

# Classes & Objects pt. 2

Ch. 7

# Private Accessor

- You may not want your instance variables accessed directly
- Using `private` restricts access to that class

```
public class Student{
```

```
    private int studentID;  
    private String name;
```

```
    . . .
```



Only accessed  
within Student

# Getter/Setter Methods

- For every private instance variable in your class, you need at least one “get” and one “set” method

```
// Add a method, inc(), to increment a Time value by 1
// second. min and sec should not reach 60 or above.
class Time {
    private int hour, min, sec;
    Time(int hour, int min, int sec) {
        this.hour = hour;
        this.min = min;
        this.sec = sec;
    }
    // your method goes here
```

```
// Write a class called Rational. Rational will consist
// of a numerator and a denominator – both fields are
// private. Rational will also have a method called times
// that will multiply one Rational number by another, and
// return the result as a Rational number. It is not
// necessary to reduce fractions to lowest terms.
```

# Bubble Sort

- Of all common sorting algorithms, this is probably the least efficient – but the easiest to implement.

# Writing to a file

```
import java.io.*;
public class WriteTest{
    public static void main (String[] args){
        FileWriter f;
        PrintWriter p;

        try {
            f = new FileWriter("output.txt");
            p = new PrintWriter(f);
            p.println("First Line");
            p.print("Second line, no carriage return ");
            p.printf("Shortened Pi: %.2f", Math.PI);
            p.close();
        }
        catch(IOException ex) {
            System.out.println(ex.toString() + " file error.");
        }
    }
}
```

# Diverting input/output stream

> writes to a specified output file. Overwrites if needed.

>> writes to a specified output file. Doesn't overwrite, only appends.

< reads input from a specified file.

Ex:

```
$ java HelloWorld > HelloWorld.txt
```



# Arrays of Objects

- Need to make sure you initialize objects in array, not just the array itself.
  - NullPointerException



# Static vs. Instance Methods

- Instance methods are called by individual objects
- Static methods are called by the class
- Calling a method from the same class only requires the method identifier...no qualifier such as object or class name
- Why use static methods? We don't always need an instance of a class in order to call a method – ex: Math class

# Static variables

- A static variable maintains the same value across all instances of a class.
  - As opposed to instance variables, in which each object has its own instance variable

# Calling Methods

- Instance to Instance 
- Instance to Static 
- Static to Instance 