Systematic Software Development

Module Leader: Peter Rosner N211
rosnerpe@lsbu.ac.uk, X7570

In this lecture we will

- Talk about general features of the module
- Discuss aspects of the coursework
- Take an initial look at some user-interface components
- Start talking about state diagrams.

Features of module

- Lectures
- Lab sessions
- Lab sessions give opportunity for practice
- Assessment is 60% coursework, 40% exam

Assessment

- Coursework, 60% of total mark
  - Part 1
    - simple prototype design
    - 5% of coursework mark
    - Full marks or no marks
  - Part 2 Activity/assessment
    - simple prototype re-design/construction
    - 95% of coursework mark
- Exam, 40% of coursework mark
  - based on material covered in lectures.

Text books

There is no fixed core text book for the module. Most material for use in exercises and coursework should be obtainable from the lectures. However useful text book resources are:

- *An Introduction to Java Programming, Comprehensive Version Ninth Edition by* Y. Daniel Yi
  - a very good general textbook on Java which includes extensive material on Swing, but be careful – the example for event handling do not follow the three layer architecture used in this module.
- *Software Engineering, a Programming approach,* 4th Edition, by Douglas Bell
  - Contains support material for some of the software engineering topics discussed in the lectures.

On-line Resources

- Oracle's **on-line tutorial** is useful
  (http://download.oracle.com/javase/tutorial/uiswing/index.html)
- **Use the on-line html Java documentation**
  (http://docs.oracle.com/javase/7/docs/api/)

Aims of Module

"The module aims to extend your understanding of the issues involved in software development. It covers a range of good practices, many of which are reinforced by the practical work that you undertake. The emphasis is on developing high quality software through the embodiment of key principles such as the separation of concerns and the "keep it simple" maxim. You will also develop new skills in the area of GUI design and implementation as well as in systematic development and testing. "

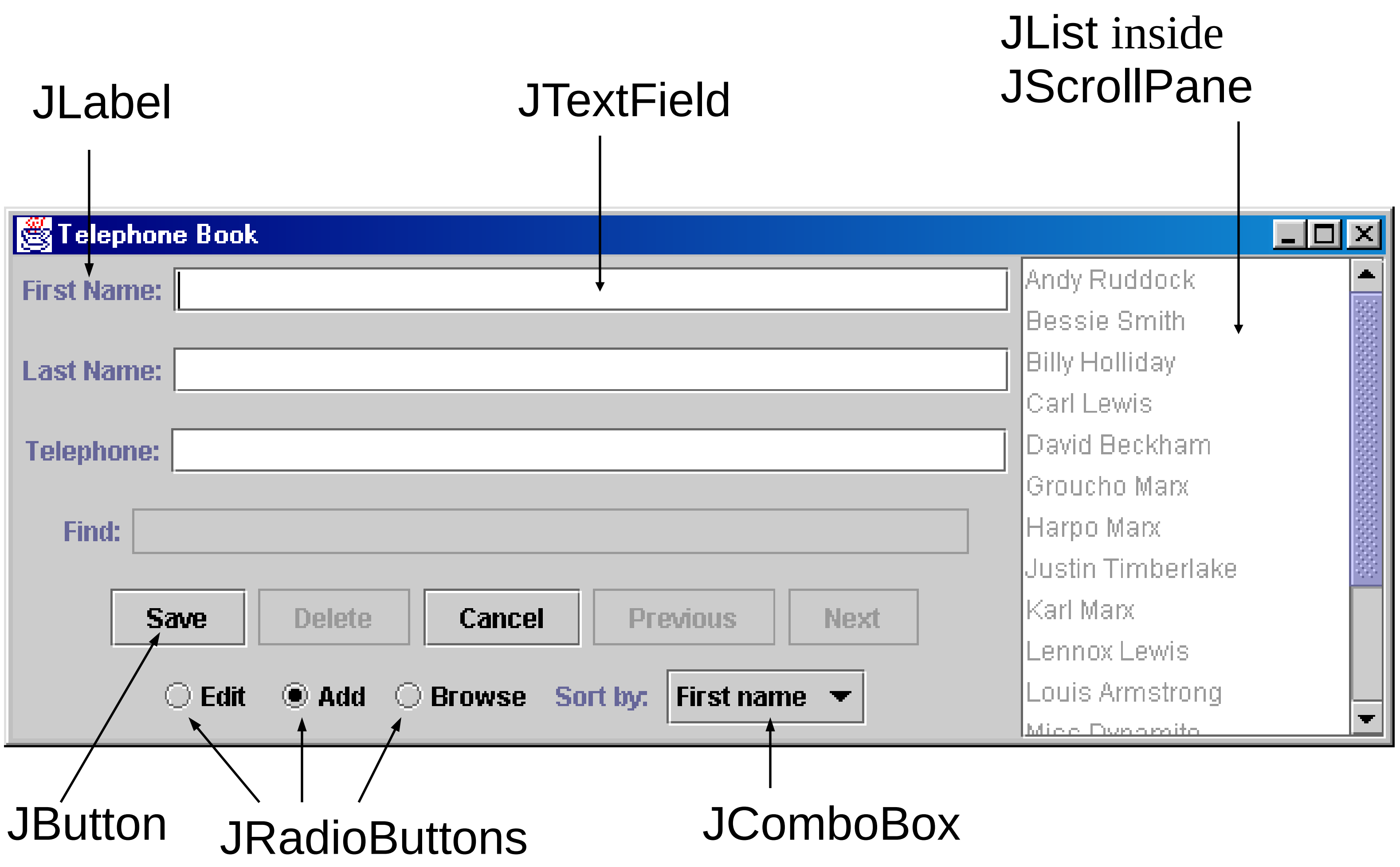
Main aspects of module

- Java implementation of user-interface
  - Use *Swing* class in the class library
- Separation of concerns using three layer architecture for interactive systems
- Software engineering issues including
  - writing a user manual
  - software development lifecycle
  - Requirements engineering
  - testing of software.

NetBeans

- The interactive development environment to be used in the unit will be the NetBeans IDE
- GUI construction can be partly done using drag and drop
  - But programmatic layout management will also be covered

Introduction to Swing components



First Name:

Last Name:

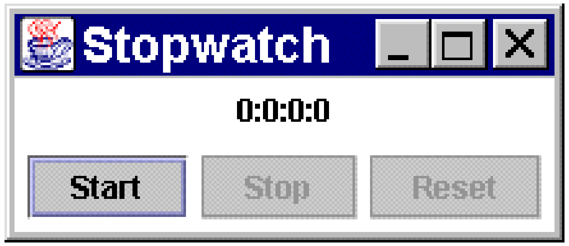
Telephone:

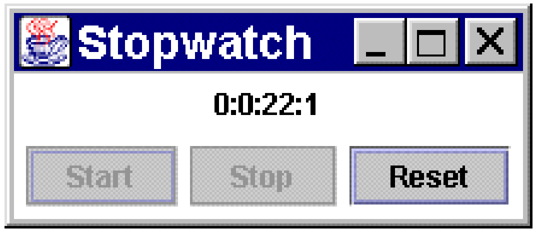
Find:

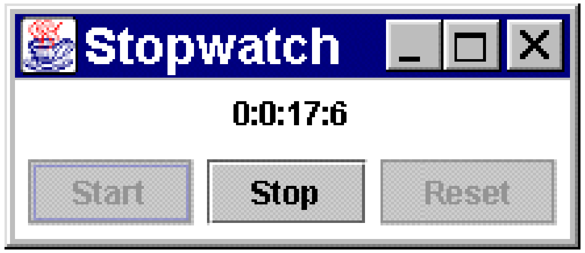
○ Edit ● Add ○ Browse Sort by: First name

Andy Ruddock
Bessie Smith
Billy Holliday
Carl Lewis
David Beckham
Groucho Marx
Harpo Marx
Justin Timberlake
Karl Marx
Lennox Lewis
Louis Armstrong

The Stopwatch artefact - visual appearance



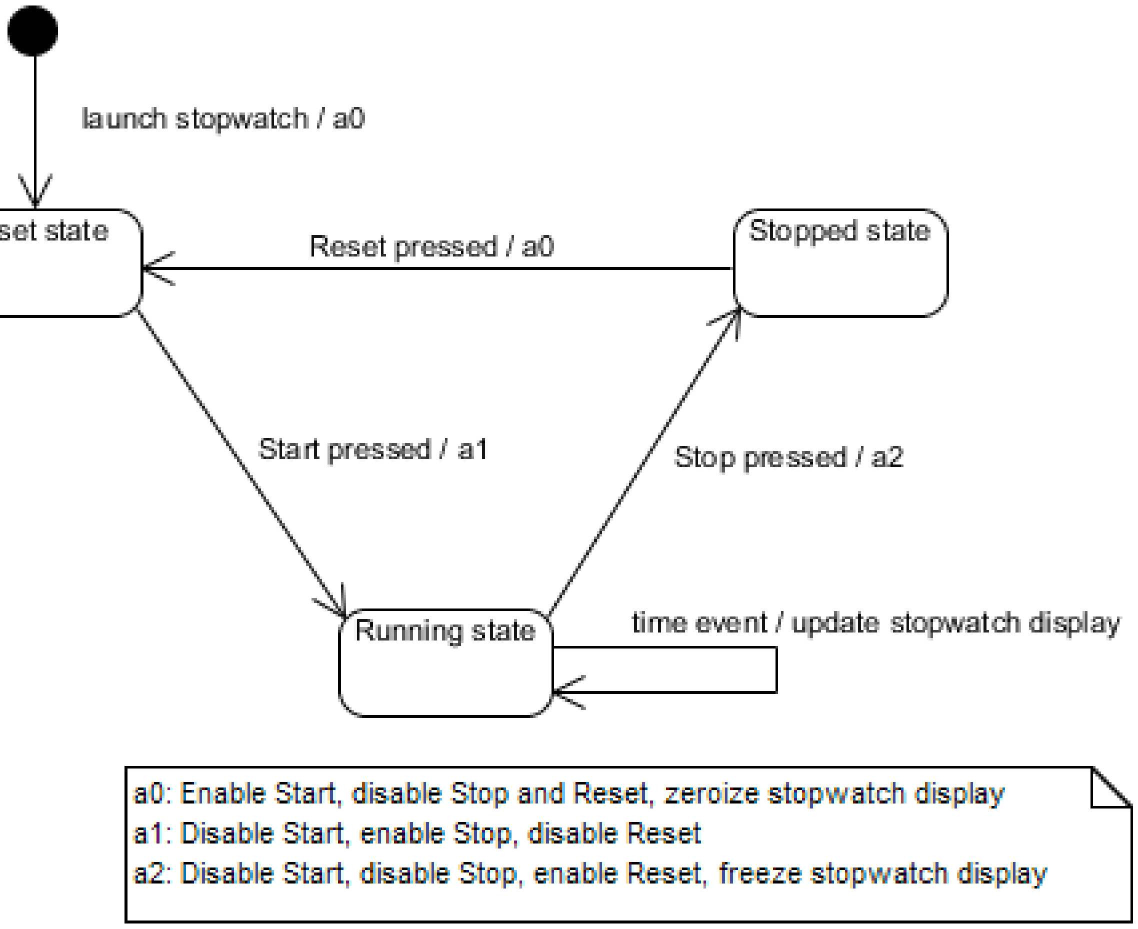




The Stopwatch artefact - visual appearance (cont'd)

- The illustrations on the previous slide show an instance of the JLabel being used in conjunction with three JButton instances to provide a simple stopwatch.
- The user initially sees the artefact in the state shown in the upper illustration, with the time zeroed and only the button labelled *Start* available.
- Pressing *Start* will cause the artefact to transit to the state shown in the lower left illustration, with the time display continually incrementing and only the *Stop* button available.
- Pressing *Stop* will cause the artefact to transit to the state shown in the lower right illustration, with the time display frozen and only the *Reset* button available.
- Pressing *Reset* will return the artefact to its initial state.

Stopwatch State Diagram



a0: Enable Start, disable Stop and Reset, zeroize stopwatch display
a1: Disable Start, enable Stop, disable Reset
a2: Disable Start, disable Stop, enable Reset, freeze stopwatch display

State Diagrams

- The State Diagram illustrates the behaviour expressed in natural language in a formalised graphical notation. As a semi-formal notation it is:
  - concise
  - less ambiguous
  - amenable to semi-formal analysis
  - non-idiosyncratic
- And can, as will be demonstrated, inform the engineering design and implementation.

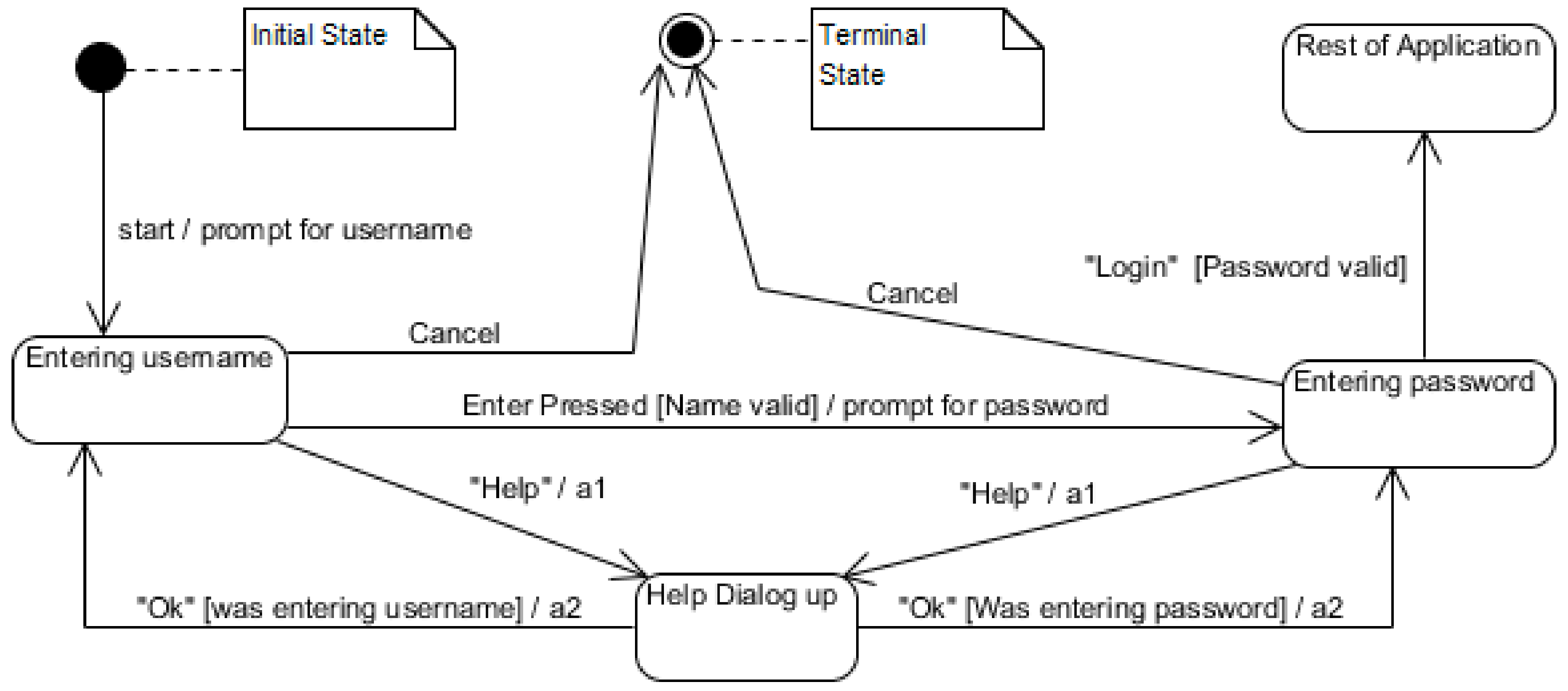
State Diagram Notation

- A State Diagram is a collection of *states* connected by *transitions*, starting with an *initial state* and possibly terminating with a *terminal state*. Simple states must be named.
- A transition represents moving from one state to another. It takes the form

  event[condition]/action   'action' sometimes called 'effect'
- The transition is only possible if the specified event occurs
- Assuming the event occurs, the transition is followed if
  - there is no condition

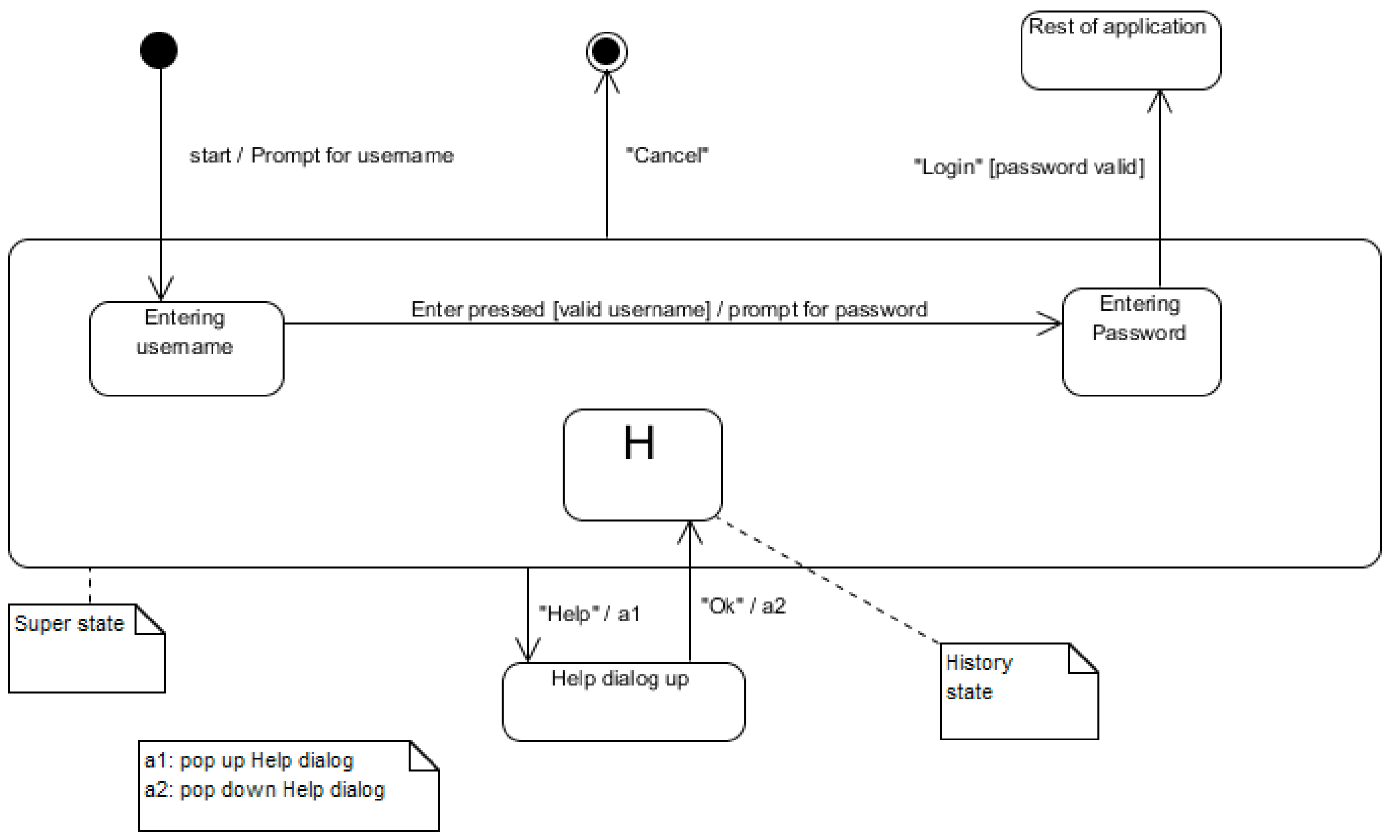
  or
  - there is a condition and it is true.
- If the transition is followed, and an action is specified, then the action will be carried out.

Another example



a1: Pop up Help Dialogue
a2: Pop down Help Dialogue

Superstate and history state



- This diagram is equivalent to the one on slide 15. A transition coming from a *superstate* is equivalent to it coming from any one of the contained states .
- Transitions returning to the *history state*, denoted by H, will go back to the contained state that the system was in when the superstate was exited.