CMSC 131
Object-Oriented Programming I

Introduction to Classes I

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Objects

- Bundles of (related)
  - data ("state")
  - operations ("behavior")
- Data often referred to as instance variables
- Operations usually called methods
- Invoking operations can change state (values stored in instance variables)
- We can visualize a method call as an object sending a message to another object
- Example of objects
  - Bank Account
  - Student
  - Scanner
- Object-Oriented Programming
  - Program is a collection of interacting objects
## Sample (Student Class)

<table>
<thead>
<tr>
<th>State</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>getAge</td>
</tr>
<tr>
<td>ID</td>
<td>date → age</td>
</tr>
<tr>
<td>DOB</td>
<td>getGrades</td>
</tr>
<tr>
<td>Major</td>
<td>sem., class → grades</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
</tr>
</tbody>
</table>
**Sample (Student **Object** )**

<table>
<thead>
<tr>
<th>State</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>getAge</td>
</tr>
<tr>
<td>ID</td>
<td>date → age</td>
</tr>
<tr>
<td>DOB</td>
<td>getGrades</td>
</tr>
<tr>
<td>Major</td>
<td>sem., class → grades</td>
</tr>
</tbody>
</table>

| etc. | etc. |

- **Name:** Kerry Keenan
- **ID:** 555*********
- **DOB:** 06-22-1987
- **Major:** CMSC
Classes

- Class → Blueprint/"Recipe" for objects
- Classes include specifications of
  - Instance variables (including types, etc.) to include in objects
  - Implementations of methods to include in objects
- Classes can include other information also, as will be seen later on
  - Static methods / instance variables
  - public / private methods
  - And so on
Student Class Example

• Instance variables:
  
  String name  
  int id  
  int dateOfBirth  
  String major

• Methods
  
  getAge()  
  getGrades()  
  etc.

• The actual class implementation will include code for the methods
• This describes a blueprint for student objects
class Student {

    /* These are the instance variables */
    String name;
    int id;
    int dateOfBirth;
    String major;

    /* Instance methods */
    getAge() {
        // put code here
    }
    getGrades() {
        // put code here
    }

    Etc.
}
How Are Objects Created?

• In Java: using `new`
  Recall:
  ```java
  Scanner sc = new Scanner(System.in);
  ```

• Invoking `new`:
  • Creates an object in a memory area called the “heap”. Space is created for instance variables
  • Returns the address/reference where the object lives
  • If you lose the reference you cannot access the object
Accessing State/Methods

- If
  - **obj** is an object reference
  - **v** is an instance variable of the object
  - **m** is a method of the object
- Then
  - **obj.v** is how to access the data v in **obj**
  - **obj.m()** is how to invoke m in **obj**
- So
  - If you have already done String str = “Joe”
  - Then str is a String
    - str is an instance of a class
    - Methods of this object→ equals, compareTo, etc.
    - str.equals(), str.compareTo(), etc. invokes these methods on that object
Example

• Let’s define a class called **SuperHero** with
  • Instance variables name and strength
  • Get/Set methods
  • print method
  • Method to increase the power of a superhero
• When creating a project in Eclipse we have two choices:
  • Everything in one folder
  • src/bin folder
• Let’s define a driver class for our example
  • Eclipse allow us to generate code 😊
  • Source → Generate Getters and Setters