M250 Overview

M250 Prsntn 2026J Overview

Contents

1 Agenda **2**

2 Adobe Connect **3**
- 2.1 Interface . . . 3
- 2.2 Settings . . . 3
- 2.3 Sharing Screen & Applications . . . 5
- 2.4 Ending a Meeting . . . 5
- 2.5 Invite Attendees . . . 5
- 2.6 Layouts . . . 6
- 2.7 Chat Pods . . . 7
- 2.8 Web Graphics . . . 7
- 2.9 Recordings . . . 8

3 M250 Overview **8**
- 3.1 Part 1 . . . 9
- 3.2 Part 2 . . . 9
- 3.3 Part 3 . . . 9
- 3.4 M250 Resources . . . 9
- 3.5 M250 Chapter Companions . . . 10
- 3.6 M250 Assessment . . . 11

4 Object-oriented Programming Terminology **11**
- 4.1 Sequence Diagrams . . . 11
- 4.2 Messages and Methods . . . 12

5 Using BlueJ **13**
- 5.1 Installing & Using BlueJ . . . 13
- 5.2 BlueJ Interface . . . 13
- 5.3 BlueJ Projects . . . 15

6 Software & Programming **15**
- 6.1 Learning Software Packages . . . 15
- 6.2 BlueJ — Package Learning Exercise . . . 16
- 6.3 Writing Programs & Thinking . . . 18

7 What Next ? **19**

8 References **20**
- 8.1 Java Documentation . . . 20
- 8.2 Books Phil Likes . . . 21
- References . . . 22

1 Agenda

- Introductions
- *Adobe Connect* — if you or I get cut off, wait till we reconnect (or send you an email)
- M250 overview
- Software resources & learning software packages
- Data types & variables
- Some useful Web & other references
- Time: about 1 hour
- Do ask questions or raise points.
- Slides/Notes M250Tutorial20261011OverviewPrsntn2026J

Introductions — Phil

- *Name* Phil Molyneux
- *Background*
  - Undergraduate: Physics and Maths (Sussex)
  - Postgraduate: Physics (Sussex), Operational Research (Brunel), Computer Science (University College, London)
  - Worked in Operational Research, Business IT, Web technologies, Functional Programming
- *First programming languages* Fortran, BASIC, Pascal
- *Favourite Software*
  - Haskell — pure functional programming language
  - Text editors TextMate, Sublime Text — previously Emacs
  - Word processing in LaTeX — all these slides and notes
  - Mac OS X
- *Learning style* — I read the manual before using the software

Introductions — You

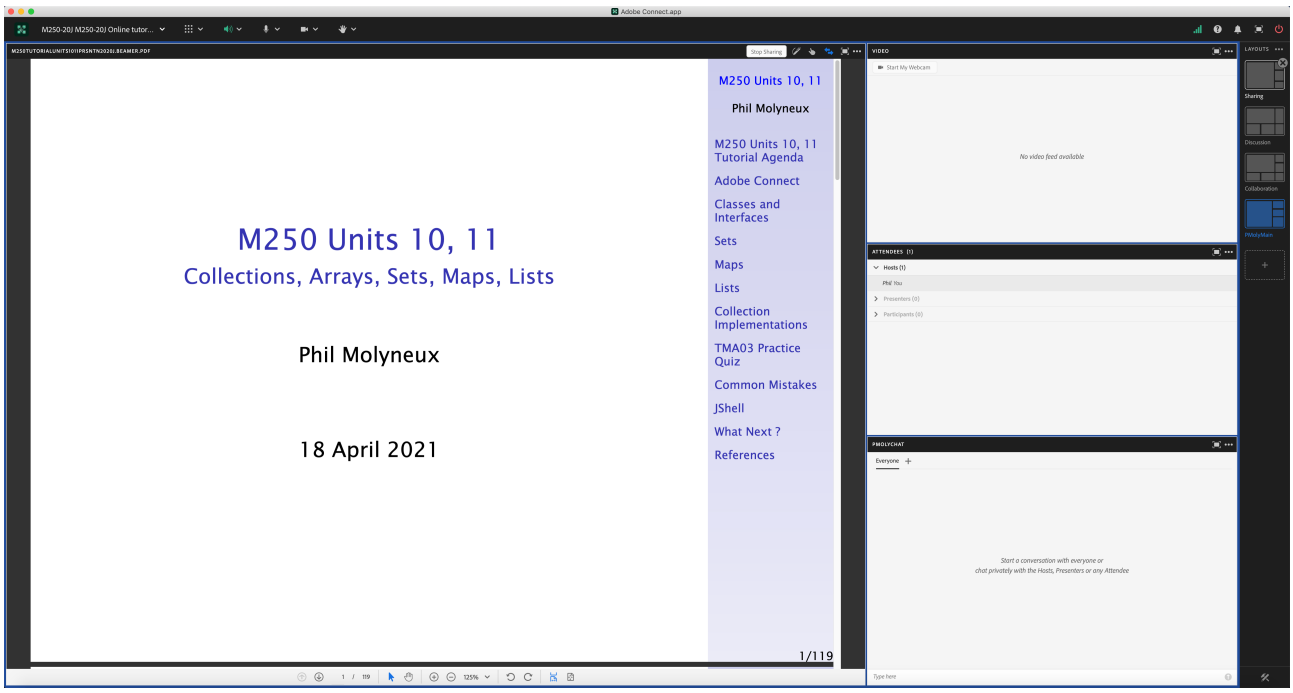
- *Name* ?
- *Favourite software/Programming language* ?
- *Favourite text editor or integrated development environment (IDE)*
- List of text editors, Comparison of text editors and Comparison of integrated development environments
- *Other OU courses* ?
- *Anything else* ?

ToC

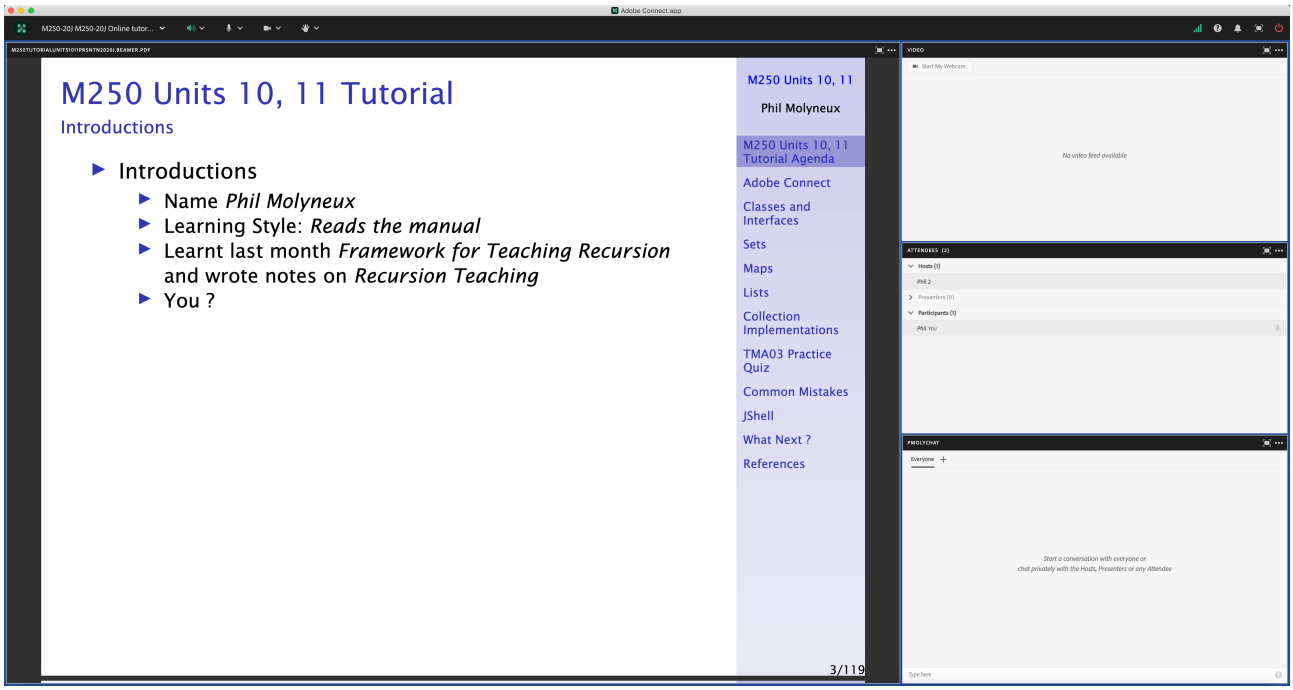
2 Adobe Connect Interface and Settings

2.1 Adobe Connect Interface

Adobe Connect Interface — Host View



Adobe Connect Interface — Participant View



2.2 Adobe Connect Settings

Adobe Connect — Settings

- **Everybody** *Menu bar* › *Meeting* › Speaker & Microphone Setup
- *Menu bar* › *Microphone* › Allow Participants to Use Microphone ✔

- Check Participants see the entire slide including slide numbers bottom right **Workaround**
  - *Disable Draw* *Share pod* > *Menu bar* > *Draw icon*
  - *Fit Width* *Share pod* > *Bottom bar* > *Fit Width icon* ✔
- *Meeting* > Preferences > General > **Host Cursor** > Show to all attendees
- *Menu bar* > *Video* > Enable **Webcam** for Participants ✔
- Do not *Enable single speaker mode*
- Cancel hand tool
- Do not enable green pointer
- **Recording** Meeting > Record Session ✔
- **Documents** Upload PDF with drag and drop to share pod
- Delete *Meeting* > Manage Meeting Information > Uploaded Content and *check filename* > *click on delete*

Adobe Connect — Access

- **Tutor Access**

  TutorHome > M269 Website > Tutorials

  Cluster Tutorials > M269 Online tutorial room

  Tutor Groups > M269 Online tutor group room

  Module-wide Tutorials > M269 Online module-wide room
- **Attendance**

  TutorHome > Students > View your tutorial timetables
- **Beamer Slide Scaling** 440% (422 x 563 mm)
- **Clear Everyone's Status**

  Attendee Pod > Menu > Clear Everyone's Status
- **Grant Access** and send link via email

  Meeting > Manage Access & Entry > Invite Participants...
- **Presenter Only Area**

  Meeting > Enable/Disable Presenter Only Area

Adobe Connect — Keystroke Shortcuts

- Keyboard shortcuts in Adobe Connect
- **Toggle Mic** ⌘+M (Mac), Ctrl+M (Win) (On/Disconnect)
- **Toggle Raise-Hand status** ⌘+E
- **Close dialog box** ⎋ (Mac), Esc (Win)
- **End meeting** ⌘+\

2.3 Adobe Connect — Sharing Screen & Applications

- Share My Screen > Application tab > Terminal for Terminal
- Share menu > Change View > Zoom in for mismatch of screen size/resolution (Participants)
- (Presenter) Change to 75% and back to 100% (solves participants with smaller screen image overlap)
- Leave the application on the original display
- Beware blued hatched rectangles — from other (hidden) windows or contextual menus
- Presenter screen pointer affects viewer display — beware of moving the pointer away from the application
- First time: System Preferences > Security & Privacy > Privacy > Accessibility

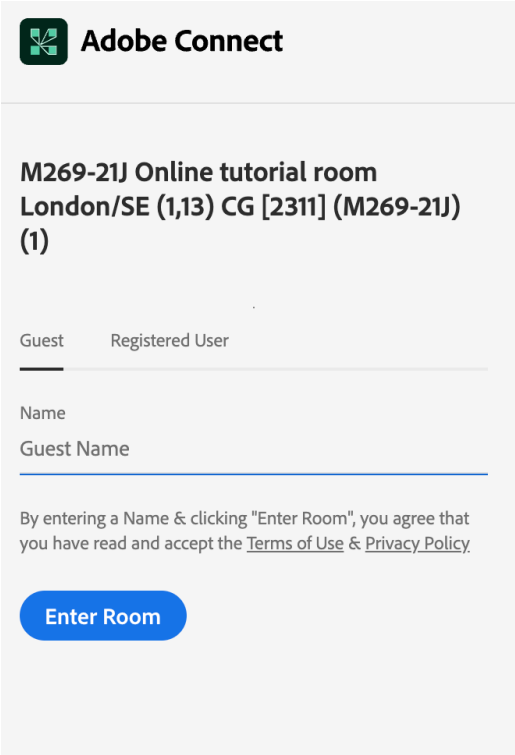
2.4 Adobe Connect — Ending a Meeting

- *Notes for the tutor only*
- **Student:** Meeting > Exit Adobe Connect
- **Tutor:**
- **Recording** Meeting > Stop Recording ✔
- **Remove Participants** Meeting > End Meeting... ✔
  - Dialog box allows for message with default message:
  - *The host has ended this meeting. Thank you for attending.*
- **Recording availability** *In course Web site for joining the room, click on the eye icon in the list of recordings under your recording* — edit description and name
- **Meeting Information** Meeting > Manage Meeting Information — can access a range of information in Web page.
- **Delete File Upload** Meeting > Manage Meeting Information > Uploaded Content tab select file(s) and click Delete
- **Attendance Report** see course Web site for joining room

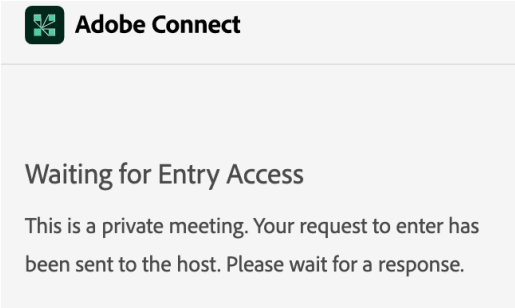
2.5 Adobe Connect — Invite Attendees

- **Provide Meeting URL** Menu > Meeting > Manage Access & Entry > Invite Participants...
- **Allow Access without Dialog** Menu > Meeting > Manage Meeting Information provides new browser window with *Meeting Information* Tab bar > Edit Information
- Check *Anyone who has the URL for the meeting can enter the room*
- Default *Only registered users and accepted guests may enter the room*
- **Reverts to default next session but URL is fixed**
- Guests have blue icon top, registered participants have yellow icon top — same icon if URL is open

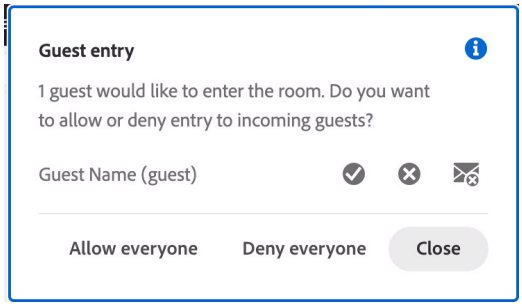
- See Start, attend, and manage Adobe Connect meetings and sessions
- Click on the link sent in email from the Host
- Get the following on a Web page
- As *Guest* enter your name and click on Enter Room



- See the *Waiting for Entry Access* for *Host* to give permission



- *Host* sees the following dialog in *Adobe Connect* and grants access



2.6 Layouts

- **Creating new layouts** example *Sharing* layout

- Menu > Layouts > Create New Layout... Create a New Layout dialog > Create a new blank layout and name it *PMolyMain*
- New layout has no Pods but does have Layouts Bar open (see Layouts menu)
- **Pods**
- Menu > Pods > Share > Add New Share and resize/position — initial name is *Share n* — rename *PMolyShare*
- **Rename Pod** Menu > Pods > Manage Pods... Manage Pods > Select > Rename or Double-click & rename
- Add Video pod and resize/reposition
- Add Attendance pod and resize/reposition
- Add Chat pod — rename it *PMolyChat* — and resize/reposition
- Dimensions of **Sharing** layout (on 27-inch iMac)
  - Width of Video, Attendees, Chat column 14 cm
  - Height of Video pod 9 cm
  - Height of Attendees pod 12 cm
  - Height of Chat pod 8 cm
- **Duplicating Layouts** does *not* give new instances of the Pods and is probably not a good idea (apart from local use to avoid delay in reloading Pods)
- **Auxiliary Layouts** name *PMolyAux0n*
  - Create new Share pod
  - Use existing Chat pod
  - Use same Video and Attendance pods

2.7 Chat Pods

- **Format Chat text**
- Chat Pod > menu icon > My Chat Color
- Choices: Red, Orange, Green, Brown, Purple, Pink, Blue, Black
- Note: Color reverts to Black if you switch layouts
- Chat Pod > menu icon > Show Timestamps

2.8 Graphics Conversion for Web

- Conversion of the screen snaps for the installation of Anaconda on 1 May 2020
- Using GraphicConverter 11
- File > Convert & Modify > Conversion > Convert
- Select files to convert and destination folder
- Click on Start selected Function or ⌘+↵

2.9 Adobe Connect Recordings

- *Menu bar* > *Meeting* > Preferences > Video
- Aspect ratio > Standard (4:3) (not Wide screen (16:9) default)
- Video quality > Full HD (1080p not High default 480p)
- **Recording** *Menu bar* > *Meeting* > Record Session ✔
- **Export Recording**
- *Menu bar* > *Meeting* > Manage Meeting Information
- *New window* > Recordings > *check Tutorial* > Access Type button
- check Public > check Allow viewers to download
- **Download Recording**
- *New window* > Recordings > *check Tutorial* > Actions > Download File

ToC

3 M250 Overview

- Object-oriented programming
- Encapsulation enforces modularity — implementations of objects not available outside some *container*
- Inheritance encourages reuse of code and organises complexity
- Polymorphism enables a structured approach to different objects responding differently to the same messages
- Java is a class-based, object-oriented general-purpose programming language with strong type system
- Type system addresses the problem of ensuring that programs have meaning — we shall look at this later but consider

  1 + 2 and 1 + '2' are they valid ?
- We are using Java to illustrate an object-oriented approach to programming
- We will not have complete coverage of every feature of Java
- Evaluate `1 + 2`

```
jshell> 1 + 2
$1 ==> 3
```

- Evaluate `1 + '2'`

```
jshell> 1 + '2'
$2 ==> 51
```

- The evaluations are using JShell a Read-Eval-Print Loop (REPL) tool for exploring Java
- What is going on here? A character is a *numeric* type

```
jshell> (int) '2'
$3 ==> 50
jshell> (char) 50
$4 ==> '2'
```

- The above are explicit *type conversions*
- Beware mixing characters and strings

```
jshell> '1' + '2'
$5 ==> 99

jshell> "1" + "2"
$6 ==> "12"
```

- The former does implicit type casting to `int`
- The latter is string concatenation
- The `(+)` operator is *overloaded*

3.1 M250 Chps 1 to 4

- Chp 1: Objects and classes
- Chp 2: Understanding class definitions
- Chp 3: Object interactions
- Chp 4: Grouping objects
- Chp 5: Functional processing of collections

3.2 M250 Chps 6 and 7

- Chp 6: More sophisticated behaviour
- Chp 7: Fixed-size collections — arrays
- Programming outside BlueJ

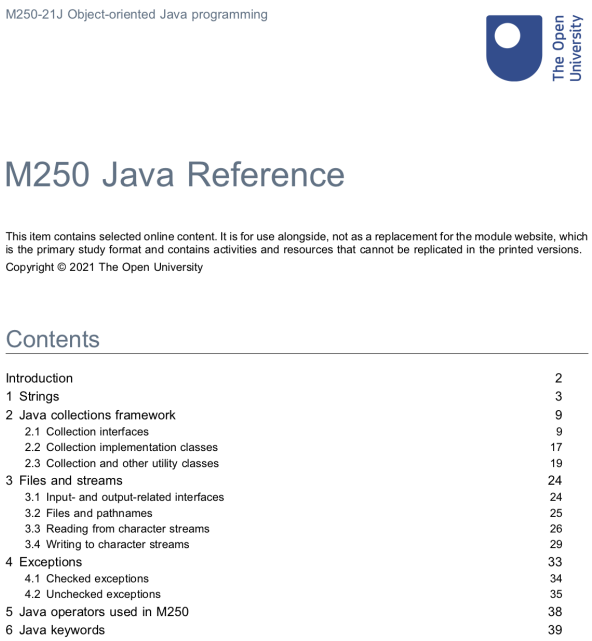
3.3 M250 Chps 9 to 12, and 14

- Chp 8: Designing classes and code conventions
- Chp 9: Well-behaved object and debugging
- Chp 10: Improving structure with inheritance
- Chp 11: More about inheritance
- Chp 12: Further abstraction techniques — Interfaces
- Chp 14: Handling errors

3.4 M250 Resources

- Module Guide (online)
- Software Guide (online)

- M250 BlueJ Activities
- M250 Java Reference
- Chapter companions — reading guides
- Exam Handbook (print/online under Assessment)
- Study calendar (HTML/PDF)
- Programming outside BlueJ (optional) (online)



M250-21J Object-oriented Java programming

The Open University

M250 Java Reference

This item contains selected online content. It is for use alongside, not as a replacement for the module website, which is the primary study format and contains activities and resources that cannot be replicated in the printed versions.

Copyright © 2021 The Open University

Contents

Introduction 2
1 Strings 3
2 Java collections framework 9
2.1 Collection interfaces 9
2.2 Collection implementation classes 17
2.3 Collection and other utility classes 19
3 Files and streams 24
3.1 Input- and output-related interfaces 24
3.2 Files and pathnames 25
3.3 Reading from character streams 26
3.4 Writing to character streams 29
4 Exceptions 33
4.1 Checked exceptions 34
4.2 Unchecked exceptions 35
5 Java operators used in M250 38
6 Java keywords 39

3.5 M250 Chapter Companions — Extra Teaching

- Key areas going beyond the textbook:
- **Chp 1 Prequel** VMs; Java history; procedural/OO programming
- **Chp 3** Additional discussion on calling methods and bounding boxes of graphical objects
- **Chp 6** Iterating over a map; maps with sets as values
- **Chp 7** Arrays holding references to objects
- **Programming outside BlueJ** supplement (not assessed)
- **Chp 8** M250 Code conventions and design
- **Chp 9** More on static and dynamic code; different kinds of errors, debugging; formatted printing (printf)
- **Chp 10** Inheritance; constructors; members

- **Chp 11** More on hash codes, equals, and `Objects.hash` method; `getClass` compared to `instanceof`; immutability of keys
- **Chp 12** The `Comparable` interface

3.6 M250 Assessment

- 3 TMAs
  - TMA01 15% Chps 1 to 4
  - TMA02 15% Chps 6 and 7
  - TMA03 20% Chps 9 to 12, and 14
  - Exam 50%
- Final exam in June
- To pass, must achieve at least 40% in both OCAS (Overall Continuous Assessment Score) and 30% OES (Overall Examinable Score)

ToC

4 Object-oriented Programming Terminology

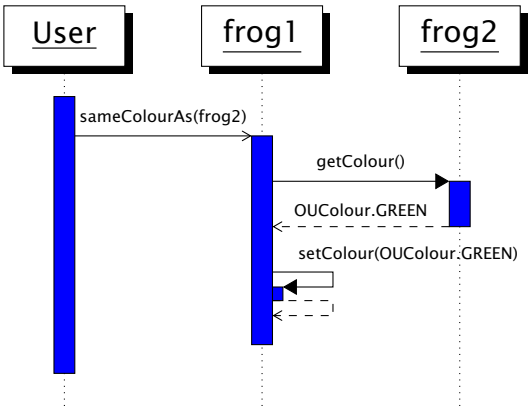
4.1 Sequence Diagrams

- From M250 version to 2020J Unit 2
- Example code 1

```
public void sameColourAs (Frog aFrog) {
   this.setColour(aFrog.getColour()) ;
}
```

```
jshell> frog1.sameColourAs(frog2)
```

- The `JShell` prompt is just indicative of code usage — usually would be graphical interface here



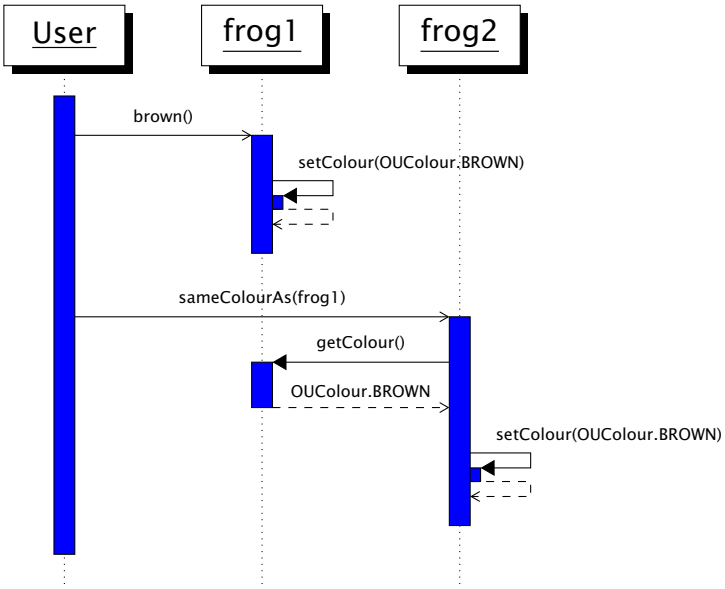
- The diagrams are similar to the UML (Unified Modelling Language) diagrams
- See also Unified Modelling Language
- See Unit 2 Section 5.3 *Collaborating objects*, page 67

- From M250 version to 2020J Unit 2
- Example code (2)

```
public void sameColourAs (Frog aFrog) {
   this.setColour(aFrog.getColour()) ;
}

public void brown()  {
   this.setColour(OUColour.BROWN) ;
}
```

```
jshell> frog1.brown()
jshell> frog2.sameColourAs(frog1)
```



4.2 Messages and Methods

- Object-oriented view: programs work using *objects* which communicate by sending *messages*

$$\underbrace{\overbrace{\texttt{objectName}}^{\text{receiver}}\texttt{.}\overbrace{\underbrace{\texttt{methodName}}_{\text{message name}}\texttt{(arg1,arg2,...)}}^{\text{message}}}_{\text{message-send}}$$

- Objects have *state* (implemented by *attributes*) and *behaviour*, its response to messages (implemented by the *protocol*: methods implementing messages)
- A message must correspond to the signature of one of the methods in the protocol
- Object terminology is in *M250 Glossary* — this is an object-oriented view of programs
- Note: the terminology is not standard across all texts — see, for example, Glossary of Java and Related Terms from Barnes (2000) (author of BlueJ texts)

ToC

5 Using BlueJ

5.1 Installing & Using BlueJ

- M250 Website > Resources > Software resources and follow the instructions
- Note: macOS users will have to disable *Gatekeeper* see Disable Gatekeeper — in Terminal say:

```
sudo spctl --master-disable
```

- On launch, BlueJ looks for **bluej.defs** for its settings — in macOS this should be at `~/M250/BlueJ/BlueJ.app/Contents/Resources/Java/bluej.defs`
- You can have spaces in file or folder names but that will make life painful, so don't
- See discussion forums for various problems and how to solve them

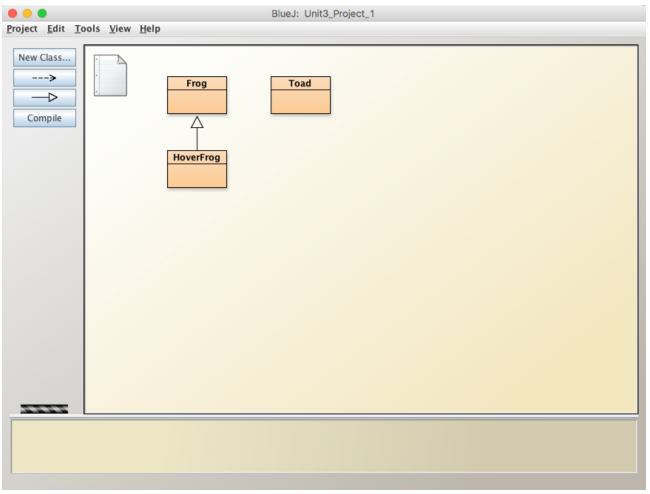
- M250 Website > Resources > Software resources and follow the instructions
- Note: macOS users will have to disable *Gatekeeper* see Disable Gatekeeper — in Terminal say:

```
sudo spctl --master-disable
```


- Install BlueJ version 4.1.4 from BlueJ Version History — (27 September 2021) site down!!

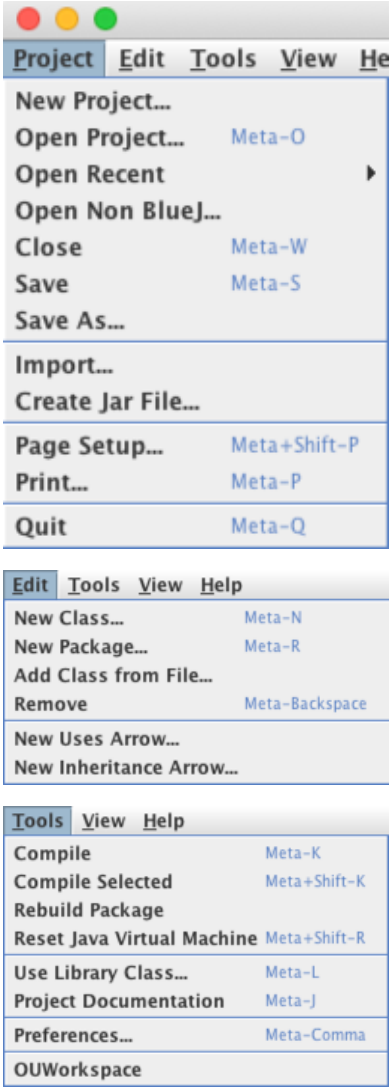
- You can have spaces in file or folder names but that will make life painful, so don't
- See discussion forums for various problems and how to solve them

5.2 BlueJ Interface

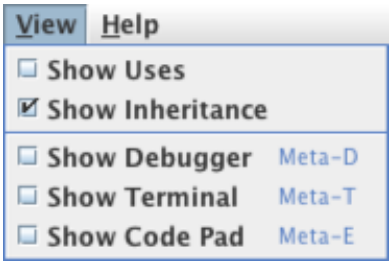


- BlueJ window for Unit 3 Project 1
- Name the parts ?

BlueJ Interface: Menus



- Note the **OUWorkspace**



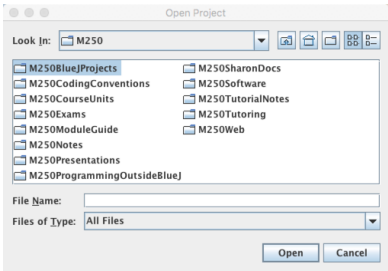
- The **Code Pad** takes part of the **Object Bench**

- The **OU Class Library** will try to launch a Web browser

- The documentation is in oudocs *near* the BlueJ app
- In Phil's case: (but yours will be somewhere else)
- file:///Users/molyneux/MyData/Documents/OU/Courses/Computing/M250/M250Software/M250Working/oudocs/index.html

ToC

5.3 BlueJ Projects



- **Open a project**
- To navigate into a folder
- Select folder and
- Double-click (Windows) or Return ↵ or
- Enter ⌅ (Mac)
- Return ↵ (Mac) means Open
- **New Project**
- Return ↵ or Enter ⌅ (Mac) both mean Create (which is wrong)

ToC

6 Learning Software Packages & Writing Programs

6.1 Learning Software Packages

1. Where is the package source ?
2. What version are you using ?
3. What documentation is available ?
4. What are the *names* for the parts of the interface ?
5. How do you leave the package ? How do you enter the package ?
6. Is there any on-line help and, if so, how is it used ?
7. Are there any initialisation files, configuration or preferences and how are they used ?
8. How do you import and export data from the package ?
9. *When all else fails,* how can you obtain advice ?

ToC

6.2 BlueJ — Package Learning Exercise

- Answer the *Key Questions* for BlueJ
- Where is the package source ?
- What version are you using ?
- Where is the package source ?

  BlueJ BlueJ Web site
- What version are you using ?

  4.1.4
- What version of Java SDK does that use ?

  Java 8
- Answer the *Key Questions* for BlueJ
- What documentation is available ?
- Answer the *Key Questions* for BlueJ
- What documentation is available ?
- What Java documentation is available ?

  Java Documentation

  JDK 17 Documentation

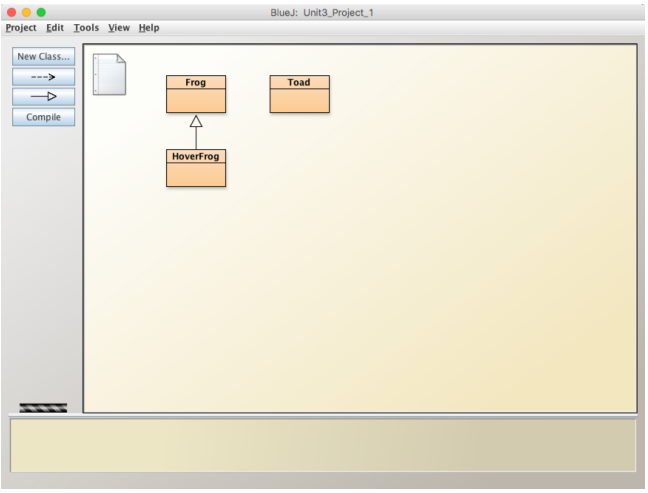
  Java Platform, Standard Edition Documentation — see the documentation for JDK 8
- Which was the first version to have JShell ?

  Java 9
- Answer the *Key Questions* for BlueJ
- What are the *names* for the parts of the interface ?
- Answer the *Key Questions* for BlueJ
- What are the *names* for the parts of the interface ?

  See diagram in BlueJ documentation

  Main window

  Object bench
- Where is the *Code Pad* ?
- How do you create new objects or inspect existing ones ?
- How do you execute methods of an object ?
- How do you edit the source code of a class ?
- Answer the *Key Questions* for BlueJ
- What are the *names* for the parts of the interface ?



- Answer the *Key Questions* for BlueJ
- What are the *names* for the parts of the interface ?

  See diagram in BlueJ documentation

  Main window

  Object bench
- Where is the *Code Pad* ? View > Show Code Pad
- How do you create new objects or inspect existing ones ? *Right-Click on Class* in main window
- How do you execute methods of an object ? *Right-Click on Object* in Object bench
- How do you edit the source code of a class ? *Right-Click on Class* and Open Editor
- Answer the *Key Questions* for BlueJ
- How do you leave the package ? How do you enter the package ?
- Answer the *Key Questions* for BlueJ
- How do you leave the package ? How do you enter the package ?
- **Enter**
- Load a project Project > Open Project...
- How do you use a project from the command line with JShell ?
- **Leave**
- Answer the *Key Questions* for BlueJ
- Is there any on-line help and, if so, how is it used ?
- Answer the *Key Questions* for BlueJ
- Is there any on-line help and, if so, how is it used ?

  Help > BlueJ Tutorial

  Help > BlueJ Web Site

  Tools > Project Documentation

Help > Java Class Libraries to view Java API Specification

but you will have to read the documentation

- Answer the *Key Questions* for BlueJ
- Are there any initialisation files, configuration or preferences and how are they used ?
- Answer the *Key Questions* for BlueJ
- Are there any initialisation files, configuration or preferences and how are they used ?

  Tools > Preferences...

  BlueJ.defs

```
##    <USER_HOME>/.bluej/bluej.properties                    (Unix)
##    C:\Winnt\profiles\<USER_NAME>\Bluej\Bluej.properties    (WinNT)
##    C:\<JDK_HOME>\Bluej\Bluej.properties                    (Win9x)
```

- Notice SaaSHub: Is BlueJ down ?
- Answer the *Key Questions* for BlueJ
- How do you import and export data from the package ?
- Answer the *Key Questions* for BlueJ
- How do you import and export data from the package ?
- **Export**
- Using code outside BlueJ
- **Import**
- Using code developed in another editor or IDE
- Answer the *Key Questions* for BlueJ
- *When all else fails,* how can you obtain advice ?
- Answer the *Key Questions* for BlueJ
- *When all else fails,* how can you obtain advice ?

  M250 Forums

  BlueJ Web site

  StackOverflow: Questions tagged [bluej]

ToC

6.3 Writing Programs & Thinking

The Steps

1. Invent a *name* for the program (or function)
2. What is the *type* of the function ? What sort of *input* does it take and what sort of *output* does it produce ? In Python a type is implicit; in other languages such as Haskell a type signature can be explicit.

3. Invent *names* for the input(s) to the function (*formal parameters*) — this can involve thinking about possible *patterns* or *data structures*
4. What restrictions are there on the input — state the preconditions.
5. What must be true of the output — state the postconditions.
6. *Think* of the definition of the function body.

The *Think* Step

- **How to Think**
  1. Think of an example or two — what should the program/function do ?
  2. Break the inputs into separate cases.
  3. Deal with simple cases.
  4. Think about the result — try your examples again.
- **Thinking Strategies**
  1. Don't think too much at one go — break the problem down. Top down design, step-wise refinement.
  2. What are the inputs — describe all the cases.
  3. Investigate choices. What data structures ? What algorithms ?
  4. Use common tools — bottom up synthesis.
  5. Spot common function application patterns — generalise & then specialise.
  6. Look for good *glue* — to combine functions together.

ToC

7 What Next ?

Programming, Debugging, Psychology

Although programming techniques have improved immensely since the early days, the process of finding and correcting errors in programming — known graphically if inelegantly as *debugging* — still remains a most difficult, confused and unsatisfactory operation. The chief impact of this state of affairs is psychological. Although we are happy to pay lip-service to the adage that to err is human, most of us like to make a small private reservation about our own performance on special occasions when we really try. It is somewhat deflating to be shown publicly and incontrovertibly by a machine that even when we do try, we in fact make just as many mistakes as other people. If your pride cannot recover from this blow, you will never make a programmer.

Christopher Strachey, Scientific American 1966 vol 215 (3) September pp112–124

- To err is human, to really foul things up requires a computer.
- Attributed to Paul R. Ehrlich in 101 Great Programming Quotes
- Attributed to Bill Vaughn in Quote Investigator

- Derived from Alexander Pope (1711, An Essay on Criticism)
- *To Err is Humane; to Forgive, Divine*
- This also contains

  A little learning is a dangerous thing;

  Drink deep, or taste not the Pierian Spring
- In programming, this means you have to *read the fabulous manual* (RTFM)

Chps 1–4, TMA01

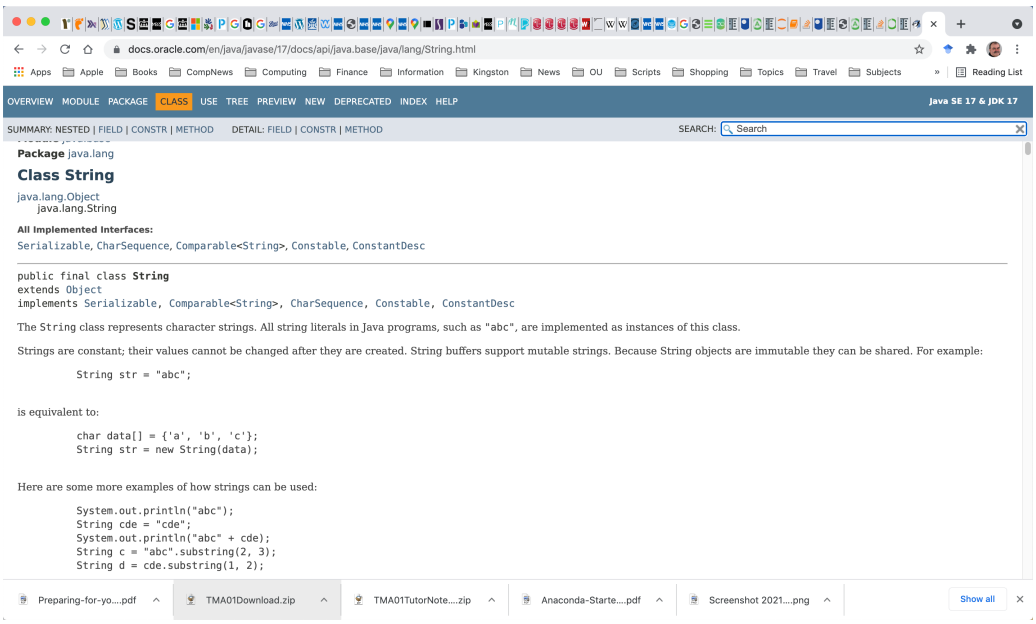
- Chps 1–3, Data types and expressions; Class definitions
- Tutorial 10:00 Sunday 9 November 2025 online
- Chps 4–5, Iteration, collections; Functional Java (optional)
- Tutorial 10:00 Sunday 16 November 2025 online
- TMA01 Thursday 11 December 2025

ToC

8 Web Links & References

8.1 Java Documentation

- Java Documentation — BlueJ has JDK 7 embedded, JDK 13 is current (2019)
- JDK 13 Documentation
- Java Platform API Specification
- Java Language Specification
- JDK Documentation > API Documentation > java.base
  - java.lang — fundamental classes for the Java programming language
  - java.util — Collections framework



docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html

OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP Java SE 17 & JDK 17

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package java.lang

Class String

java.lang.Object
java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>, Constable, ConstantDesc

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc
```

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

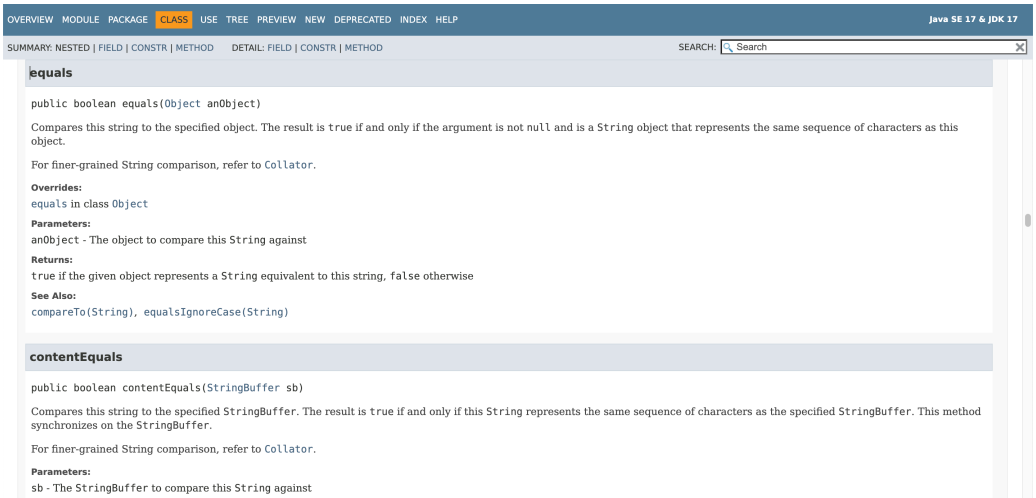
is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2, 3);
String d = cde.substring(1, 2);
```

- `Strings` are *immutable* objects
- See java.lang.StringBuilder for *mutable* strings
- In a *functional programming approach* everything is immutable — it makes life simpler (but at a cost)



OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP Java SE 17 & JDK 17

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

equals

```
public boolean equals(Object anObject)
```

Compares this string to the specified object. The result is true if and only if the argument is not null and is a String object that represents the same sequence of characters as this object.

For finer-grained String comparison, refer to Collator.

Overrides:
equals in class Object

Parameters:
anObject - The object to compare this String against

Returns:
true if the given object represents a String equivalent to this string, false otherwise

See Also:
compareTo(String), equalsIgnoreCase(String)

contentEquals

```
public boolean contentEquals(StringBuffer sb)
```

Compares this string to the specified StringBuffer. The result is true if and only if this String represents the same sequence of characters as the specified StringBuffer. This method synchronizes on the StringBuffer.

For finer-grained String comparison, refer to Collator.

Parameters:
sb - The StringBuffer to compare this String against

- Remember (==) tests for *identity* — what does this mean ?

ToC

8.2 Books Phil Likes

- M250 is self contained — you do not need further books — but you might like to know about some:
- Sestoft (2016) — the best short reference
- Evans et al. (2023) — the best longer reference
- Lutz (2025) *Learning Python* — a leading Python book with a different view on Python type annotations

- Barnes and Kölling (2016) — the BlueJ book — see www.bluej.org for documentation and tutorial
- Bloch (2017) — guide to best practice

ToC

References

Barnes, David (2000). *Object-Oriented Programming with Java: An Introduction*. Prentice Hall. ISBN 0130869007. 12

Barnes, David J. and Michael Kolling (2009). *Objects First with Java*. Pearson Education, fourth edition. ISBN 0-13-606086-2. URL http://www.bluej.org/objects-first/.

Barnes, David J. and Michael Kolling (2011). *Objects First with Java*. Pearson Education, fifth edition. ISBN 0132835541. URL http://www.bluej.org/objects-first/.

Barnes, David J. and Michael Kölling (2016). *Objects First with Java*. Pearson, sixth edition. ISBN 1292159049. URL http://www.bluej.org/objects-first/. 22

Bloch, Joshua (2017). *Effective Java*. Addison-Wesley Professional, third edition. ISBN 9780134685991. 22

Darwin, Ian F (2014). *Java Cookbook*. O'Reilly, third edition. ISBN 9781449337049.

Evans, Benjamin J.; Jason Clark; and David Flanagan (2023). *Java in a Nutshell*. O'Reilly Media, eighth edition. ISBN 1098131002. 21

Evans, Benjamin J and David Flanagan (2014). *Java In A Nutshell*. O'Reilly, sixth edition. ISBN 1449370829. URL https://github.com/kittylyst/javanut6-examples.

Evans, Benjamin J and David Flanagan (2018). *Java In A Nutshell*. O'Reilly, seventh edition. ISBN 1492037257.

Felleisen, Matthias and Daniel P. Friedman (1998). *A Little Java, A Few Patterns*. MIT Press. ISBN 0262561158. URL http://felleisen.org/matthias/BALJ-index.html.

Gosling, James; Bill Joy; Guy L. Steele Jr.; Gilad Bracha; and Alex Buckley (2014). *The Java Language Specification, Java SE 8 Edition (Java Series) (Java (Addison-Wesley))*. Addison Wesley, eighth edition. ISBN 013390069X. URL https://docs.oracle.com/en/java/javase/12/index.html.

Lutz, Mark (2025). *Learning Python*. O'Reilly Media, sixth edition. ISBN 1098171306. 21

Naftalin, Maurice and Philip Wadler (2006). *Java Generics and Collections*. O'Reilly. ISBN 059610247X.

Sestoft, Peter (2002). *Java Precisely*. MIT Press. ISBN 0-262-69276-7.

Sestoft, Peter (2005). *Java Precisely*. MIT, second edition. ISBN 0262693259.

Sestoft, Peter (2016). *Java Precisely*. MIT, third edition. ISBN 0262529076. URL http://www.itu.dk/people/sestoft/javaprecisely/. 21

Waldo, Jim (2010). *Java: The Good Parts*. O'Reilly. ISBN 9780596803735. URL http://shop.oreilly.com/product/9780596803742.do.

ToC

Author Phil Molyneux Written 11 October 2026 *Printed 9th October 2026*
Subject dir: ⟨*baseURL*⟩/OU/Courses/Computing/M250/M250Presentations/M250Prsntn2026J
Topic path:
/M250Prsntn2026JTutorials/M250Tutorial20261011OverviewPrsntn2026J/M250Tutorial20261011OverviewPrsntn2026J.pdf