Today

• Course evaluations
• Final exam questions
• Classes

Next time: have course evaluations completed before class
Short Questions?
Quiz
Final Exam

• Comprehensive
• Same format as the last three exams
Data Associated with Classes: Instance Variables

Instance Variables: the data that make up a single instance of a class

• Each instance has its own variables
• Must have created an instance of the class (with “new”) for these variables to exist
• Instances variables are either:
  • private: access only by non-static class methods
  • public: generally accessible
Accessing Instance Variables

```java
public int count;
private String name;
```

- **Access from non-static methods:** can refer directly to these variables by name:
  ```java
count = 5;
name = “Bob”;  
```

  OR: `this.count = 5;` OR `reference.count = 5;`

- **Access from static methods or from outside the class:**
  ```java
  reference.count = 5;
  reference.name = “Bob”;  // NOT ALLOWED
  ```
Data Associated with Classes: Class Variables

Class variables: the data associated with the entire class
• All instances share these variables
• In fact, we don’t even need an instance to access these variables
Accessing Class Variables

```java
public static int numObjects = 0;
private static String baseName;

• Inside the class (static and non-static methods): can refer just to the variable name
  numObjects++;
  baseName = "Foo";

  Or: classname.numObjects++;

• Outside the class:
  System.out.println(classname.numObjects);
  System.out.println(classname.baseName);  // NO
```
Class Methods

Methods (either static or non-static) can be declared as public or private

• public: any other method can call the method
• private: can only be called by other methods in the class

• Also: non-static methods can only be called through an object instance
Accessing Class Variables

public void addTransition (int event, State, nextState)

• Inside the class (static and non-static method): can refer just to the method by name
  addTransition(8, state);
      or
  reference.addTransition(8, state);
      or
  this.addTransition(8, state);

• Outside the class:
  reference.addTransition(8, state);
Classes as Encapsulation/Aggregation

- Primitive types capture simple, small ideas
- Classes allow us to capture much larger ideas, possibly composed of many primitive values

Points, triangles, meshes ...
Classes as Contracts

When we are writing code for others (or for larger projects), we want to be able to guarantee that the code we provide adheres to certain, agreed-upon rules

• Publically declared instance variables allows outsiders to make arbitrary changes to the variables, possibly violating these rules
• Declaring instance variables as private disallows this. Outsiders are then left to access the instance variables indirectly through public methods
Wrap Up

Due:
• HW 11: due Thursday
• Project code reviews

Next time:
• Final: Tuesday, December 9, 4:30-6:30