Compact Views of Java Memory
Steven Reiss, Brown University

For Large Arrays:
- [Type*Size]
For Large Strings:
- String*Size
For Collections:
- Class←Source

For each class C₁:
- # instances
- Total size
For each pair of classes, C₁, C₂:
- # references from C₁ to C₂

Assigning storage per reference based on:
- # references
- Size correlations
- Count correlations
- Reference correlations
Using constrained least-squares fit

For Large Arrays:
- [Type*Size]
For Large Strings:
- String*Size
For Collections:
- Class←Source

For each class C₁:
- # instances
- Total size
For each pair of classes, C₁, C₂:
- # references from C₁ to C₂

Assigning storage per reference based on:
- # references
- Size correlations
- Count correlations
- Reference correlations
Using constrained least-squares fit

TOP HATCHING: % of new objects
FORK: Current time of the dump.
GRAPH: Storage owned by class over time
HUE: Storage for this class's objects
SATURATION: Fraction represented by node
HEIGHT: Storage owned by class
BOTTOM HATCHING: Increase in size