Cpt S 422: Software Engineering Principles II

Testing levels – Integration Testing

Dr. Venera Arnaoudova

WASHINGTON STATE UNIVERSITY

The Mars Climate Orbiter Mission

- Failed in September 1999, lost at beginning of Mars orbit

- Completed successful flight: 416,000,000 miles (665.600.600 km) and 41 weeks of flight

- An integration fault: Lockheed Martin Astronautics used English units for acceleration calculations (pounds), and Jet Propulsion Laboratory used metric units (newtons).

- NASA announced a US$ 50,000 project to discover how this happened.

Integration testing

- Goal
  - Gain confidence in the way the different components of the software are interacting, i.e., correct functionality across components and correct interfacing
- Assumptions
  - The individual components are unit tested

The “big bang” integration

- No...
  - stubs
  - drivers
  - strategy
- And very difficult fault isolation
- (Named after one of the theories of the origin of the Universe)
- This is the practice in an agile environment with a daily run of the project to that point.

Better way to perform integration testing?

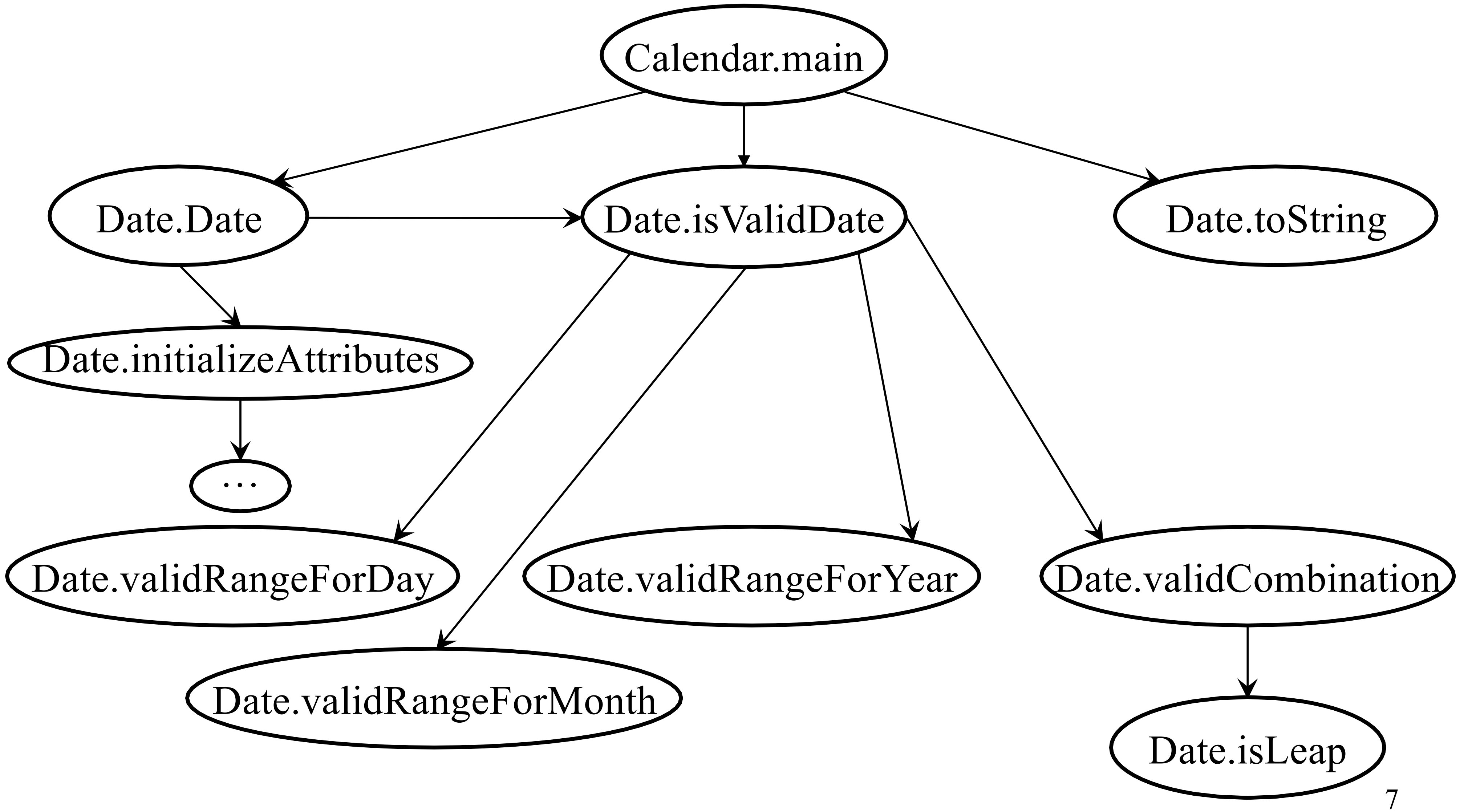
- **Iteratively**: integrate one unit at the time!
  - Easier to isolate faults

- Several integration strategies
  - Today we will practice **Call Graph-based integration**
    - Top-down
    - Bottom-up
    - Pair-wise

Program call graph

- ❑ Directed graph
- ❑ Nodes are units (e.g., methods)
- ❑ Edges are messages or calls to units (e.g., method calls)

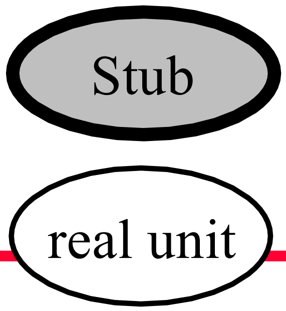
Partial call graph of the Calendar example

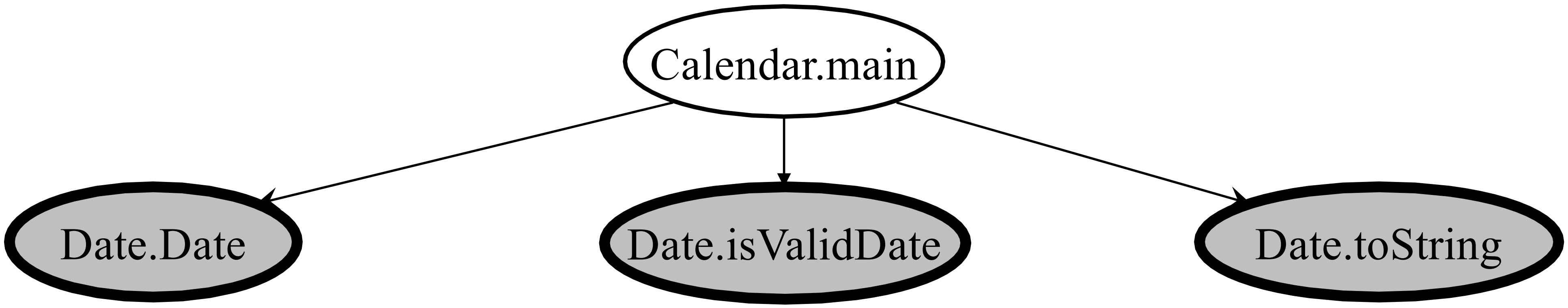


Top-down strategy

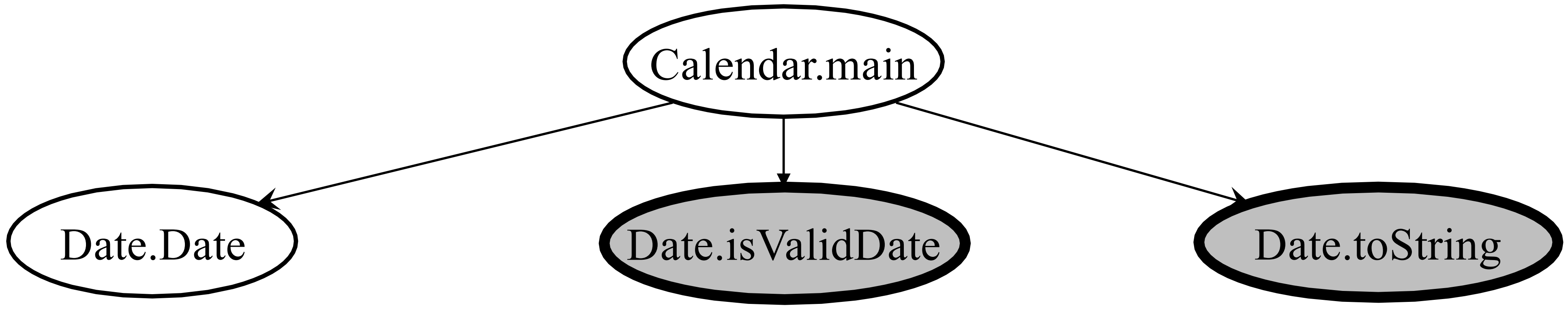
- Breadth-first traversal
- First step: Check main program logic, with all called units replaced by stubs that always return correct values.
  - Sounds familiar?
- Move down one level
  - Replace one stub at a time with actual code
  - Any fault must be in the newly integrated unit

Top-down in action - 0

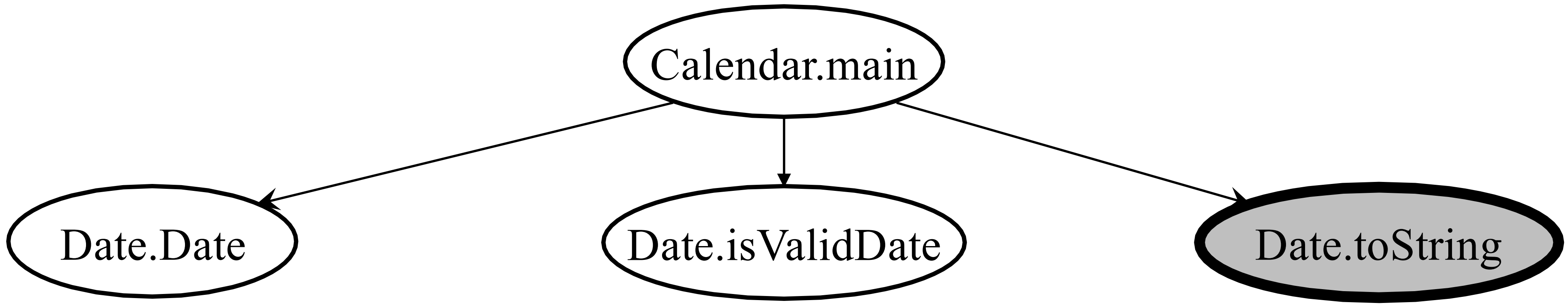




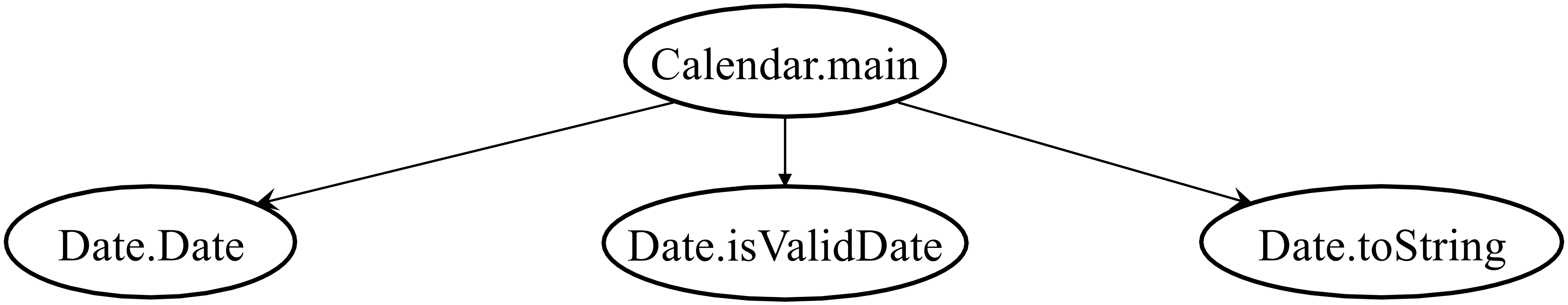
Top-down in action - 1



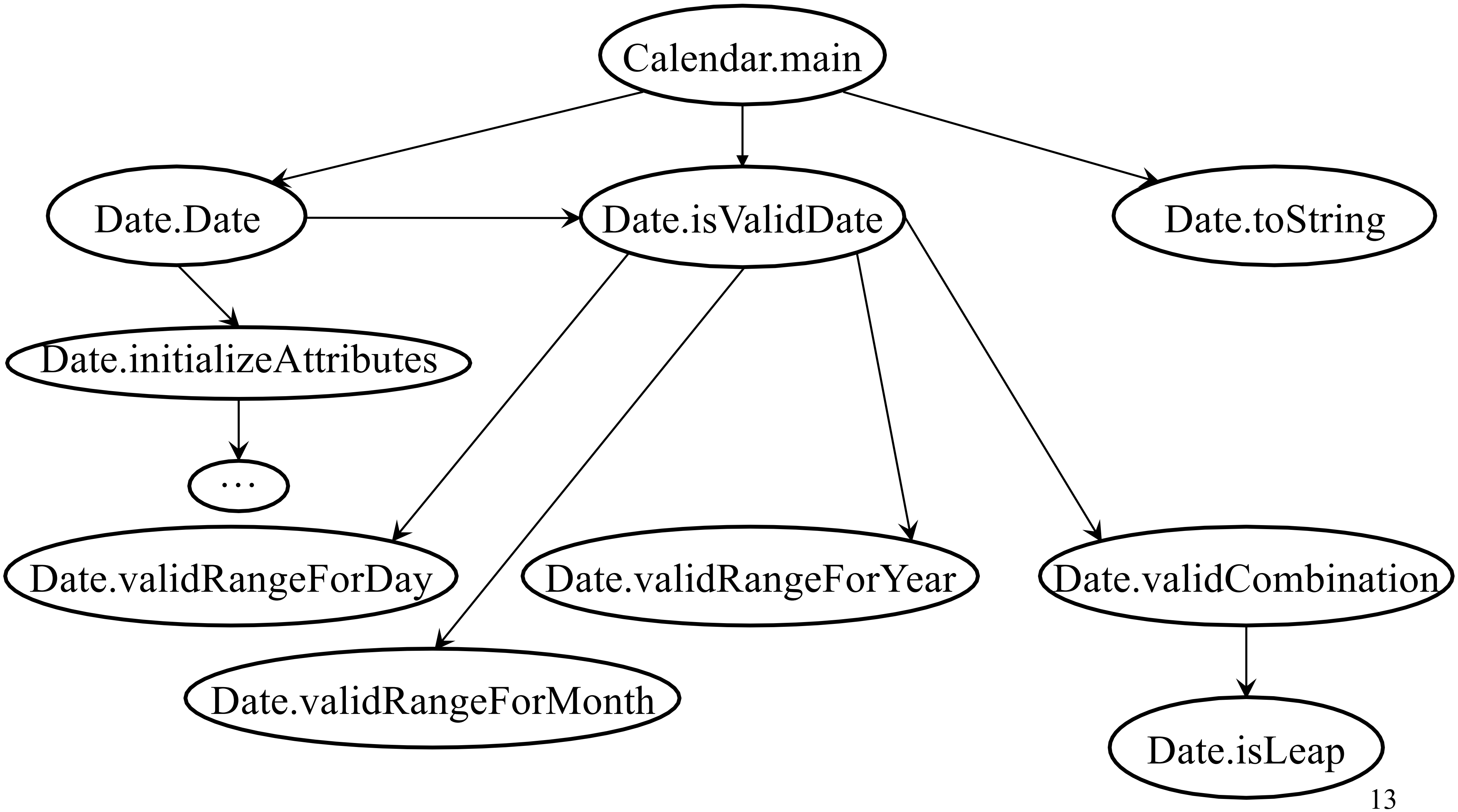
Top-down in action - 2



Top-down in action - 3



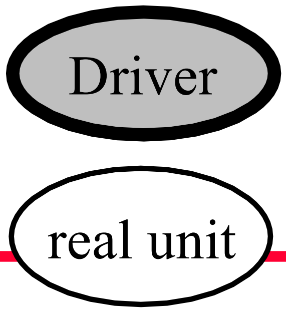
Top-down in action – n (where **n** is the number of nodes)

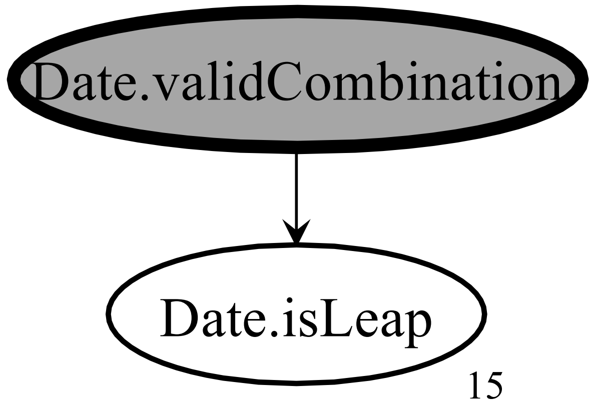


Bottom-up strategy

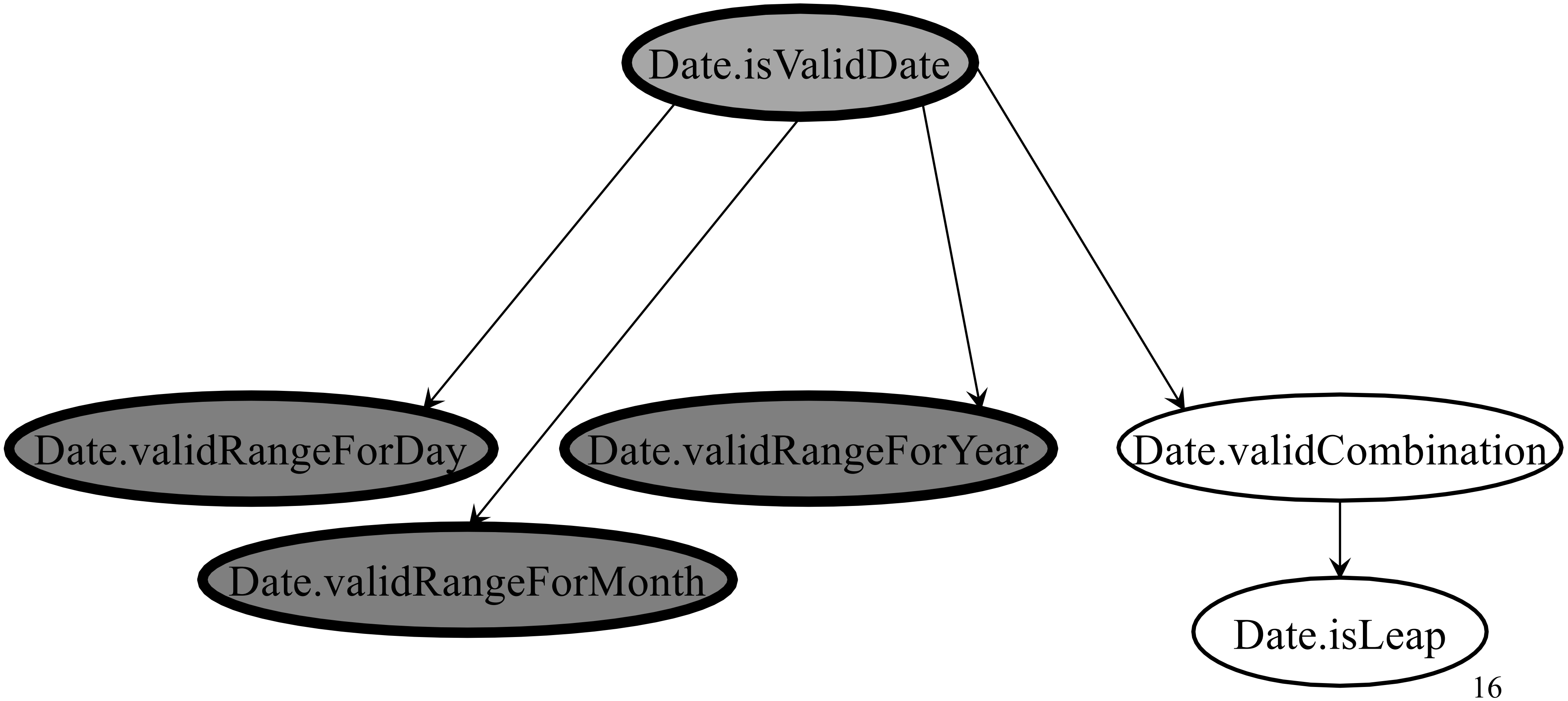
- Reverse of top-down integration
- Start at leaves
- Driver units...
  - call next level unit
  - serve as a small test bed
  - “drive” the unit with inputs
  - drivers know expected outputs
- As with top-down integration, one driver unit at a time is replaced with actual code.
- Any fault is (most likely) in the newly integrated code.

Bottom-up in action - 0

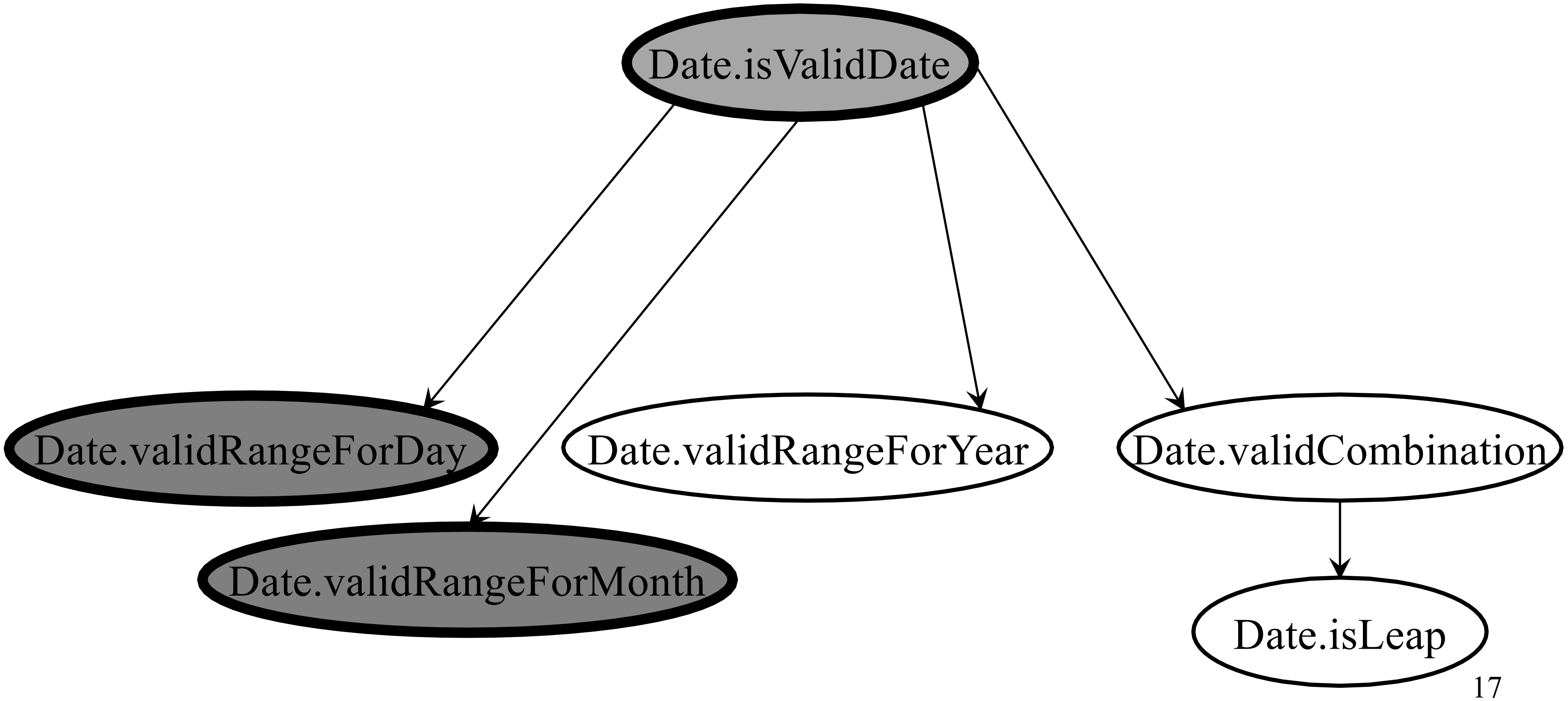




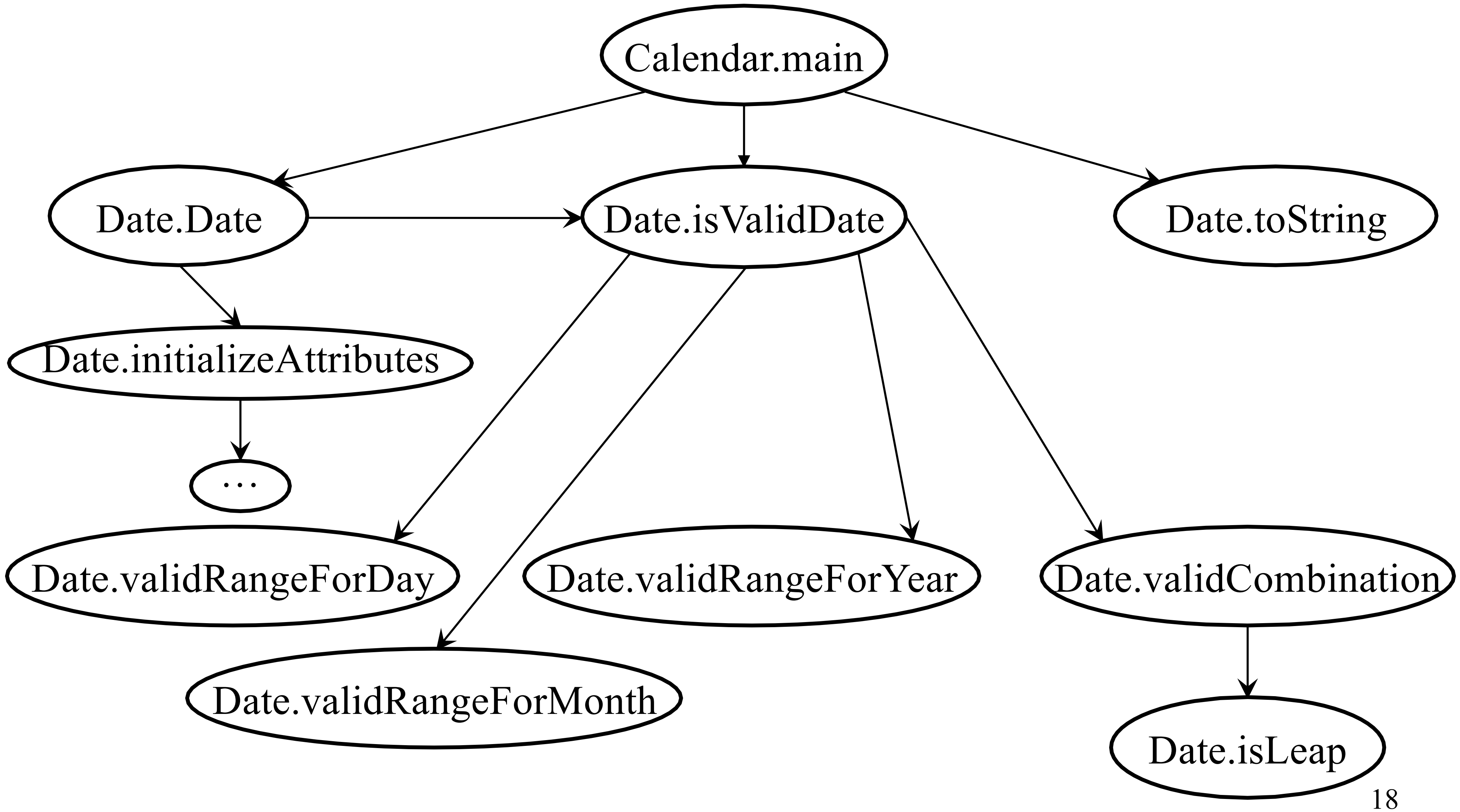
Bottom-up in action - 1



Bottom-up in action - 1



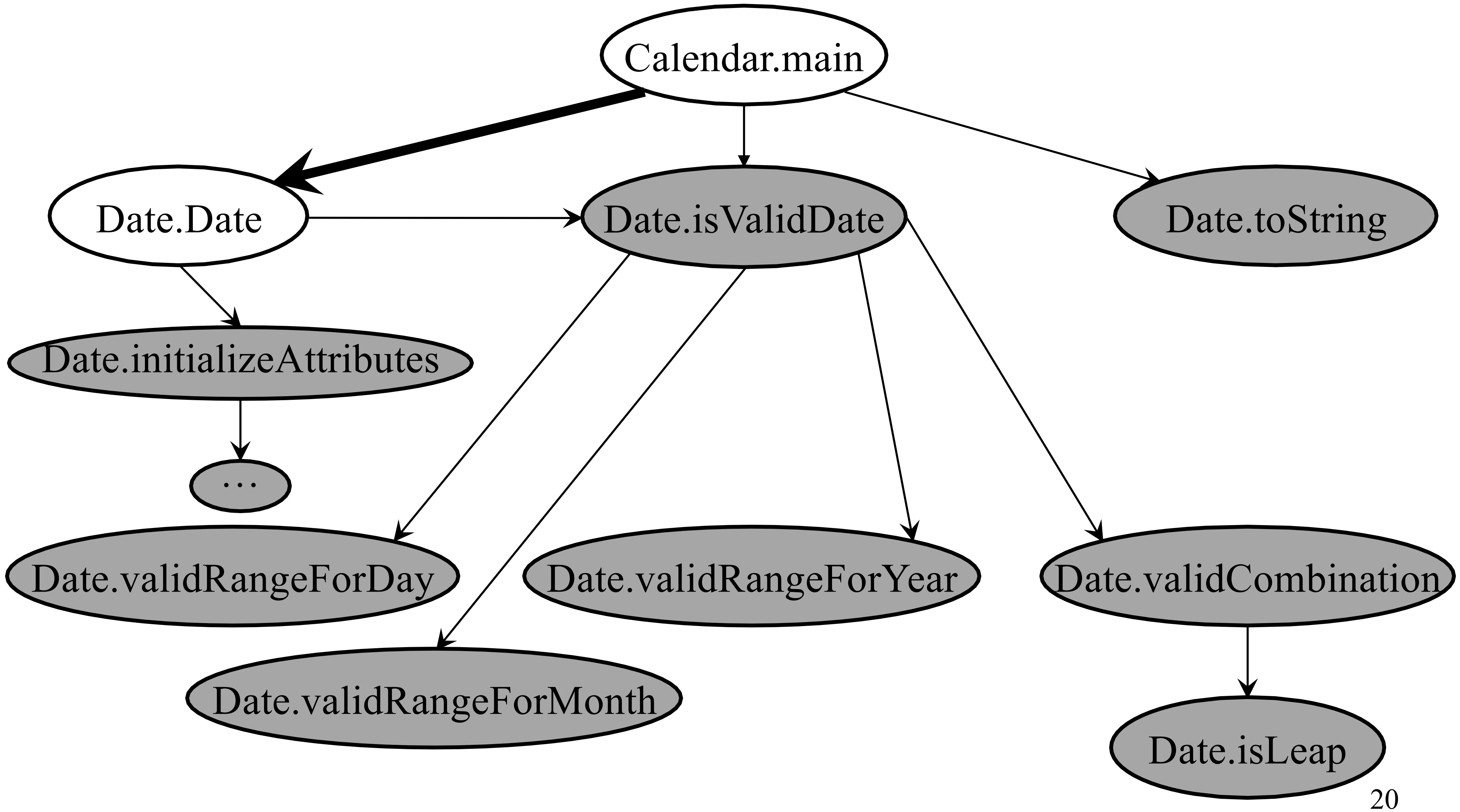
Bottom-up in action – n (where **n** is the number of nodes)



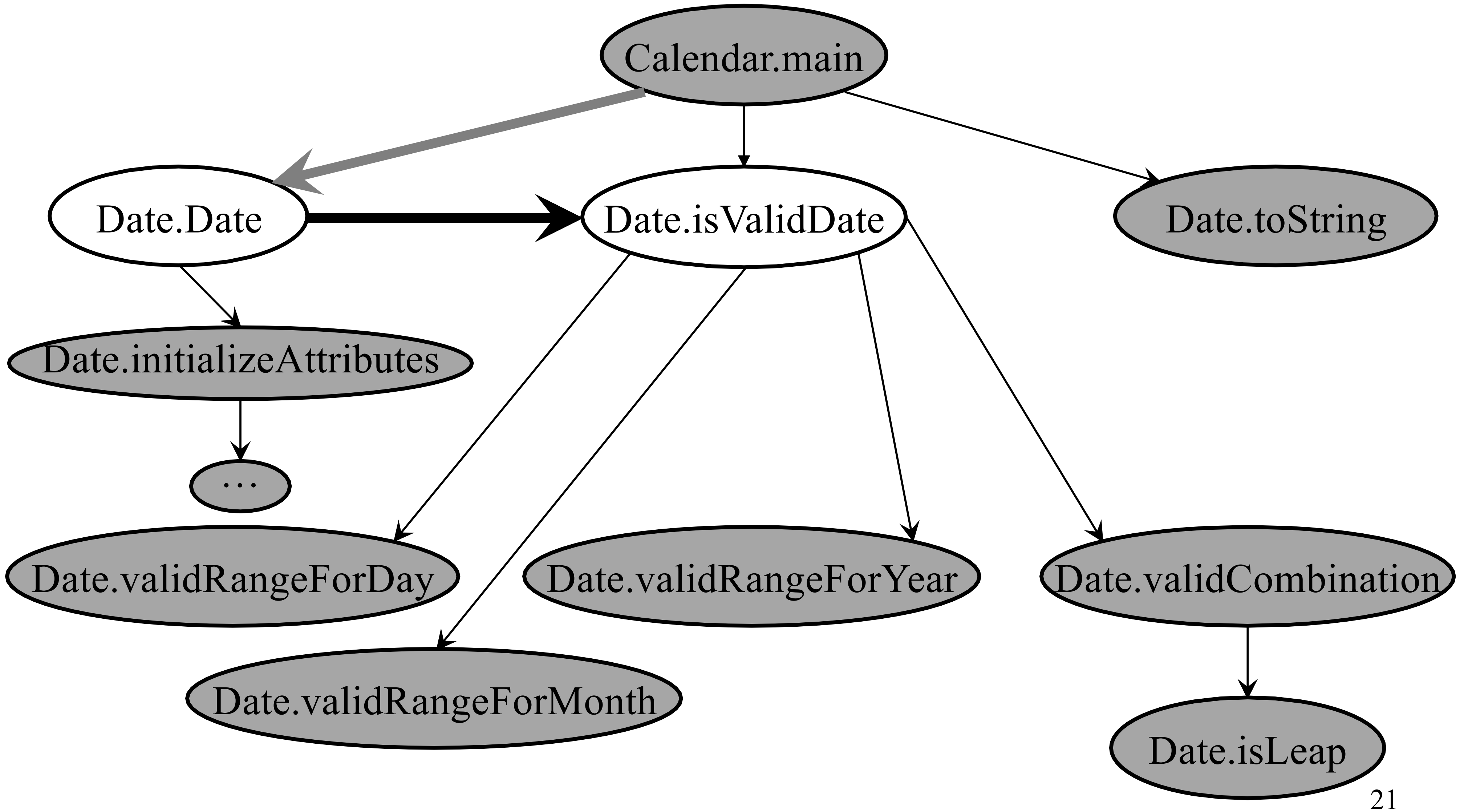
Pair-wise strategy

- By definition, and edge in the Call Graph refers to an interface between the units that are the endpoints of the edge
- Every edge represents a pair of units to test
- Still might need stubs and drivers
- Fault isolation is localized to the pair being integrated

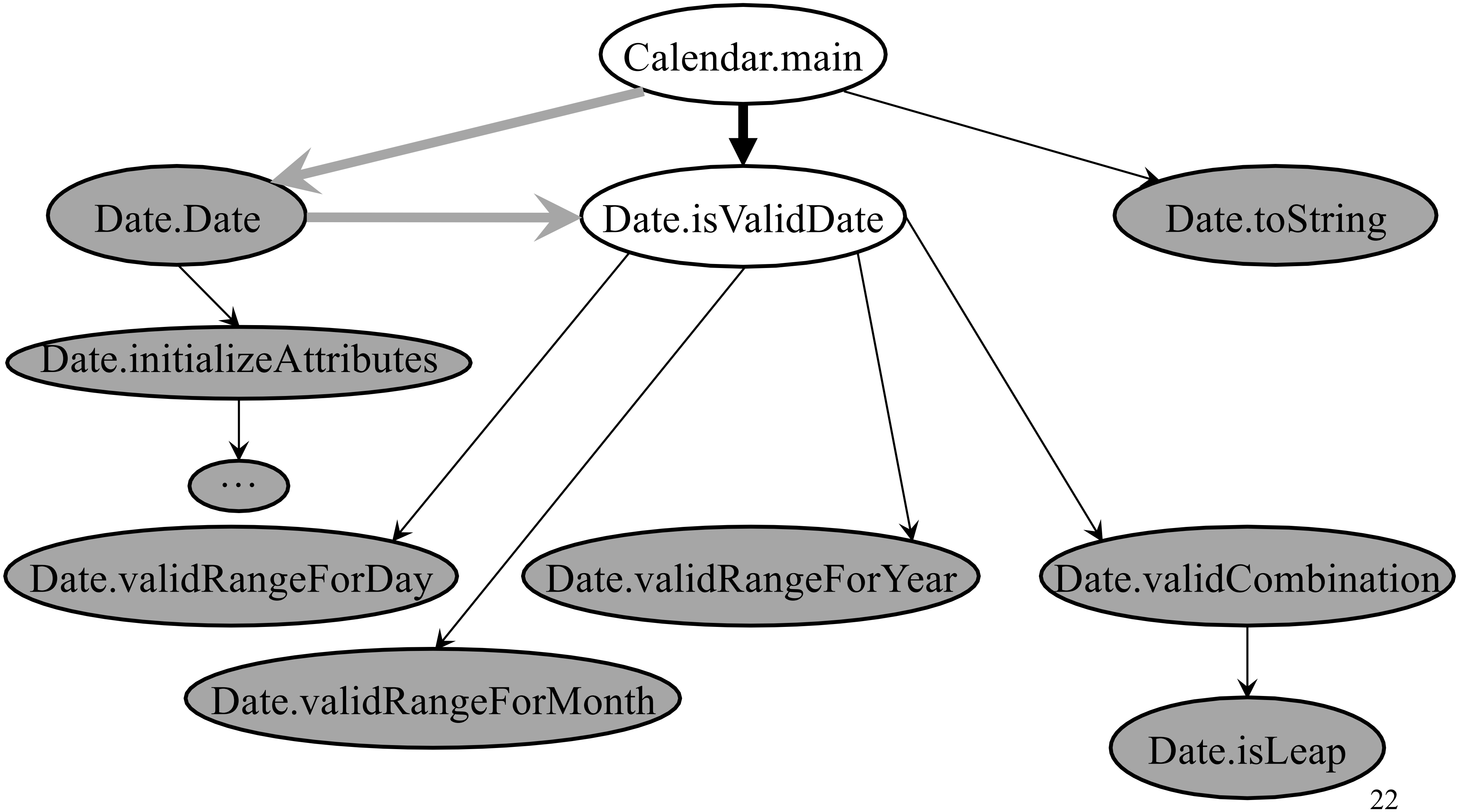
Pair-wise in action – 1



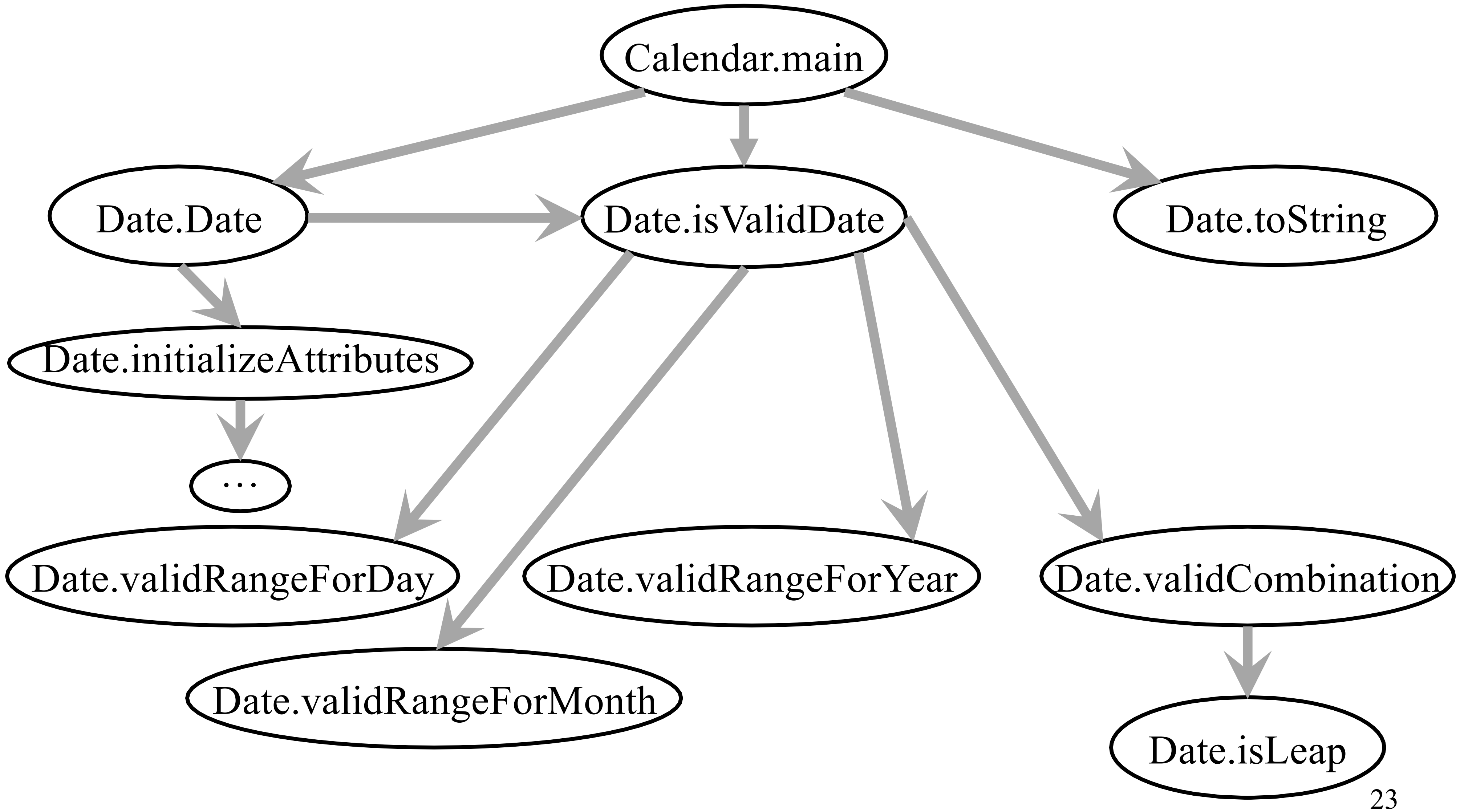
Pair-wise in action – 2



Pair-wise in action – 3



Pair-wise in action – n (where **n** is the number of edges)



Tasks for today

- Practice integration testing on our Calendar example using a
  - Bottom-up strategy
  - Pair-wise strategy
  - Top-down strategy