CSC 207
Algorithms and Object-Oriented Design

Weiss, from the beginning of chapter 3 through section 3.2 and sections 3.4 through 3.7 (pages 69–73 and pages 76–90)

I am indebted to my colleague, John David Stone, for assistance with this handout.
This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit
https://creativecommons.org/licenses/by-sa/4.0/
or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Information Hiding and Encapsulation

• Encapsulation is a way of abstracting the internal details of an object from outside world.
• Encapsulation means grouping class — and — to form an aggregate.
• Information hiding means that object internal details can be accessed by —.
• To enforce information hiding, class members are placed in the public/private section.
Information Hiding and Encapsulation (cont.)

Figure 3.1

A complete declaration of an IntCell class

```
// IntCell class
// int read() --> Returns the stored value
// void write( int x ) --> x is stored

public class IntCell
{
    // Public methods
    public int read() { return storedValue; }
    public void write( int x ) { storedValue = x; }

    // Private internal data representation
    private int storedValue;
}
```

Figure 3.3

A simple test routine to show how IntCell objects are accessed

```
// Exercise the IntCell class

public class TestIntCell
{
    public static void main( String [ ] args )
    {
        IntCell m = new IntCell();
        m.write( 5 );
        System.out.println( "Cell contents: " + m.read() );
    }
}
```
public void finalTransfer(Account rhs) {
    if (this == rhs) return;
    dollars += rhs.dollars;
    rhs.dollars = 0;
}
Visibility Levels

• Public: accessible by any other class
• Protected: accessible by classes that inherit from or are in the same package as such classes
• Package: accessible by classes in the same package
• Private: accessible by code that is inside the class definition
equals

• The *equals* method checks whether two objects store the same value.
  – Check the values of fields for any two objects