Abstraction

- General concept: describing an entity in terms of its aspects (qualities or features)
- Modeling entities in software
- Only essential aspects should be captured
  - Attributes
  - Behavior
- What is essential depends on situation

Example Abstraction

A “Passenger” Abstraction

- Abstraction of a passenger for a flight reservation system

Attributes:  
Behaviors:
Properties of a Good Abstraction

- well named – clearly identifies abstraction
- coherent – sensible description
- accurate – only attributes of entity
- minimal – no irrelevant attributes
- complete – everything needed

Mapping Abstraction to Software

<table>
<thead>
<tr>
<th>real-world</th>
<th>abstraction</th>
<th>software</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>attributes</td>
<td>{data, data, …}</td>
</tr>
<tr>
<td></td>
<td>behavior</td>
<td>{method, method, …}</td>
</tr>
</tbody>
</table>

Mapping Abstraction to a Class

```
className
public
private
{data, data, …}
{method, method, …}
```

Separation

- Separation of *what* a component does from *how* it does it (ex. a procedure)
- Define classes by independently specifying the interface for objects in that class, and the implementations of that interface

Separation of Interface and Implementation

```
visible
Interface

hidden
Implementation
```

Interchangeability

```
Interface

Implementation 1
Implementation 2
```
Separation of Classes

- Class → Interface
  - Provides method

- Interface
  - Identifies available methods

- Class
  - Uses method