Formatting Date Time

There are many ways to format date and time.

A couple of standardized ones are shown here.

These apply to the formatted method on String, as well as the printf method.

'R'	Time formatted for the 24-hour clock as "%tH:%tM"
'T'	Time formatted for the 24-hour clock as "%tH:%tM:%tS".
'r'	Time formatted for the 12-hour clock as "%tI:%tM:%tS %Tp". The location of the morning or afternoon marker ('%Tp')
'D'	Date formatted as "%tm/%td/%ty".
'F'	ISO 8601 complete date formatted as "%tY-%tm-%td".
'c'	Date and time formatted as "%ta %tb %td %tT %tZ %tY", e.g. "Sun Jul 20 16:17:00 EDT 1969".

This information was retrieved from the link I show here.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Formatter.html#dt

Formatting Date and Time

This slide explains the code I'm using in a bit more detail.

Format Specifier Description

```
%[argument_index$][flags][width]conversion
```

Example In Practice, using argument indices, one date time field and one String

```
"%1$tD %1$tT : %2$s %n".formatted(LocalDateTime.now(),  message)
```

%t – date time conversion

%tD - Date %tT - Time

%1$tD – Date retrieved from 1st argument

%1$tT – Time retrieved from 1st argument

It's common when using date time conversions, to use the argument index feature, which is called **Explicit Indexing**.

Controlling Change

Java provides mechanisms to control changes, and extensibility of your code, at many different levels.

You can prevent:

- Changes to data in Instance fields, which is called the state of the object, by not allowing clients or subclasses to have access to these fields.
- Changes to methods, by not allowing code to override or hide existing functionality.
- Your classes from being extended.
- Instantiation of your classes.