CS 1233 Object Oriented and Design

Chap 8 -- Classes and Objects

What we learned so far?

Procedural programming (verb oriented)

- Program = collection of functions with variables
- Tell program to do this
- Tell program to do that

```
void doThis(...){
}

int  doThat(...){
}

double anotherFunction(...){

}

void main(String[] args){

}
```

Procedural Programming

Large programs have thousands of variables and functions

- Hard to understand
- Hard to debug
- Not very well organized

Why Object Oriented?

- OOP philosophy: Software is a simulation of the real world.
- We know (approximately) how the real world works.
- Design software to model the real world.
- Modular design

```
public class Player{

}
```

```
public class Score{

}
```

```
public class Ennemy{

}
```

Object Oriented Programming (OOP)

OOP (noun-oriented)

- Programming paradigm based on data types.
- Identify objects that are part of the problem domain or solution.
- Identity: objects are distinguished from other objects (references).
- State: objects know things (instance variables).
- Behavior: objects do things (methods).

OOP Principles

1. Encapsulation
2. Inheritance
3. Abstraction
4. Polymorphism

1. Encapsulation

- Separate implementation from design specification.
- Class provides data representation and code for operations.
- Client uses data type as black box.
- API specifies contract between client and class.

2. Inheritance

- Parent / Child relationships
- Parent contains common attributes
- Child inherits these attributes

3. Abstraction

- hide attributes/methods to reduce complexity
- related to encapsulation
- specifies contract between client and class

4. Polymorphism

- Objects with the same functions but differ in implementation
- Ability to determine what method to run based on object types
- Related to abstraction

Class: structure of an object

```
public class ObjectName
{
  // List of attributes

  // List of functions (a.k.a methods)
}
```

Class Example (1)

```
public class House {

  // List of attributes
  public int doors;
  public int windows;
  public int rooms;
  public int bathrooms
  public double size;

  // List of functions
  public void describe(){

  }

}
```

Class Example (2)

```
public class Book {

  // List of attributes
  public int pages;
  public String title;
  public String author;


  // List of functions
  public int getNumberOfPages(){

  }

  public void printTitle(){

  }

}
```

Multiple Classes

```
// Game.java
public class Game {
  public Player [] players;
  public Score score;
  public Enemy [] enemies;

  public void play(){}
  public void pause(){}
}

// Player.java
public class Player {
}

// Score.java
public class Score {
}

//Enemy.java
public class Enemy {
}
```