Object-Oriented Programming (OOP)

- Approach to improving software
  - View software as a collection of objects (entities)

- Motivated by software engineering concerns
  - To be discussed later in the semester
Techniques – Abstraction

Abstraction
- Provide high-level model of activity or data

Procedural abstraction
- Specify what actions should be performed
- Hide algorithms

Data abstraction
- Specify data objects for problem
- Hide representation
Techniques – Encapsulation

Encapsulation

- Confine information so it is only visible / accessible through an associated external interface

Approach

- For some entity X in program
  - Abstract data in X
  - Abstract actions on data in X
  - Collect data & actions on X in same location
  - Protects and hides X

Extension of abstraction
Abstraction & Encapsulation Example

Abstraction of a Roster

- **Data**
  - List of student names

- **Actions**
  - Create roster
  - Add student
  - Remove student
  - Print roster

Encapsulation

- Only these actions can access names in roster

<table>
<thead>
<tr>
<th>ROSTER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>List of names</td>
<td></td>
</tr>
<tr>
<td>create()</td>
<td></td>
</tr>
<tr>
<td>addStudent()</td>
<td></td>
</tr>
<tr>
<td>removeStudent()</td>
<td></td>
</tr>
<tr>
<td>print()</td>
<td></td>
</tr>
</tbody>
</table>
Java Programming Language

Language constructs designed to support OOP

- **Example**
  - Interface – specifies a contract
  - Class – implements/defines contracts, supports encapsulation of implementation

Class libraries designed using OOP principles

- **Example**
  - Java Collections Framework
  - Java Swing
Java Interface

- An Interface defines a contract
  - Collection of
    - Abstract methods; no implementations
    - Constants
  - Can not be instantiated
- Classes can implement interfaces
  - Must implement all methods in interface
- Example
  ```java
class Foo implements Bar { … }
```
Java Collections Framework

- **Collection**
  - Object that groups multiple elements into one unit
  - Example: ArrayList, Stack

- **Collection framework** consists of
  - Interfaces
    - Abstract data type
  - Implementations
    - Reusable data structures
  - Algorithms
    - Reusable functionality

- **Collection – Java Interface is the Root for everything!**
  - See Java API entry for Collection

- **EXAMPLE: CollectionExample.java**
Project #1

Let’s go over the check out process and the submit server information