Slide 1

Administrivia

- Reminder: Homework 1 code due today (11:59pm).
- Reminder: All homework is considered pledged work. Write “pledged” on hardcopy work, and include it in comments for programming assignments. (No problem if you’ve already turned in Homework 1 without doing this, but please remember for future assignments.)
- Quiz 1 Thursday. About 10 minutes long, at the end of class. Open book, open notes. Should be no real need to study if you have kept up with reading and material presented in class. Likely kinds of questions are “what does this code do?” or “write some code to do X”.
- Homework 2 design due Thursday. A little more about the assignment today.

Slide 2

Where Were We? (Recap)

- Topics: Java/OO basics, a little about generics.
- Project: Initial design, first (small) program.

Slide 3

A Little More About Generics

- "Generics" are a relatively new feature of Java, and can feel a little complicated to those whose first exposure to the language didn't include them. We talk about them now because they're so useful, and because you need them for the project.
- Let's look at a couple of simple examples . . .

Slide 4

Homework 2 — General Comments

- Design phase is meant to be about defining classes and interfaces. For every class (or interface) and every method, I want comments (can be be brief). For classes, these should describe (to the best of your understanding) how they fit into your game (e.g., "class for wall blocks").
- In order to generate the HTML documentation ("javadoc"), probably have to have something minimally compilable. As suggested in assignment — create skeleton/stub versions of methods, and fill in real code in code phase.
- Be sure to get the updated JAR file (should have name `PAD2F07Assn2.jar`). With every assignment there will be a new JAR file, as you replace various parts of the starter code with your code.
- Method `instance` in `BasicGameSetup` mentions "singleton". What's that about? Reference to "singleton design pattern" — idea that for some classes there should only ever be one instance.

Slide 5

Homework 2 — Design

- Interfaces `YourBlock`, `YourEntity`: In project API, referred to as "general block type" and "general entity type". You will use these as replacements for `BasicBlock` and `BasicEntity`, and everywhere else you use one of the framework's generic classes.
- Player and game setup classes. Copy code from `BasicPlayer` and `BasicGameSetup` and edit (change `package` line, block and entity types). May want to change game setup more during code phase. Also edit your main class from the first assignment.

  Don't worry about player for now — you will start writing your own in the next assignment.

Slide 6

Homework 2 — Design Continued

- Block class(es). These are blocks that make the playing field for your game. Should have one class for each kind of block (floor, walls, ladders, anything that doesn't move). Try to define as many as you can. Copy code from `BasicBlock`.
- Screen class (class implementing `Screen` interface). This is the most work in this assignment. Eclipse can make stub methods for you. Copy and paste comments from API.

Slide 7

How to Approach Defining a Class

- What methods do I need? If implementing an interface, you at least need the methods in the interface. May want additional methods. If making a subclass, remember you automatically inherit all methods from superclass. Can override them and/or provide additional methods.
- What variables do I need to implement the needed methods? e.g., if defining a `Rectangle` class that has a `getArea` method, probably need either area or width and height.

Slide 8

A Little About Arrays in Java

- Arrays are objects — unlike in C/C++, where they're basically pointers.
- Declaring (references to) arrays — denote by putting brackets after type.
- Creating arrays — use `new`, e.g.,

  ```
  new int[10]
  new String[n]
  ```

  (Remember that the second one only creates *references*.)
- All arrays have `length` variable.
- Otherwise, syntax is same as C/C++; indices start at 0.
- Java runtime does automatic bounds-checking — unlike in C/C++, get ArrayBoundsException rather than random problems.

Slide 9

Multidimensional Arrays

- "Arrays of arrays", e.g.,

  ```
  int[][] x = new int[10][100];
  ```

  declares an array of 10 arrays of 100 `int`s.
- Reference elements with row, column indices, e.g.,

  ```
  x[row][col] = 10;
  ```
- Both dimensions accessible:

  `x.length` = ?

  `x[0].length` = ?
- Note that order of indices (row then column) is the opposite of the "graphics convention" used in the game.

Slide 10

Minute Essay

- Write code to define an array of four `String`s and fill it with data of your choice.
- Write code to define a two-by-three array of `int` and set each element to the sum of its row and column.
- If I declare an array of `MyClass` references:

  ```
  MyClass[] objs = new MyClass[10];
  ```

  do all the elements of `objs` have to be instances of `MyClass`, or can they be instances of some other class?

Slide 11

Minute Essay Answer

- One solution (array of `String`s):

```
String[] s = new String[4];
s[0] = "hello";
/* other three lines similar */
```

- One solution (array of `int`s):

```
int[][] a = new int[2][3];
for (int row = 0; row < a.length; ++row)
    for (int col = 0; col < a[0].length; ++col)
        a[row][col] = row + col;
```

- Elements of an array declared as `MyClass[]` can be instances of any "subtype" of `MyClass` — `MyClass` itself, or any subclasses. (Trick question!)