Introduction to FlexGantt

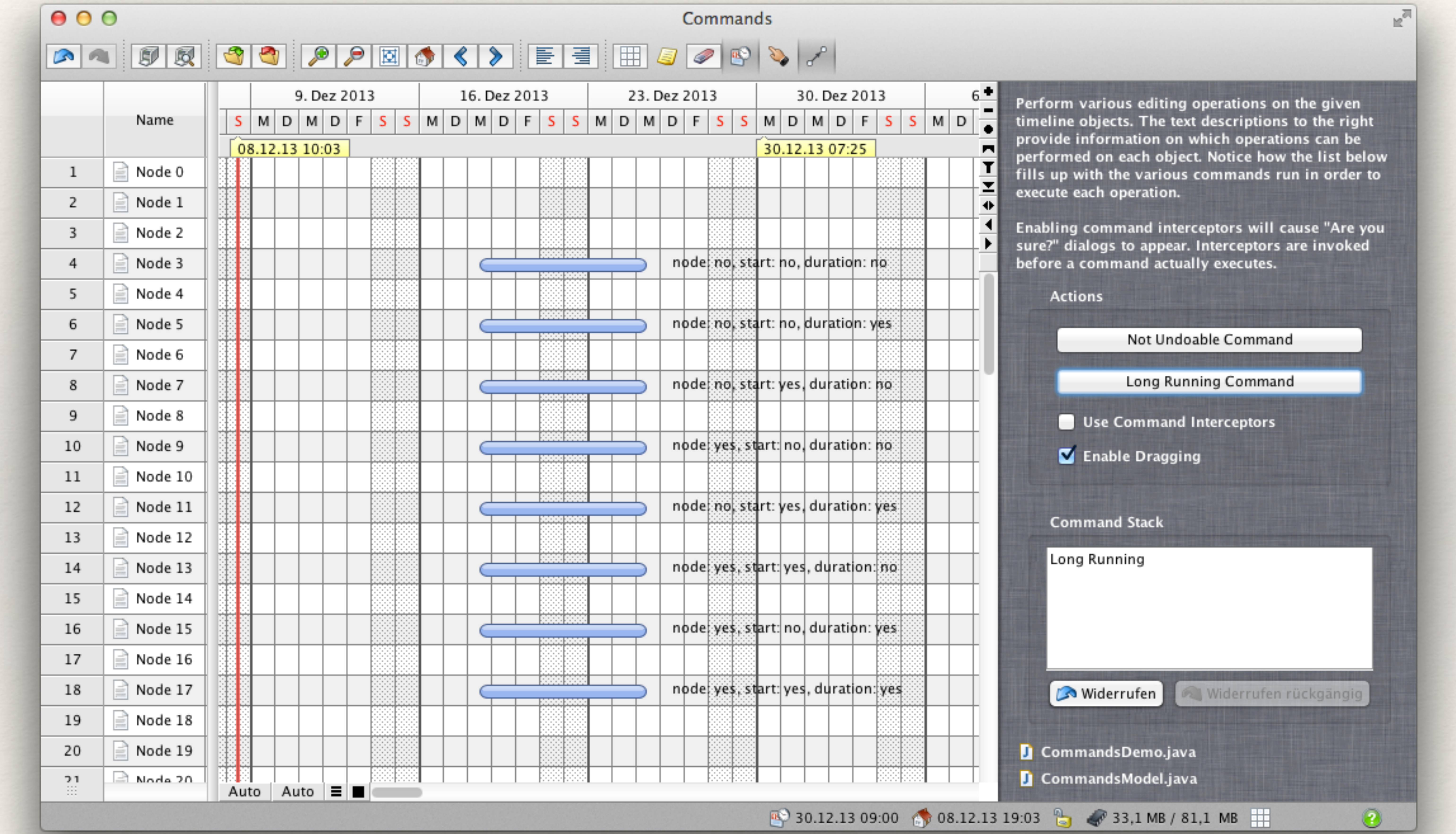
Topic: Commands

Dirk Lemmermann
Software & Consulting
Zurich, Switzerland

Command

- Commands are executed in the background.
- Run in separate thread.
- Passed to a command stack.
- Can be undone and redone.
- Report their execution progress.
- Can be composed to composite commands.

Commands Demo



	Name	
1	Node 0	
2	Node 1	
3	Node 2	
4	Node 3	node: no, start: no, duration: no
5	Node 4	
6	Node 5	node: no, start: no, duration: yes
7	Node 6	
8	Node 7	node: no, start: yes, duration: no
9	Node 8	
10	Node 9	node: yes, start: no, duration: no
11	Node 10	
12	Node 11	node: no, start: yes, duration: yes
13	Node 12	
14	Node 13	node: yes, start: yes, duration: no
15	Node 14	
16	Node 15	node: yes, start: no, duration: yes
17	Node 16	
18	Node 17	node: yes, start: yes, duration: yes
19	Node 18	
20	Node 19	

Perform various editing operations on the given timeline objects. The text descriptions to the right provide information on which operations can be performed on each object. Notice how the list below fills up with the various commands run in order to execute each operation.

Enabling command interceptors will cause "Are you sure?" dialogs to appear. Interceptors are invoked before a command actually executes.

☐ Use Command Interceptors

☑ Enable Dragging

Command Stack:

Long Running

ICommand

```
public interface ICommand extends Serializable {

   void executeCommand(IProgressMonitor monitor) throws CommandException;

   void undoCommand(IProgressMonitor monitor) throws CommandException;

   void redoCommand(IProgressMonitor monitor) throws CommandException;

   boolean isUndoable();

   boolean isRedoable();

   boolean isRelevant();

   String getName();
}
```

Example: Set Key on Node

```
public class DefaultChangeKeyCommand extends AbstractCommand {
   private Object oldKey;
   private Object newKey;
   private DefaultGanttChartModel model;

   …

   public void executeCommand(IProgressMonitor monitor)
      throws CommandException {
      monitor.beginTask("Changing key to: " + newKey, 1);
      oldKey = model.getKey(node);
      model.setKey(node, newKey);
      monitor.done();
   }

   …
}
```

Composite Commands

- Executes individual commands as a single command.
- All sub-commands are executed, undone, redone as one.

CommandStack

- Central place for executing commands.
- Each Gantt chart has its own command stack.
- Applications can choose to set the same stack on all Gantt charts and related views.

ICommandStack

```java
public interface ICommandStack {
    void execute(ICommand cmd, IProgressMonitor monitor);
    void undo(IProgressMonitor monitor);
    void redo(IProgressMonitor monitor);
    …
    void addCommandStackListener(ICommandStackListener l);
    void removeCommandStackListener(ICommandStackListener l);
```

Command Stack Listener

- Listeners can be attached to the stack to receive events when commands are started, executed, cancelled, failed, undone.

```
public interface ICommandStackListener extends EventListener {

    /**
     * Gets called whenever the command stack changed. The event object that is
     * passed to this method contains information about the type of event and a
     * reference to the command that caused the event.
     */
    void commandStackChanged(CommandStackEvent e);
}
```

Command Stack Event

```
public enum ID {
   COMMAND_CANCELED, COMMAND_EXECUTED, COMMAND_FAILED,
   COMMAND_STARTED, COMMAND_UNDONE
}

public CommandStackEvent(ICommandStack stack, ICommand command,
      ID id, Exception ex) {
   …
}

public ID getId() {}

public ICommand getCommand() {}
```

Progress Monitor

- Used to report progress on an activity.
- Much more sophisticated approach then just min, max, value progress.
- Supports sub progress monitors.
- NullProgressMonitor for unknown amount of work.
- Implemented by GanttChartProgressMonitor (standard Swing progress monitor), GanttChartGlassPane, and GanttChartStatusBar.

Command Interceptors

- Used to „intercept“ commands BEFORE they are being executed.
- Registered with the Gantt chart (not the command stack).
- Only called when using AbstractGanttChart.executeCommand(ICommand);
- Used for user feedback, populate commands with data, cancel commands.

Command Interceptor Example

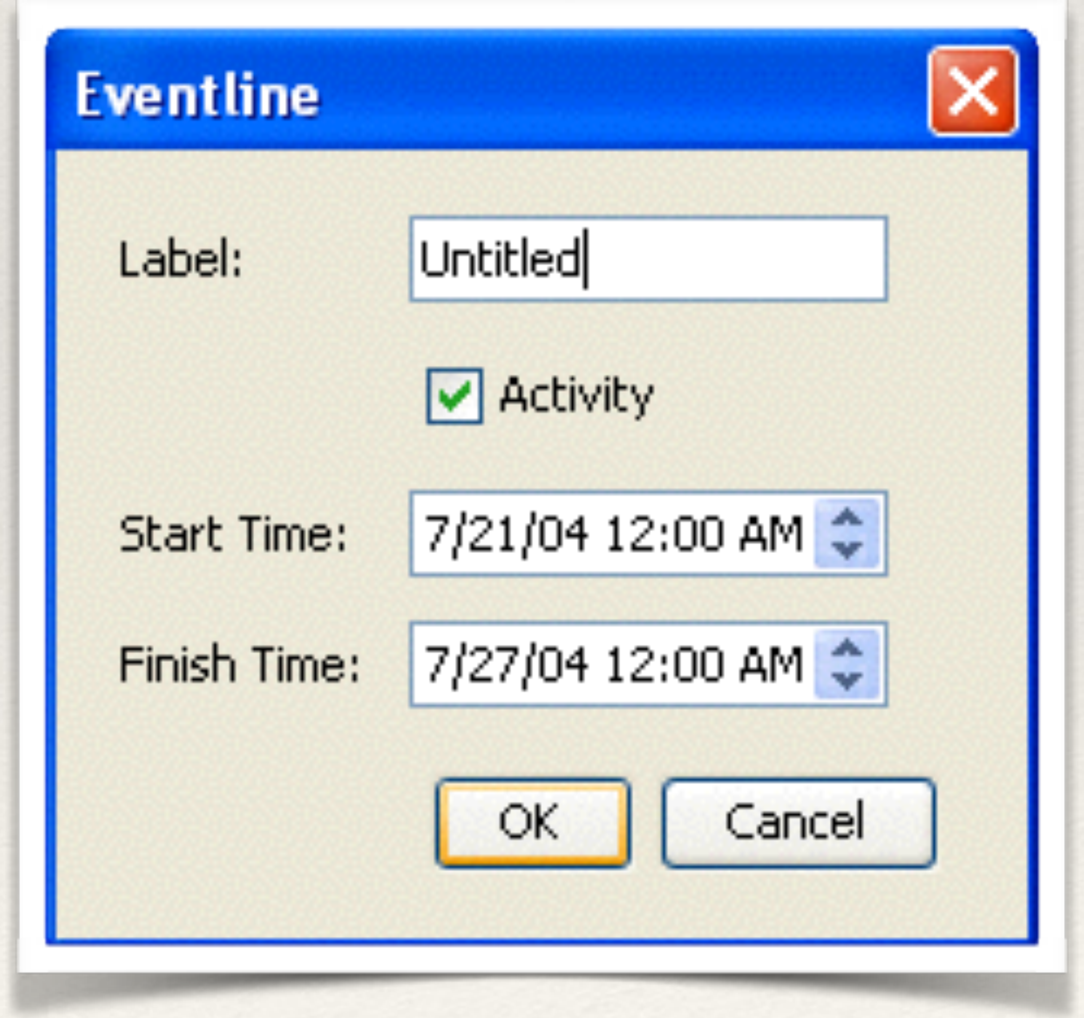
- By default every event line registers a listener to intercept the command that creates an eventline object.

```
gc.setCommandInterceptor(DefaultCreateEventlineObjectCommand.class,
            new ICommandInterceptor() {
                  public boolean intercept(AbstractGanttChart gc, ICommand cmd) {
                        DefaultCreateEventlineObjectCommand createCmd = (DefaultCreateEventlineObjectCommand) cmd;

                        EditDialog dialog = new EditDialog(createCmd);
                        dialog.setVisible(true);

                        if (!dialog.isCancelled()) {
                              createCmd.setTimeSpan(panel.getTimeSpan());
                              createCmd.setEventlineObjectName(panel
                                          .getLabelField().getText());
                              return true;
                        }

                        return false;
                  }
            });
```



Label: Untitled

☑ Activity

Start Time: 7/21/04 12:00 AM

Finish Time: 7/27/04 12:00 AM

- Create NotifyUserCommand, implement ICommand
- Pass command to GanttChart.commandExecute()
- Create NotifyUserCommandInterceptor, implement ICommandInterceptor
- Bring up a dialog in intercept() method to confirm command execution
- Register interceptor via AbstractGanttChart.setCommandInterceptor()
- Run command again