

Practical Session Briefing

COMSM0086

Dr Simon Lock and Dr Sion Hannuna

Overview

Last week you encountered Maven for the first time

For each exercise we'll provide a Maven template

Avoids you having to do all the boring setup work

Means you can get on and focus on programming

There are features of Maven we haven't covered yet

Worth spending time exploring them in more detail

Maven Recap

Maven is a cross-language build environment
A bit like 'make', but a LOT more sophisticated

Maven is very useful for managing dependencies
Not only for **specifying** required libraries...
...but also for **installing** them

You may have noticed a lot of text scrolling past ?
(especially the first time you ran 'mvnw compile')
That was Maven installing various libs and plugins


Maven in More Detail


Core to Maven is the 'Project Object Model' file:
POM


This describes a project and its dependencies

Most IDEs support Maven POM files - including IntelliJ !
So you can open/import a Maven project seamlessly

Maven also defines various standards & conventions
Including structure and content of the project folder...

 mvnw

 mvnw.cmd


 pom.xml


▼  src


▼  main

▼  java


▼  edu

▼  uob


 Circle.java


 Colour.java

 MultiVaria...

 Rectangle.ja


 Shapes.java

 Triangle.jav

 TriangleVari

 TwoDime...l

▶  test

▶  target

Testing and Reporting

There is more to Maven than compiling & running
It supports various other development activities:

Code Analysis, Unit & Integration Testing, Reporting

In this unit we make extensive use of testing tools
Test Driven Development (TDD) is a key activity
So much so that it is touched on in all three units !

We'll revisit this topic in detail later in this briefing

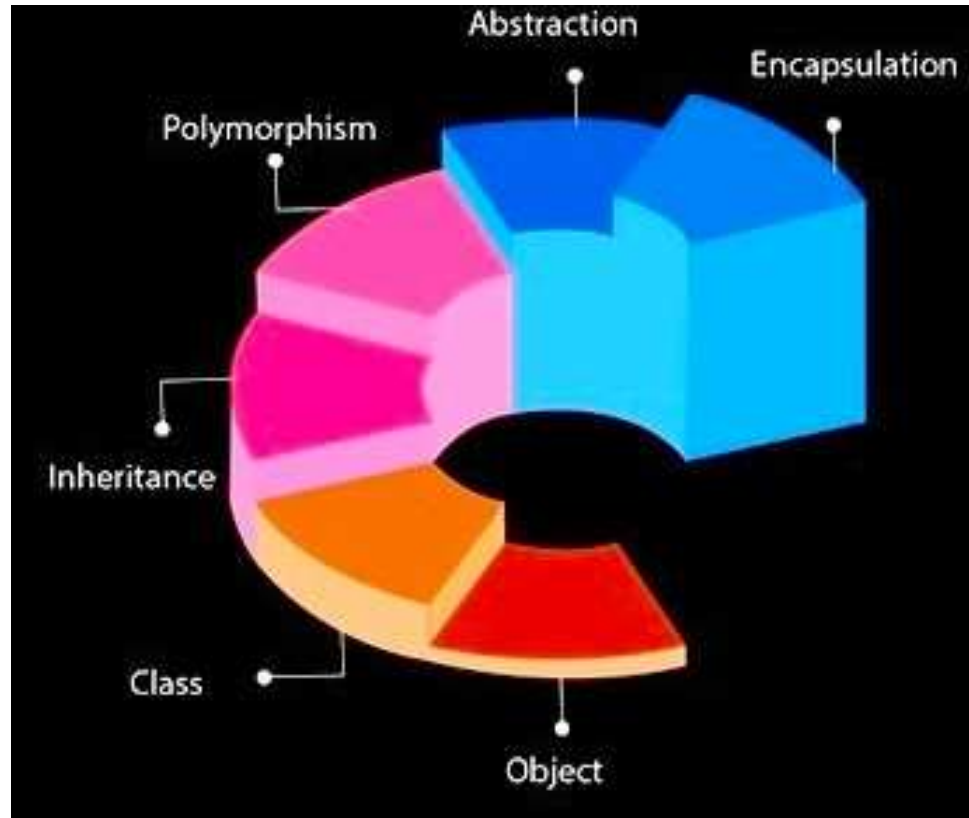
This Week's Workbook

But now let's take a look at this week's workbook
It's a fairly easy and straightforward set of tasks
It's all based around various 2 dimensional shapes
The main shape of interest will be the Triangle !

You may have attempted some of the tasks already
This won't always be the case for future workbooks
Unit starts off slowly(so no-one gets left behind)
We will pick up pace as we get further into term

Workbook: Task 1

This section mainly just introduces the workbook
Including a recap on a number of key OO concepts:



Workbook: Task 2

Slides/video refresh memory on Objects & Classes

A few "fairly gentle" practical tasks to achieve:

- Add a constructor method to the Triangle class
- Add 3 parameters to constructor (side lengths)
- Store side lengths as int variables ('attributes')
- Write a method that returns the longest side

Then add a few lines of code to test above features

Create a few triangles to be sure everything works

Workbook: Task 2 cont.

Add a 'toString' method that describes the triangle:

`This is a Triangle with sides of length 4, 5, 7`

Note that ALL Java Objects have a 'toString' method
It's good practice to override the default with your own
You should always try to return something descriptive
If you don't provide a 'toString' method, you just get:

`edu.uob.Triangle@754dd69e`

Workbook: Task 3

Explores the topics of Inheritance & Polymorphism
Slides and Video fragments from previous lecture

"Introduce" Triangle into hierarchy (using 'extends')
Use polymorphism to store different shapes...
All in the same 'TwoDimensionalShape' variable !

Some additional "PRO" material for deeper insight
(referencing, avoiding duplication, inheritance)
From the UG unit - provided "for interest"

Workbook: Task 4

Explores the topics of Abstraction & Encapsulation
Again, slides/video fragments from previous lecture

Add a 'Colour' variable to 'TwoDimensionalShape'
Important that 'Colour' is private (hidden inside)

Again, some optional "PRO" materials available
Find out about the subtleties of public/private

Workbook: Task 5

Refresher slides on "getters" and "setters"
Officially called 'accessor' and 'mutator' methods

Add 'getColour' & 'setColour' methods to shape class
Add colour details to string returned from 'toString'

Question: Where is the best place to add this code ?
HINT: Which shapes can have a colour ?
HINT: Could you use overriding and chaining ?

Workbook: Task 6

We provide an enum to represent triangle "variants":

EQUILATERAL, ISOSCELES, SCALENE, RIGHT, FLAT etc.

Add code to constructor to work out which variant

Store the result in a private attribute inside the class

To start, focus on whether or not it is EQUILATERAL
(You can consider other variants in a later task)

Write a 'getVariant' method that returns the attribute
In order to allow other objects to access the variant

Workbook: Task 7

Now work through the remaining types of triangle:

EQUILATERAL, ISOSCELES, SCALENE, RIGHT, ILLEGAL etc.

Add code to decide which type of Triangle it is

Order you consider variants in code IS important:

- Check "bad" variants first (ILLEGAL, IMPOSSIBLE)
- Then check "simple" variants (e.g. EQUILATERAL)
- Finally move on to check for "difficult" variants

Workbook: Task 7 cont.

Rather than manually adding/removing test code
(In order to create test triangles and do printlns)

We will instead do something more systematic !

We have provided a JUnit test script:

TriangleTests

Drop this file into your project and then...

Use IntelliJ to run the test methods



```
@Test  
void testEquilateral() { assertShapeVariant(TriangleVariant.EQUILATERAL,
```


Workbook: Task 7 cont.

Just *reading* test script is a bit tedious:

`TriangleTests`

So I wrote a graphical test visualiser (just for fun !):

`TriangleTestViewer`

This was created using a platform called 'Processing'
(Popular Java-based audio/visual framework)

Warning: Final Tests

Final couple of tests use some VERY large triangles
We must remember that data types are constrained
There is a limit to range of numbers an int can store
Also, float variables have limited precision (~ 7 DP)

HINTS

Explore the variety of primitive data types in Java
Be selective about WHAT calculations you perform
Also think about the ORDER you perform them in

Workbook: Task 8

At some point you'll need to test via the command line

Terminal
project-folder

Key Commands:

```
./mvnw clean
```

```
./mvnw compile
```

```
./mvnw test
```

Broken Maven Project ?

Terminal
project-folder

Need to ensure `./mvnw` script is executable `'chmod'`

Switching platforms can corrupt `./mvnw` script

The hidden `.mvn` folder can be missing (redownload)

Mismatch between installed and POM version of java

Be aware of difference between `'mvn'` and `'mvnw'`

To work !