

Slide 1

Administrivia

- Reminder: Homework 2 design due today.
- Homework 2 code due next Tuesday (deadline extended).

Slide 2

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 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 `PAD2F06Assn2.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.

Homework 2 — Design

Slide 3

- 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.

Homework 2 — Design Continued

Slide 4

- 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 5

String Class — Example of Using a Class

- In C, “strings” are just arrays of characters, terminated by null character. In Java, there’s a library class, `String`.
- To see what’s available, look at the API . . .

Slide 6

String Class, Continued

- In general, no operator overloading in Java, with one exception — “+” for strings. Non-string objects converted using (their) `toString` method. Primitives converted in the “obvious” way.
- To compare two strings, “==” is rarely what you want. Instead, use `equals`.
- Strings are “immutable” — once created, can’t be changed. (Why? allows them to be safely shared.) Methods you would think might change the value return a new string.
- Use `StringBuffer` if you need something you can change, or for efficiency.
- Let’s do some examples . . .

Defining a Class

Slide 7

- 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.

Minute Essay

Slide 8

- None — quiz.