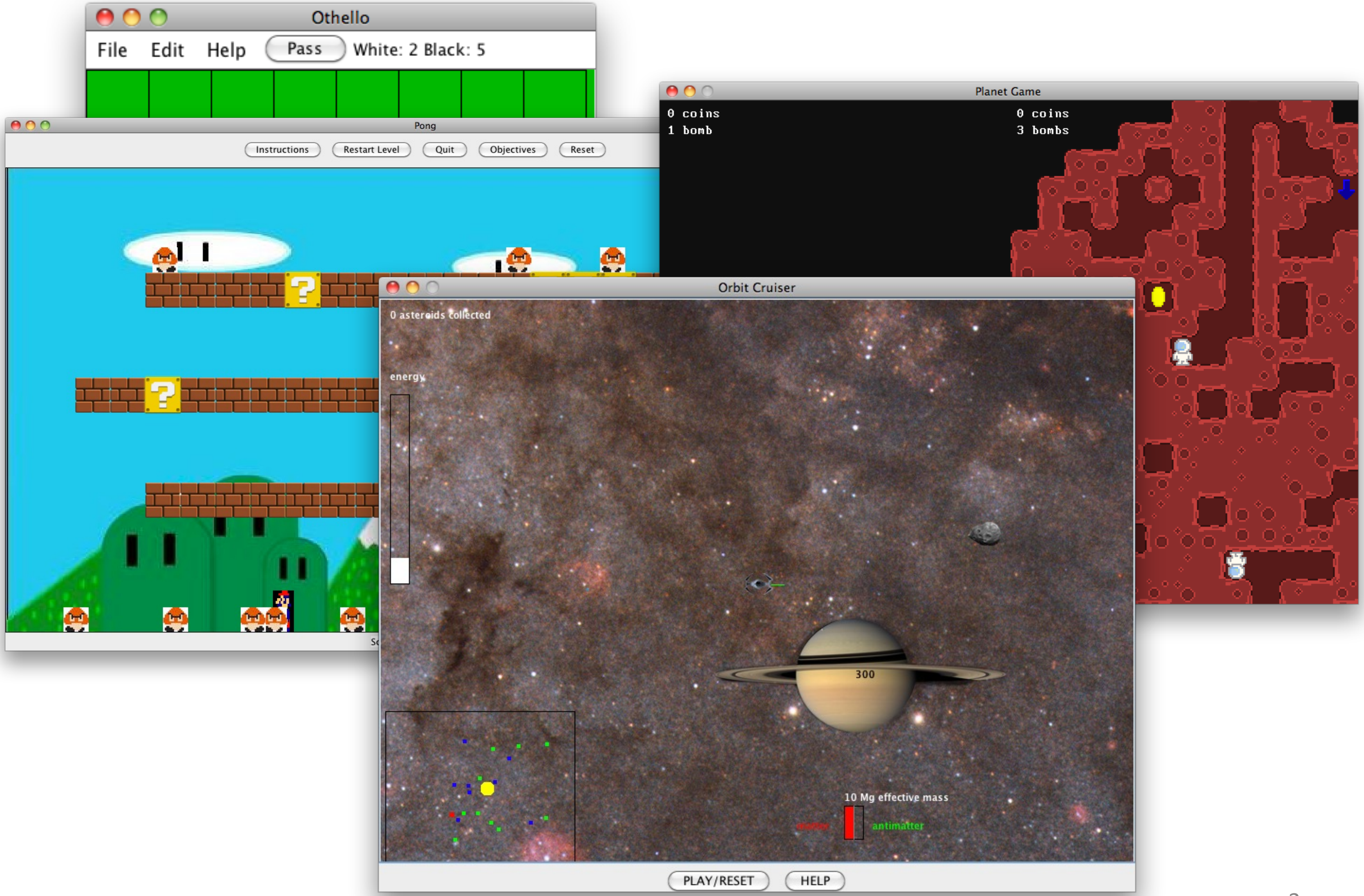
Chapter 28

Announcements (1)

- Midterm 2
  - Grades and solutions will be posted by Friday
- HW08: TwitterBot*
  - Due on November 26th
  - Practice with I/O and Collections

* Maybe should be called "XBot" or "TheProjectFormerlyKnownAsTwitterBot"?

HW9: Game Project



HW9: Game project

- Game Design Proposal Milestone Due: (8 points)
  Thursday, November 21st at Midnight = 11:59PM!
  - (Should take about 1 hour)
  - Submit on GRADESCOPE
  - TAs will give you feedback soon
- Final Program Due: (92 points)
  Monday, December 9th at 11:59pm
  - Submit zipfile online, submission *only* checks if your code compiles
  - IntelliJ is **strongly recommended** for this project
  - You may distribute your game (after the deadline) if you do not use any of our code
- Grade based on demo with your TA during/after reading days
  - Grading rubric on the assignment website
  - Recommendation: don't be too ambitious.
- *NO LATE SUBMISSIONS PERMITTED*

Announcements (3)

- Plans for the week of Thanksgiving
  - HW08 due on Tuesday at 11.59pm
  - No recitations that week
  - TA OH till Tuesday will be virtual
  - No OH from Wednesday to Sunday

  - Wednesday, November 27th – Bonus Lecture
    - Material is not needed for HW or Exams
    - Should be fun!
    - (will be recorded)
  - No lecture on Friday

Announcements (4)

- TA position applications are available
  - CIS 1100, 1200, 1600, 1210 (see https://tinyurl.com/2tn2t22f)
  - Other CIS and NETS classes (see https://www.cis.upenn.edu/ta-information/)
  - Accepting applications until Friday, November 22nd
  - Intro CIS TA Panel
    - Recording should be available

Design Example: Histogram.java

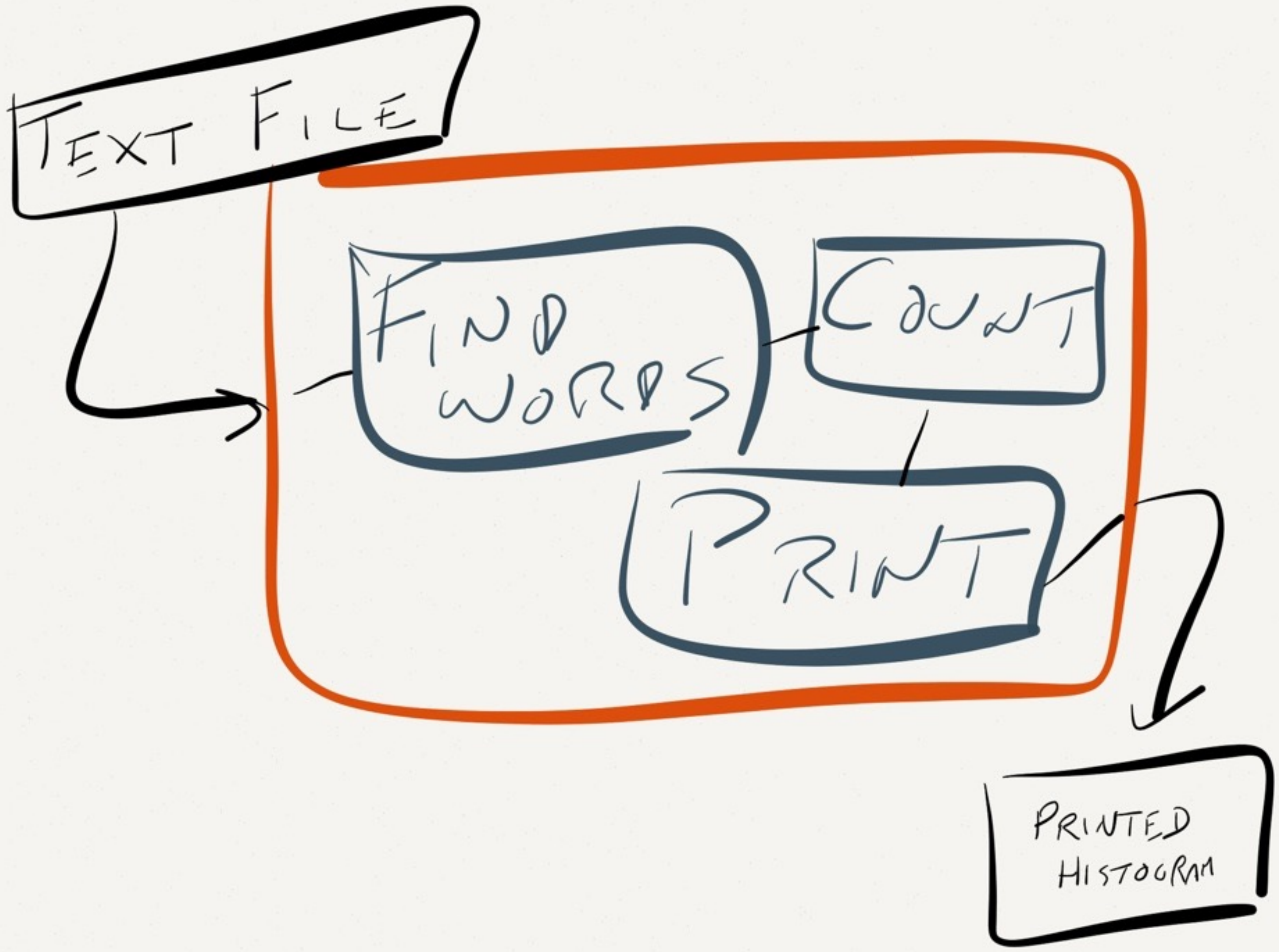
A design exercise using java.io and the generic collection libraries

(SEE COURSE NOTES FOR THE FULL STORY)

Problem Statement

Write a program that, given a filename for a text file as input, calculates the frequencies (*i.e.*, number of occurrences) of each distinct word of the file. The program should then print the frequency distribution to the console as a sequence of “word: freq” pairs (one per line).

```
Histogram result:
The : 1              each : 1            line : 2             should : 1
Write : 1            file : 2            number : 1           text : 1
a : 4                filename : 1        occurrences : 1      that : 1
as : 2               for : 1             of : 4               the : 4
calculates : 1       freq : 1            one : 1              then : 1
command : 1          frequencies : 1     pairs : 1            to : 1
console : 1          frequency : 1       per : 1              word : 2
distinct : 1         given : 1           print : 1
distribution : 1     i : 1               program : 2
e : 1                input : 1           sequence : 1
```


TEXT FILE
FIND WORDS
COUNT
PRINT
PRINTED HISTOGRAM

Decompose the problem

- Sub-problems:
    1. How do we iterate through the text file, identifying all of the words?
    2. Once we can produce a stream of words, how do we calculate their frequency?
    3. Once we have calculated the frequencies, how do we print out the result?

- What is the interface between these components?
- Can we test them individually?

How to produce a stream of words?

1. How do we iterate through the text file, identifying all of the words?

```java
public interface Iterator<T> {
   // returns true if the iteration has more elements
   public boolean hasNext();
   // returns the next element in the iteration
   public T next();
   // Optional: removes last element returned
   public void remove();
}
```

- **Key idea:** Define a class (WordScanner) that implements this interface by reading words from a text file.

Coding: Histogram.java

WordScanner.java

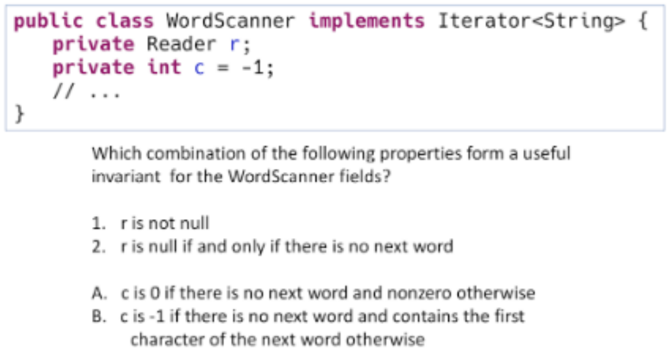
Histogram.java

Iterator – hasNext() – First Attempt?

```java
@Override
public boolean hasNext() {
    boolean value = true;
    try {
        int c = r.read();
        if (c == -1) {
            value = false;
        }
    } catch (IOException io) {
        System.out.println("IO Exception happened");
    }
    return value;
}
```

32: Which combination of the following properties form a useful invariant for the WordScanner fields?

✓ 0



```java
public class WordScanner implements Iterator<String> {
    private Reader r;
    private int c = -1;
    // ...
}
```

Which combination of the following properties form a useful invariant for the WordScanner fields?

1. r is not null
2. r is null if and only if there is no next word

A. c is 0 if there is no next word and nonzero otherwise
B. c is -1 if there is no next word and contains the first character of the next word otherwise

1 & A — 0%

1 & B — 0%

2 & A — 0%

2 & B — 0%

```
public class WordScanner implements Iterator<String> {
	private Reader r;
	private int c = -1;
	// …
}
```

Which combination of the following properties form a useful invariant for the WordScanner fields?

1. r is not null
2. r is null if and only if there is no next word

A. c is 0 if there is no next word and nonzero otherwise
B. c is -1 if there is no next word and contains the first character of the next word otherwise

ANSWER: 1 & B