Elementary, my dear Java

Detecting Patterns in Object-Oriented Code

Jeremy.Singer@glasgow.ac.uk

Joint work with Yossi Gil

Modularity in OO languages

Package	milli-pattern
Class	micro-pattern
Method	nano-pattern

Definitions

A *pattern* is a discernable order or arrangement of elements, which may be repeated.

A *high-level design* pattern describes the structure of an OO system, based on class relationships and software architecture.

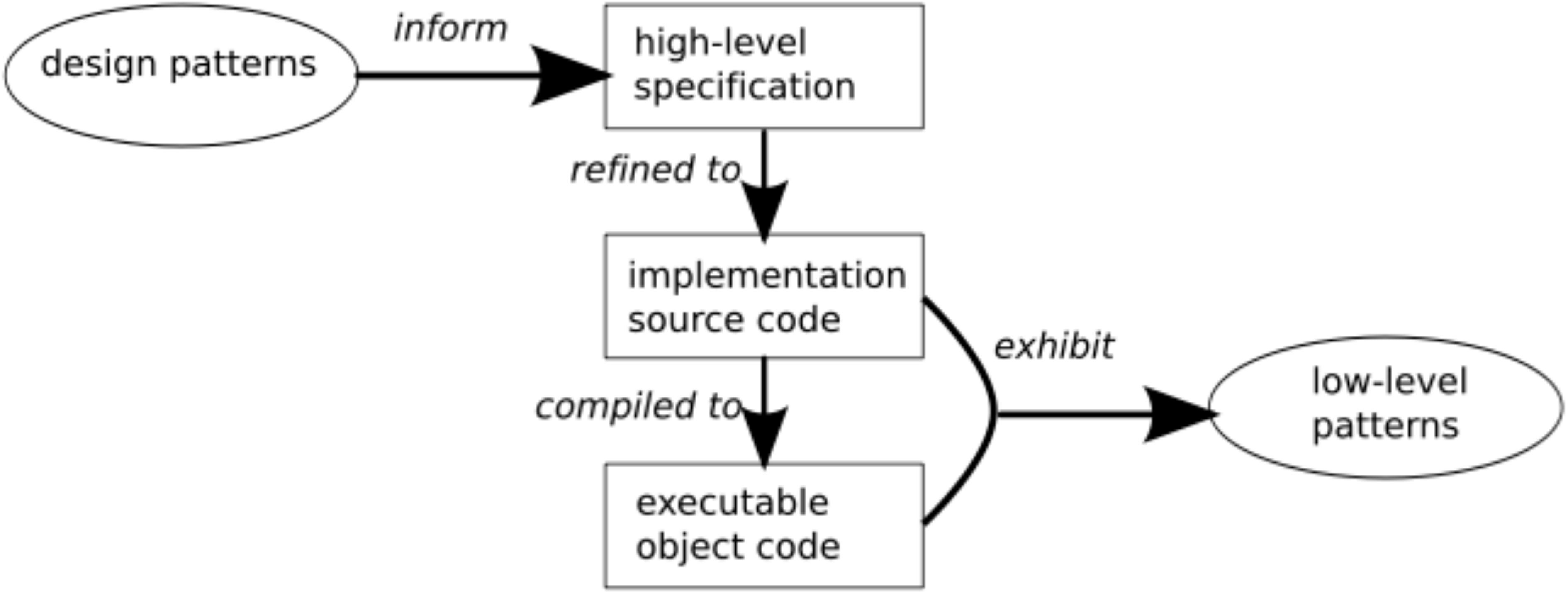
A low-level pattern …

Low-level patterns

A *low-level pattern* captures emergent properties of software, expressed in terms of source code artifacts.

Low-level patterns are:

- recognizable
- localized
- meaningful


design patterns
inform
high-level specification
refined to
implementation source code
compiled to
executable object code
exhibit
low-level patterns

Example patterns

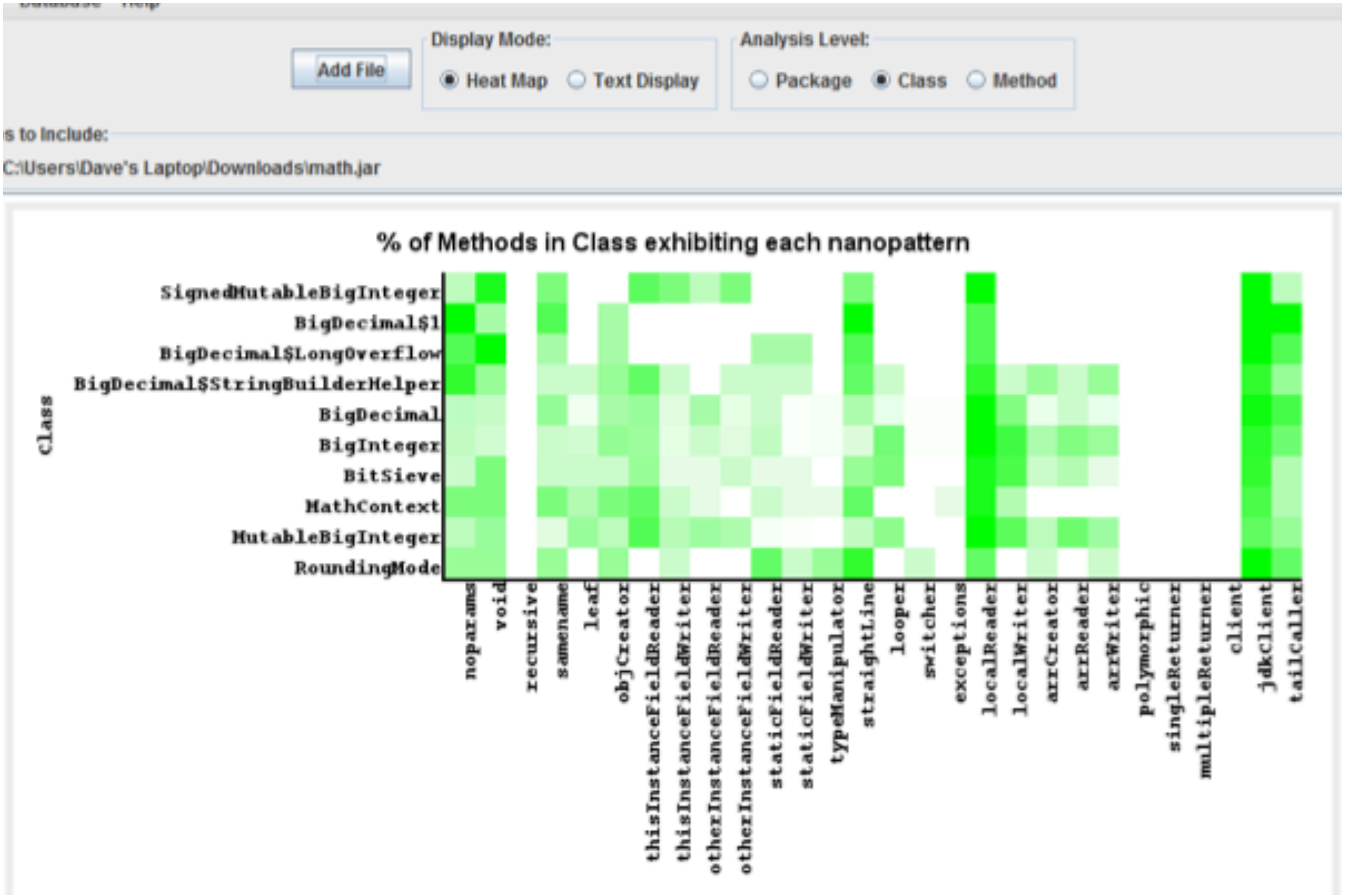
- Leaf package e.g. `java.math`
- Sampler class e.g. `java.awt.Color`
- Delegator method e.g. `size()`

Background

- Three key tasks – inter-dependent
  - specification
  - identification
  - detection

Pattern-based analysis

- Identifier name analysis
- Runtime behaviour analysis
- Bug detection analysis
- Program comprehension



Add File

Display Mode: ● Heat Map ○ Text Display

Analysis Level: ○ Package ● Class ○ Method

s to Include:

C:\Users\Dave's Laptop\Downloads\math.jar

Future work

- Construct a suite of package-level patterns
- Create a detection tool for these patterns (MSc project)
- Write up a survey paper on low-level patterns

Conclusion

- Low-level patterns are everywhere!

- This semantic information can be correlated with other static/dynamic properties and used for program analysis.