**Real-time display of profiling data**

With JProfiler, users can immediately observe profiling data as their application is running. CPU, memory and thread profiling views are updated live and can be inspected without the need to wait for the measurement to complete.

|  |
| --- |
| **Demonstration of unique heap walker** |
| Profiler’s heap walker has intuitive interface to solve both simple and complex memory problems. Five different views show different aspects of the current set of objects.   |  | | --- | | **Providing powerful CPU profiler** | | Fixing performance bottlenecks is the most frequent use case for a profiler. CPU data can be overwhelming in its level of detail and the way data is collected can make a huge difference in usability. With JProfiler, users have a decisive advantage when trying to find the reason for a problem.  **Support for Java Enterprise Edition**  Dedicated support for EE is present in most of the views. Using JEE aggregation level you can see the call tree in terms of the JEE component in your application. Also the call tree is split based on the up for each request URI | |

**Probe System**:

Presents high level information about files, sockets and processes within various views. You get an instant overview of the activity of open connections, files etc. To easily analyze the data flow in complex subsystems like JPA, JProfiler can present you a chronological list of all events of a selected probe.

**Providing extensive QA capabilities**

JProfiler is ideally suited as a QA tool, both during development as well as for dedicated QA teams. The rich functionality around snapshot comparisons makes it easy to track progress. JProfiler has strong support for command line operations