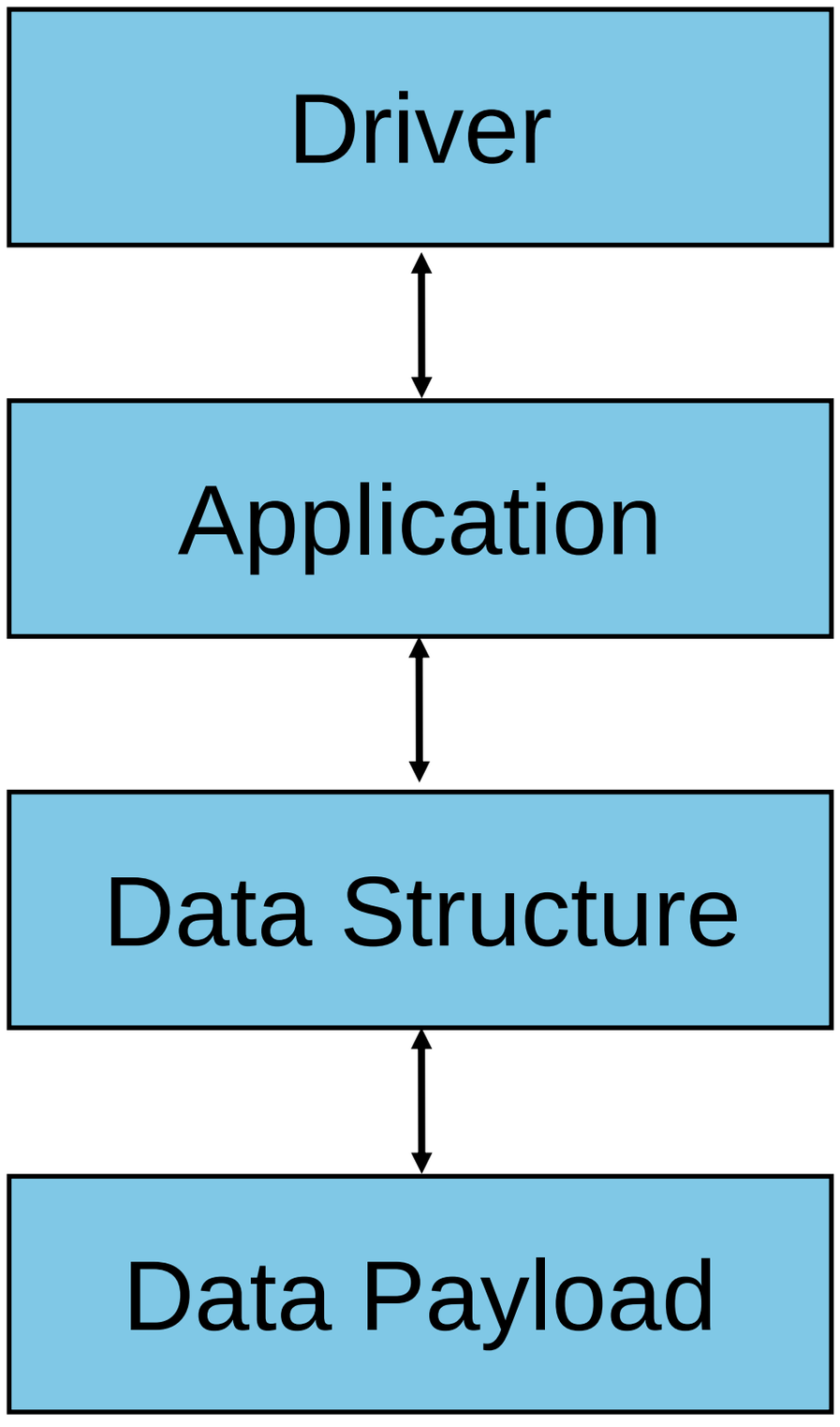
Chapter 2
18 January 2011

Objectives

- A refresher on Java
- Basic structure of a program
- Documentation requirements
- A brief introduction to UML

A complete program



Code that opens and closes files and invokes the application

This is the code that actually does what the program is supposed to do, such as manage a phone book

This is the class or classes that implement the data structure

This is the abstract data type (ADT) that handles an individual instance of a data record

The Data Payload – An ADT

- Instance variables – one copy for each instance of the object
- Static variables – one copy across all instances of the class
- Explicit labelling of variables as `public` or `private`

- Primitive data types
- Class types (like `Integer`) that wrap the primitive types
- Built-in classes (like `String`)
- User defined classes as data types

The Data Payload (2)

- Constructor method invoked when an instance is created
- Accessors (getters) and mutators (setters) for controlled access to instance variables
- We will use `Scanner` for input and `PrintWriter` for output

The Data Payload (3)

The `toString` method

- Returns a `String` value that is the formatted user-friendly display of the data
- Makes in unnecessary for invoking programs to know what's inside the ADT—they see only the formatted string
- Makes it unnecessary for the ADT to know to where to display the data (i.e., don't pass in a `PrintWriter` to which to display, just return a `String` for the invoking program to display)

The Data Payload (4)

The `readRecord` method

- Passed an opened `Scanner`, returns a `Record` value after having read in an instance
- Makes in unnecessary for invoking programs to know what's inside the ADT—they see only the returned value
- Encapsulates error checking, data munging (like lastname/firstname/Jr. testing) inside the ADT

The Data Payload (5)

The `compareTo` method

- Invoked as `thisRecord.compareTo(thatRecord)` just as with `String` data
- Makes it unnecessary for invoking programs to know *why* “this” is less than “that” (or vice versa)—they see only the returned value
- Permits elaborate comparisons and multiple-level comparisons (e.g., last name, first name, middle name)

The Main/Driver Class

- Opens and closes files, checks for proper open/close, etc.
- After this example, we will use a `FileUtils` class that has the testing written and tested once and that decreases clutter in the text of the main
- In this simple example, the main creates the data structure (the `ArrayList`) and also contains the “application” code. In later examples, this will go into a separate application class.
- Done right, the driver class can be reasonably standard for all programs in Java. It merely manages the housekeeping of “getting the program to run”

Documentation

- Standard Javadoc tags are required in this course
  - @author
  - @version
  - @param
  - @return
- <code> code font </code>
- Note that nested slash-star commenting doesn't work
- But slash-slash commenting nests inside slash-star

UML—Unified Markup Language

- UML is becoming the standard for displaying the organization of programs and the interrelationships of variables and methods
- UML comes close to being a Java `interface`
- There exist tools to create UML from existing code and to create code from UML diagrams

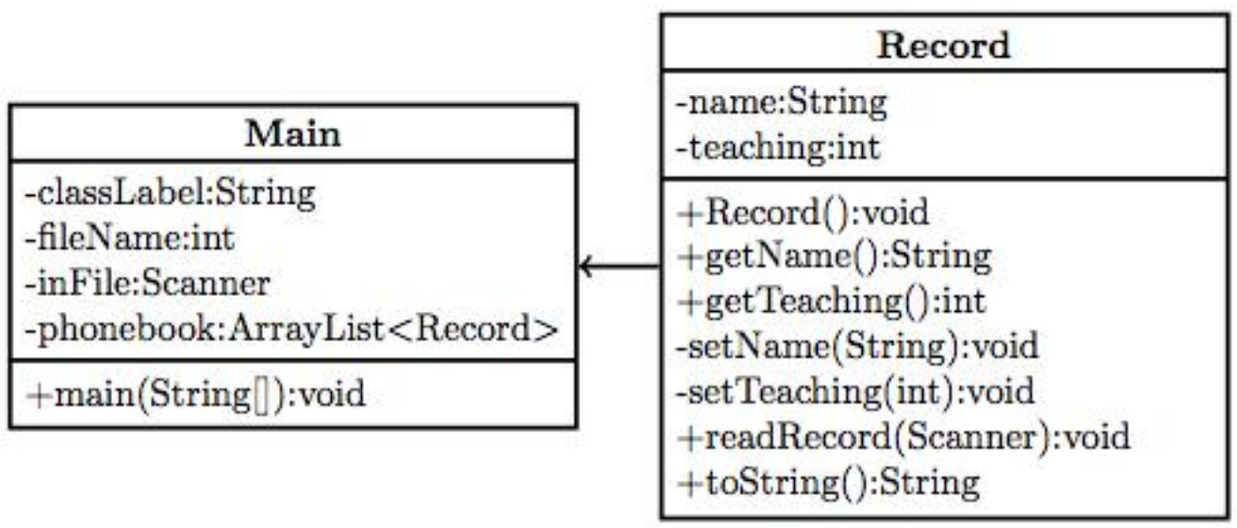


Figure 2.1 The UML diagram for the phone book