# Objectreferenceanalyser {Perusal}

[http://sf.net/projects/refanalyse/]

Quick introduction

# Content

1. The features
2. The use
3. ORA Application UI explanation
4. Example1: Tracking of newly created objects or (not) freed objects.
5. Example2: Persist state of objects

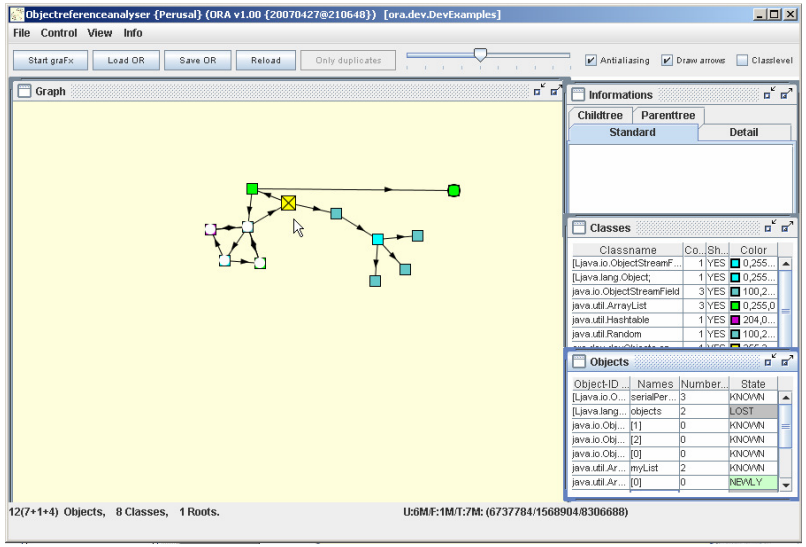
# 1. The features

1. Object analyzing engine
   a) Customizable and abstracted framework for extensionality
   b) Public API for direct control or automated tasks
   c) Analyze of objects and their variables including references recursively
   d) Persistence of reference model and object contents for later/offline analyze and tracking
2. Object reference visualizer
   a) Object overview and details
   b) Class overview
   c) Class level reference visualization (~ ER-Diagram at runtime)
   d) Graphical representation of object references
3. Tool library
   a) Object cloning framework for deep duplication of object structures

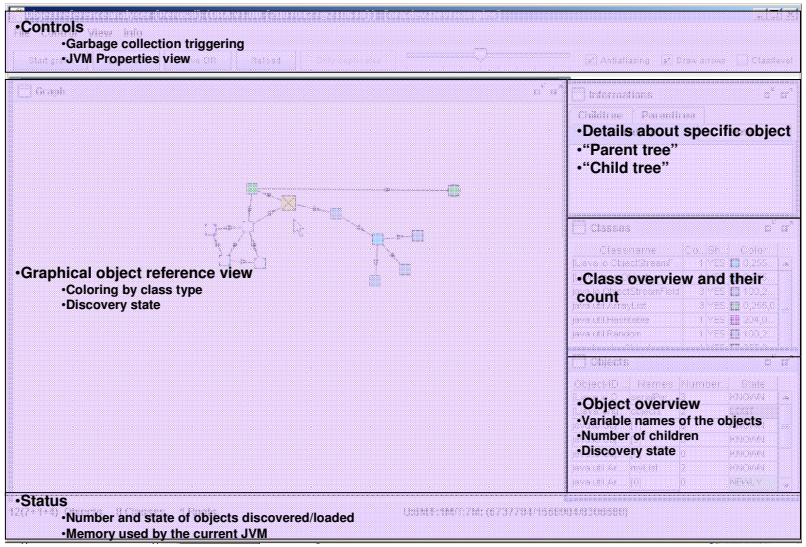
# 2. The use

1. Identifying and understanding architecture/design of applications
2. Determination design and architectural mistakes or deficits in applications
3. Isolation of memory leaks which impact application stability and performance
4. Bug and exception discovery/troubleshooting by capturing object model at runtime

# ORA Application UI explanation




# ORA Application UI explanation




# Example1: Tracking of newly created or freed objects

- Required Steps:
  1. Integrate ORA into the application
  2. Start application
  3. Prepare application for start point
  4. Request ORA to reload
  5. Prepare application for end point
  6. Request ORA to reload

# Example1: Tracking of newly created or freed objects

1. Integrate ORA into the application:
   a) Integrate ORA-jar into classpath
   b) Decide about to be analysed "root"/"source"-object
   c) Link this object to ORA using code like below:

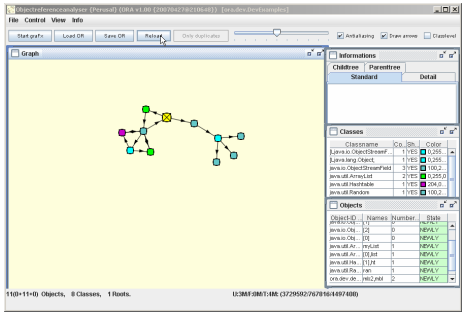
```
public void analyseAndShow(Object obj) throws Exception
{
    AnalyserInterface ref = new ObjectReferencesExceptSunsUIPackages(getClass().getName());
    ref.addAndAnalyseRootObject(obj);
    new Visualizer(ref).start();
}
```

# Example1: Tracking of newly created or freed objects

1. *Integrate ORA into the application* ☑
2. Start application
   a) Insure that the ORA-UI appears by reaching the previous mentioned code.
3. Prepare application for start point
4. Request ORA to reload

# Example1: Tracking of newly created or freed objects

1. *Integrate ORA into the application* ☑
2. *Start application* ☑
3. *Prepare application for start point* ☑
4. Request ORA to reload

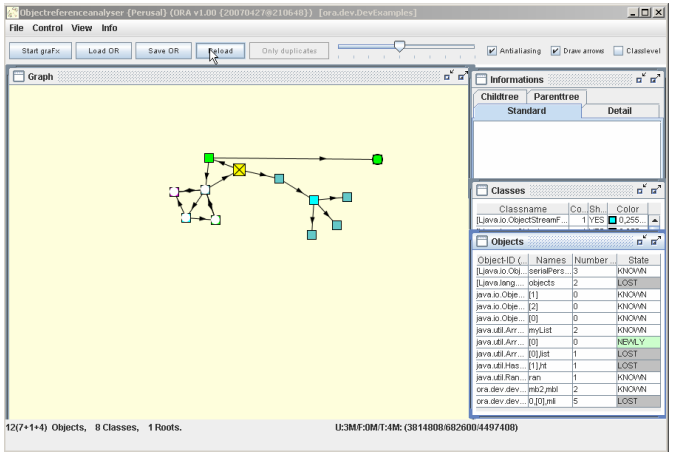


# Example1: Tracking of newly created or freed objects

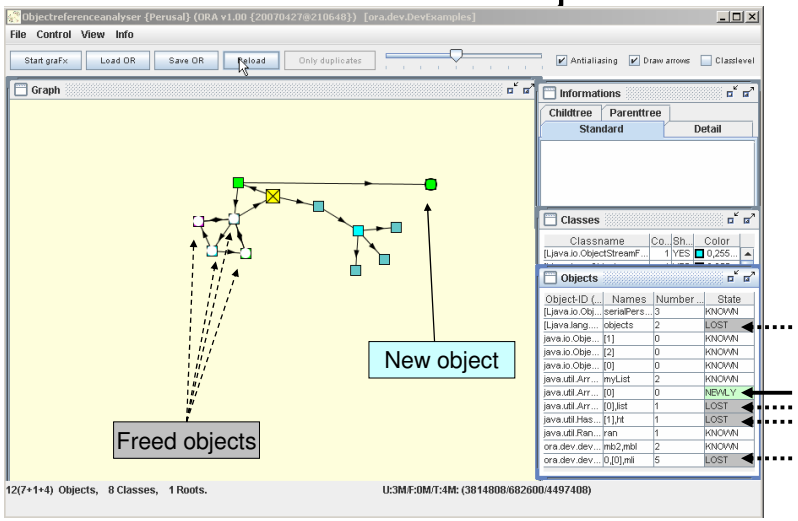
1. *Integrate ORA into the application* ☑
2. *Start application* ☑
3. *Prepare application for start point* ☑
4. *Request ORA to reload* ☑
5. *Prepare application for end point* ☑
6. Request ORA to reload

# Example1: Tracking of newly created or freed objects

6. Request ORA to reload ☑



# Example1: Tracking of newly created or freed objects




# Example2: Persist state of objects

The following code analyzes given object and stores the result in a file.

```
public void takeSnapshotOfMyObjectInFile(Object obj, String filename) throws Exception
{
    // This code snippet analyses given object
    // and saves referencemodel to given filename
    AnalyserInterface or = new ObjectReferencesExceptSunsUIPackages();
    or.addAndAnalyseRootObject(obj);
    or.save(new File(filename));
}
```

Snapshots can be taken at any point of time out of any reason and analyzed at any other time.

# The end

- Thank you