Objectives

- Project: Do-over Preliminary Implementation Demo
- Design discussion – Assignment 10, Assignment 11

PRELIMINARY IMPLEMENTATION – DO OVER

Plan Through Dec 18?

- Final Implementation
  - Team deadline?
  - Decision on 3 extensions?
- "Post-mortem Analysis" – Due Dec 18, 5 p.m.
  - Overview
  - Planning
  - Status/Details
  - Conclusions
  - Collaboration
  - Future Work

ASSIGNMENT 10 DISCUSSION

Observations

- Many similar critiques, solutions
  - Lack of comments
  - Long methods
    - Extract method
  - Difficult to test!
    - Easier with extracted methods
- Many variations on designs
  - Even though a small project/assignment, there are lots of design decisions!

Excerpts from Good Critiques

- The majority of the bin-fitting process was handled inside the main method. This probably made the code easy to write, but is disadvantageous for a number of reasons:
  - Readability: …
  - Maintainability: …
  - Testing: *unit testing does not break down into small pieces to test.* There is just one big main method
  - Debugging: …

Excerpts from Good Critiques

```
public Disk(int id) {
      this();
      myId = id;
}
```

→

```
public Disk() {
      myId = idCount++;
}
```

- Added a static field "ID" to track the ID of a disk rather than wasting the extra code lines of having an extra constructor to specify the ID and forcing [others to] track the IDs of the disks it is creating…

What are the tradeoffs to this approach?

Excerpts from Good Critiques

```
public Disk(int id) {
      this();
      myId = id;
}
```

→

```
public Disk() {
      myId = idCount++;
}
```

- Added a static field "ID" to track the ID of a disk rather than wasting the extra code lines of having an extra constructor to specify the ID and forcing [others to] track the IDs of the disks it is creating…
- The downside of this approach is that we can't directly specify what we want the ID of a disk to be. On the other hand, it is a much more direct and efficient way to ensure that we are always getting a *unique set* of IDs for a set of disks.

Excerpts from Good Critiques

- One of the cons of [my refactored] solution I can see is that the generateResults() method, [describes issue…]

```
public static String generateResults() {
   System.out.println("worst-fit decreasing method");
   System.out.println("number of pq used: " + pq.size());
   while (!pq.isEmpty()) {
       System.out.println(pq.poll());
   }
   System.out.println();
}
```

What is the issue? Why is it a problem?

Excerpts from Good Critiques

- One of the cons of [my refactored] solution I can see is that the generateResults() method, [describes issue…]

```
public static String generateResults() {
   System.out.println("worst-fit decreasing method");
   System.out.println("number of pq used: " + pq.size());
   while (!pq.isEmpty()) {
       System.out.println(pq.poll());
   }
   System.out.println();
}
```

Unexpected side effect of method
Symptom of a poorly designed API

Excerpts from Good Code Critiques

- I chose to make Bins a separate class only responsible *for* adding files and creating disks. This makes the code more extensible for future use…
- Bins was trying to do too much with reading from a file so I moved this to the BinsRunner files since the important part about Bins is not how it gets the data, but what it does once it has the data.

Excerpts from Good Code Critiques

- I chose to make Bins a separate class *only responsible for* adding files and creating disks. This makes the code more *extensible* for future use…
- Bins was trying to do too much with reading from a file so I moved this to the BinsRunner files *since the important part about Bins is not how it gets the data, but what it does once it has the data.*

Excerpts from Good Critiques

- I thought about how this program *is likely to change*. Right now we have two different methods to fit files onto disks; however, these two are certainly not the only two methods, and in the future *perhaps we will want to use other methods* in the Bins class. For this reason, I decided to make the fitFilesToDisk method abstract in the Bins class and to make a WorstFit class that inherits from the Bin class….

Excerpts from Good Code Critiques

- After looking back over the code and the changes I've made, I think there will *almost always be more changes possible*. For example, the code for the different heuristic types could be extracted to a separate class thats [sic] *only job* is to define the heuristics.
- Also, the Disk class could be changed to accommodate any type of storage media, not just DVDs.

Testing Conundrum

```
@Test
public void TestWorstFit(){
   List<Integer> results = Bins.readData("data/
example.txt");
   Method t = Bins.worstFit(results, "test fill");
   assertEquals(t.getName(), "test fill");
   assertEquals(t.getTotal(), 1950000);
   PriorityQueue<Disk> pq = t.getPq();
   assertEquals(pq.poll().toString(), "2\t850000:\t
150000");
   assertEquals(pq.poll().toString(), "0\t100000:\t
700000 200000");
   assertEquals(pq.poll().toString(), "1\t100000:\t
800000 100000");
}
```

What is an issue in this code?

Testing Conundrum

Rule of Thumb: when you're having trouble testing, refactor to make it easier to test.

```
@Test
public void TestW
   List<Integer> results = Bins.readData("data/
example.txt");
   Method t = Bins.worstFit(results, "test fill");
   assertEquals(t.getName(), "test fill");
   assertEquals(t.getTotal(), 1950000);
   PriorityQueue<Disk> pq = t.getPq();
   assertEquals(pq.poll().toString(), "2\t850000:\t
150000");
   assertEquals(pq.poll().toString(), "0\t100000:\t
700000 200000");
   assertEquals(pq.poll().toString(), "1\t100000:\t
800000 100000");
}
```

Problem: Difficult to test, relies on formatted String
Fix: Add better `equals` method for `Disk`

Comparing APIs

```
/** @param arg file name
 * @return String list of file sizes from text file
 */
public static List<Integer> readData(String arg)
```

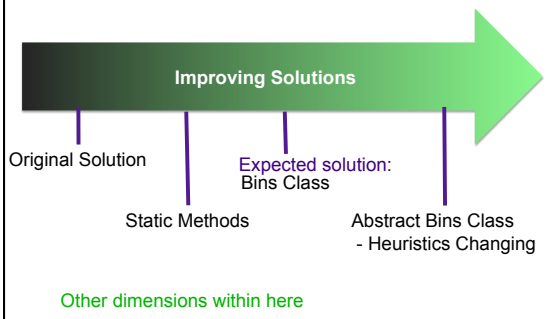
```
/** @param arg Scanner that reads data from a text file
 * @return String list of file sizes from text file
 */
public static List<Integer> readData(Scanner arg)
```

- Which API would you prefer to use as a user?

Static, Static, Static?

- Some of your refactored classes had all static methods
- What are the tradeoffs of having a class with all static methods versus creating a class that can be instantiated?

Reviewing Refactoring: Bins



Other dimensions within here

Course Evaluations

- On Sakai
  - Anonymous → I'll see a submission number
  - At the end, it says "Submit for Grading", but you won't be graded
  - Won't be viewed until after grades submitted
- Let me know if anything doesn't work, and we'll switch to paper
- Two evaluations:
  - "Course Evaluation"
  - "Supplemental Evaluation"—Specific to this course and improving for next time
- "Rationale" box is for comments related to the question
- Incentive: all four complete both surveys: 5% off the total points possible for the assignments grade