# Introduction[¶](#Introduction)

**Jon Minton. E63**

This document addresses the task listed in A.D.3

## Problem[¶](#Problem)

Include in your screenshot:

* An object diagram

The object diagram must contain:

* At least two objects contained in rectangles with the title of the class at the top (capitalised). This should also have the specific object title (e.g. for a class titled "Member", the object diagram would be titled "Alex: Member")
* At least one attribute for each object with values indicated (e.g. “name: Alex”, “age: 32”)
* Lines or "relationships" to indicate the structure of the objects and how they interact with each other. Types of relationships, such as many-to-many, do not need to be indicated at this level

Submit a comment explaining your understanding of:

* The object diagram you’ve submitted and its use
* Design decisions made when producing the diagram

Remember:

* Attach examples as image files or screenshots
* Use original work that is your own (i.e. no examples handed out as course materials (unless PDA specific), copied from the internet, or submitted by another student)
* Reattach all files on any resubmission
* Do NOT reuse examples from another submission

## Solution[¶](#Solution)

The figure below shows three objects, first\_entry, second\_entry and jblog of two classes, Entry and Blog. The objects first\_entry and second\_entry are instances of the class Entry, and the object jblog is an instance of class Blog.

The object jblog contains within it the objects first\_entry and second\_entry in its field entries. The links between these objects are shown with lines. The position of objects does not matter.

The two blog entries are created as instances of the same Entry class because they can be expected to contain the same types of attribute/field, even though their contents can be expected to be different. Although only one instance of the Blog class exists, jblog, other objects of this class could also exist, and will be expected to have the same kinds of attribute/field. An important design decision is that instances of the Blog class accept instances of the Entry class.

Both the Entry and Blog class may be expected to contain relevant class-specific methods, although these are not shown here. For example, and Entry class object might be expected to have .create(), .edit() and .delete() methods, and the Blog class may be expected to have .add\_entry(), .delete\_entry() and .count\_total\_views() methods, the last of which might iterate through the times\_viewed attribute of each entry in the entries list attribute.

## Appendix[¶](#Appendix)

The following shows the PlantUML code used to generate the figure above

@startuml three\_objects\_of\_two\_classes object "jblog: Blog" as jblog object "first\_entry : Entry" as first\_entry object "second\_entry : Entry" as second\_entry object first\_entry { name: "My first blog entry" date\_created: "10 April 2019" contents: "Hello. This is my first blog entry" times\_viewed: 24 } object second\_entry { name: "My second blog entry" date\_created: "4 January 2022" contents: "Hi. This is my second entry." times\_viewed: 4 } object jblog { name: "Jon's Blog" date\_created: "6 April 2019" first\_entry: "10 April 2019" last\_entry: "4 January 2022" entries: [first\_entry, second\_entry] } first\_entry -- jblog second\_entry -- jblog @enduml