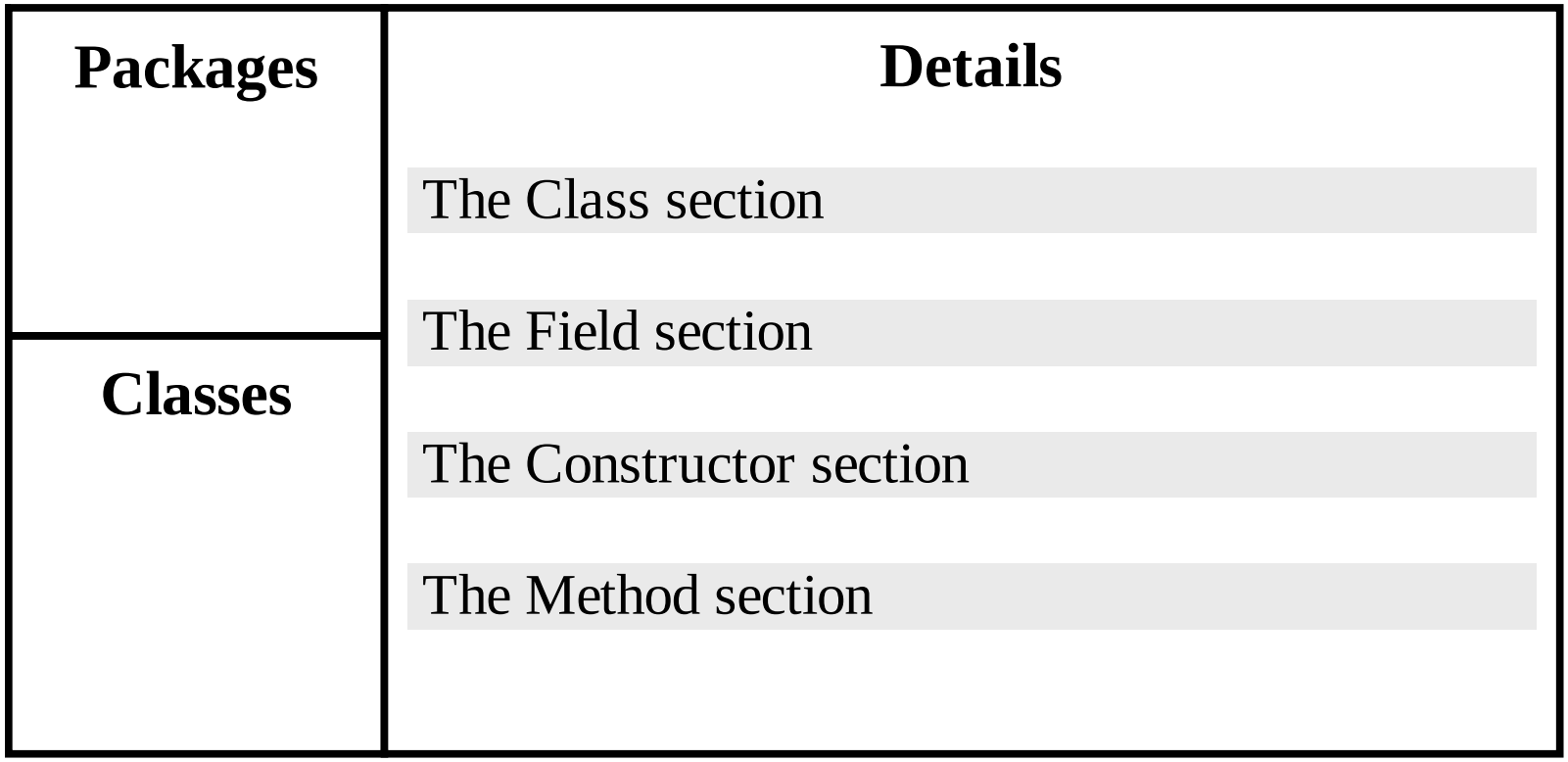
Java By Abstraction: Chapter 3

Using APIs
(Application Programming Interfaces)

Some examples and/or figures were borrowed (with permission)
from slides prepared by Prof. H. Roumani

API Layout


Packages
Classes
Details
The Class section
The Field section
The Constructor section
The Method section

API - Fields

Field Summary

`static double`	**`PI`** The `double` value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

Field Detail

`PI`

```
public static final double PI
```

The double value that is closer than any other to pi, the ratio of the circumference of a circle to its diameter.

See Also: Constant Field Values

API - Methods

Math class in java.lang package

Method Summary	
`static double`	**`abs`**`(double a)` Returns the absolute value of a `double` value.
`static int`	**`abs`**`(int a)` Returns the absolute value of an `int` value.
`static double`	**`pow`**`(double a, double b)` Returns the value of the first argument raised to the power of the second argument.

Scanner class in java.util package

Method Summary	
`double`	**`nextDouble`**`()` Scans the next token of the input as an `double`.
`int`	**`nextInt`**`()` Scans the next token of the input as an `int`.
`String`	**`nextLine`**`()` Advances this scanner past the current line and returns the input that was skipped.
`long`	**`nextLong`**`()` Scans the next token of the input as an `long`.

Passing Parameters

- Syntax
  - In API: *methodName*(*paramType param*)
  - In Code: *methodName*(*identifierOrLiteral*)
- Example
  - In API: print(String s)
  - In Code:

```
String prompt = "Enter value: ";
System.out.print(prompt);     // OR
System.out.print("Enter value: ");
```

Passing Parameters

- Similar to assignment statement
  - Parameter type evaluated
  - Promoted (if necessary)
  - Resulting value passed to method
- Technique referred to as **pass-by-value**

Overloading Methods

- Method signature
  - Method name + parameter types
  - Must be unique within a class regardless of return type
- Overloaded methods
  - Same name, but take different parameters
- Example
  - Math.abs(double a)
  - Math.abs(long a)

Input Validation

- Upon encountering invalid input, you could
  - Terminate the program and display a message
  - Explain the problem and have user re-enter input
  - Trigger an appropriate runtime error – crash
- For simplicity's sake, let's simply crash
  - type.lib.ToolBox.crash(boolean b, String s)
- Example
  - ToolBox.crash(amount < 0, "A negative amount!");

Development Process

- Task Analysis
- Algorithm Design
- Code Implementation
- Program Testing and Debugging

Task Analysis

- Input
  - amount: the present value
  - rate: the interest rate (% per annum)
- Output
  - The monthly payment formatted to two decimal places with a thousands separator
- Throws
  - Exception if amount < 0
  - Exception if rate <= 0
  - Exception if rate >= 100

Algorithm Design

$$P = \frac{rA}{1 - \frac{1}{(1+r)^n}}$$

P is the monthly payment, **r** is the monthly interest rate, **A** is the mortgage amount, and **n** is the number of months.

Code Implementation

- (To be demonstrated in class)

Program Testing and Debugging

- Pick example inputs for the program
- Use a calculator to determine expected results
- Compare expected results with actual ones
- Identify any sources of error in the program
  - Debug using print statements to output variable values
- Correct any errors
- Retest program