Objects

□ An object in Java is
  – A set of *methods* (think: functions)
  – A set of *data* (think: variables)

□ Fancy CS buzzwords:
  – Objects *encapsulate data and functionality*
  – Objects *encapsulate behavior and state*
Objects: Concept

- The idea is that a program manipulates data via methods (functions)
  - In P6, you move around the maze using methods
  - So, too, was the names of the text files submitted by the user on the command line
  - You wrote a method (called main)
Object Example: String

- You have been using objects all along
- String is an example of an object in Java
  - The characters are the data in the object
  - Methods include:
    - length() : how long is the string?
    - charAt(int): what character is at a given position?
- Syntax:
  - You call an object’s method using ‘.’ and args ()
    - E.g.: word.charAt(5); word.length()
  - You access an object’s data using just ‘.’
Another example: Scanner

- Scanner is a more complex object
- Its data is a stream of characters
  - May come from a file
  - May come from the terminal (a stream)
  - May come from a string
- Its actions are to parse and interpret the characters
  - next() returns the next valid string
  - nextInt() returns the next valid integer
  - nextDouble() returns the next valid double
  - … and there are many more (see on-line Java reference)
Yet another example: arrays

- Arrays are the simplest objects
  - Their data is a set of variables of the type provided
  - Their data includes a variable that stores the length of the array
    - Note that array lengths are stored in variables, not computed by functions
    - This is why there are no ()’s after args, if args is an array.
Classes as data types

- Classes are data types (just like primitives):
  ```java
  int counter;
  String word;
  MyClass example;
  ```
- By convention, class names are capitalized
- Variables with object types still need names
  - E.g. counter, word, and example above
- Variables cannot be used until they are assigned values
  - True for both primitive and object types
Object Instances

- The value assigned to a variable of an object type is an object instance
  - Just like “Alice” is an instance of a String
- For example:
  ```java
  String word = new String("the");
  ```
  - `word` is a variable of type `String`.
  - `String("the")` creates an instance of `String`
- All Object instances are created using the keyword `new`. 
Methods inside a class

- Order of writing methods is arbitrary
  - Generally constructors are written first
- Shared data problem: what if two methods need to share data?
  - One subtask reads input and creates an array of words
  - Another subtask checks each word in the array and does something with it
Solution #1

- Method1 for subtask 1 returns a value, v
- Method2 for subtask 2 uses the value, v
- Example:

```java
public static void main(String[] args) {
    String[] wordList = readInput();
    processWords(wordList);
}
```
Solution #2

- Use instance variables
  - Define String[] wordList; as an instance variable
  - Any method of a class can access its variables
    - readInput() can create & write the array
    - processWords() can access it
Data Variables in Classes

- How does a method access data in a class?
  - Every method can access the class instance it is called on
    - Think of word.length(); it can access the data in the string ‘word’
    - Think of the class instance as a ‘hidden’ argument to the method
  - Class variables look like any other variables in the code of a method
    - They do not need to be ‘re-declared’
public class Course {
    private String department;
    private String number;
    private String[] sections = new String[2];
    public Course(String dept, String num) {
        department = dept;
        number = num;
    }
    public String getFullName(){
        return new String(department + " " + number);
    }
    public static void main(String[] args) {
        Course c1 = new Course("CS", "160");
        System.out.println(c1.getFullName());
    }
}