Introduction to UML
(Unified Modelling Language)
Part one Classes and instances

Written by: Robin Beaumont e-mail: robin@organplayers.co.uk

Date last updated: Wednesday, 05 October 2011
Version: 1
How this document should be used:
This document has been designed to be suitable for web based and face-to-face teaching. The text has been made to be as interactive as possible with exercises, Multiple Choice Questions (MCQs) and web based exercises.

If you are using this document as part of a web-based course you are urged to use the online discussion board to discuss the issues raised in this document and share your solutions with other students.

Whom this document is aimed at:
This document is aimed at three types of people:

- Those who wish to become involved in systems development but are not interested in the nuts and bolts of programming. Such people are commonly called domain experts and act as bridges between a professional group (e.g. medics, Solicitors etc) to which they belong and IT experts.
- Those just beginning professional computer science courses.
- Those who wish to become involved in some form of analysis activity. This might be Process re-engineering or Data flow analysis etc.

I hope you enjoy working through this document.

Robin Beaumont
Contents

1 BEFORE YOU START .................................................................................................................. 4
2 LEARNING OUTCOMES - UML CLASSES/INSTANCES .......................................................... 5
3 BACKGROUND .......................................................................................................................... 6
   3.1 WHY LEARN ABOUT UML CLASS DIAGRAMS? ............................................................... 6
      EXERCISE 1. MCQs .............................................................................................................. 7
4 INTRODUCTION ......................................................................................................................... 7
5 OBJECT ORIENTED APPROACHES AND UML ................................................................. 8
6 CLASS INSTANCES ..................................................................................................................... 9
7 CLASSES ..................................................................................................................................... 10
   EXERCISE 2. INTRODUCTORY VIDEO ....................................................................................... 10
   7.1 DEFINED BY CONTEXT ....................................................................................................... 11
   7.2 PRESENTATION IN UML .................................................................................................... 12
   EXERCISE 3. MCQs .................................................................................................................. 13
   EXERCISE 4. CLASSES AND INSTANCES ............................................................................. 15
   EXERCISE 5. UML CLASSES AND INSTANCES .................................................................. 15
   EXERCISE 6. MCQs .................................................................................................................. 16
   7.3 IDENTIFYING CLASSES ..................................................................................................... 17
      7.3.1 Analyzing a Narrative Description to identify candidate classes ................................. 17
      EXERCISE 7. MARKING NOUNS ......................................................................................... 17
      7.3.2 Developing candidate Classes .................................................................................... 19
   EXERCISE 8. IDENTIFYING CLASSES FROM A NARRATIVE ............................................. 20
      7.3.3 Applying naming standards to Classes ........................................................................ 20
   EXERCISE 9. APPLYING STANDARDS TO CLASSES .......................................................... 20
      7.3.4 Workshops .................................................................................................................. 21
   7.4 DRAWING UML CLASS DIAGRAMS - CASE TOOLS .................................................... 21
   7.5 CLASS DIAGRAMS AND AMOUNT OF DETAIL SHOWN ............................................. 21
   7.6 VIEWS ............................................................................................................................... 21
   EXERCISE 10. MCQs .................................................................................................................. 22
   EXERCISE 11. DIFFERENT VIEWPOINTS ............................................................................. 22
   7.7 WARNING ABOUT THE ‘IS A KIND OF’ MISUNDERSTANDING .................................... 22
   7.8 TIGHT COUPLING .............................................................................................................. 23
   EXERCISE 12. INAPPROPRIATE ATTRIBUTES ..................................................................... 23
   7.9 CLASS DIAGRAMS AND DATABASES ............................................................................. 23
      7.9.1 The Relationship between Classes/Instances to Databases .................................. 23
   EXERCISE 13. LINKING CLASSES AND INSTANCES TO DATABASES ............................... 24
      7.9.2 Specifying data types .................................................................................................. 24
      7.9.3 Derived Attributes and Default values ....................................................................... 24
      7.9.4 Enumerated types ....................................................................................................... 24
   7.10 OPERATIONS ..................................................................................................................... 25
   7.11 SUMMARY ......................................................................................................................... 25
   EXERCISE 14. MCQs .................................................................................................................. 26
8 REFERENCES .............................................................................................................................. 26
9 WEB LINKS .................................................................................................................................. 27

Video of this chapter
You can see an overview of this chapter as the first video "UML class diagrams - basic introduction" at:
http://www.youtube.com/playlist?list=PL5D590974B7967000
1 Before you start

This document assumes that you have the following knowledge and skills:

**Skills/ knowledge:**
You should be able to explain and provide examples of the following database concepts:

- table,
- record,
- index,
- foreign key,
- relations specifically: one to one, one to many, many to many

If you are unsure of what these terms mean and are unable to provide examples, read through:

http://www.robin-beaumont.co.uk/virtualclassroom/chap7/s2/dbcon1.pdf - this provides details of what a table, record and index are.

http://office.microsoft.com/en-us/access-help/database-design-basics-HA001224247.aspx - Microsofts Database design basics page, goes further than you need but provides a good section on relations and how they work.

**Required Resources**

You need the following resources to work through this document:

- The “Scenarios for practicing modelling techniques” handout available from [http://www.robin-beaumont.co.uk/virtualclassroom/contents.html](http://www.robin-beaumont.co.uk/virtualclassroom/contents.html) and follow the links

- A UML case tool - Many of the concepts introduced in this document are difficult to grasp at first and are helped by experimenting with a Case Tool in addition to carrying out the exercises with pen and paper. Two such tools are Visual UML and MagicDraw. You can download a community version of MagicDraw at [http://www.magicdraw.com/](http://www.magicdraw.com/). In addition if you are reading this document as part of a course you are undertaking with the Royal college of Surgeons (Edin) or Edinburgh University you are registered as part of the academic programme for Magicdraw which means you are entitled to use the full personal edition of MagicDraw. To be able to use the full personal edition you need to contact your course administrator who has the codes to unlock MagicDraw.

**Video of this chapter**

You can see a overview of this chapter as the first video "UML class diagrams - basic introduction" at:

[http://www.youtube.com/playlist?list=PL5D590974B796700](http://www.youtube.com/playlist?list=PL5D590974B796700)
## 2 Learning Outcomes - UML Classes/Instances

This section aims to provide you with the following skills and information. After you have completed it you should come back to these points, ticking off those you feel happy with.

<table>
<thead>
<tr>
<th>Learning outcome</th>
<th>Tick box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Be able to describe the main characteristics of object oriented modelling</td>
<td></td>
</tr>
<tr>
<td>Be able to provide brief details of the relationship between OMT and UML</td>
<td></td>
</tr>
<tr>
<td>Be able to describe briefly the history of UML</td>
<td></td>
</tr>
<tr>
<td>Be able to describe what an instance is, as used in object oriented (OO) modelling</td>
<td></td>
</tr>
<tr>
<td>Be able to describe the three parts of a UML class</td>
<td></td>
</tr>
<tr>
<td>Be able to describe the two parts of a class instance (often called an object)</td>
<td></td>
</tr>
<tr>
<td>Be able to explain the difference between a UML class instance and a UML class</td>
<td></td>
</tr>
<tr>
<td>Be able to describe misunderstanding between UML classes/instances and the 'is a type of' concept</td>
<td></td>
</tr>
<tr>
<td>Be able to discuss the importance of context in identifying UML classes</td>
<td></td>
</tr>
<tr>
<td>Be able to produce a list of candidate classes from a narrative description</td>
<td></td>
</tr>
<tr>
<td>Be able to use Rumbaugh's criteria to help refine an initial list of candidate classes</td>
<td></td>
</tr>
<tr>
<td>Be able to use the Reingruber &amp; Gregory criteria to standardise class names</td>
<td></td>
</tr>
<tr>
<td>Be aware of the use of workshops and informal class diagrams to develop UML class diagrams in a group setting</td>
<td></td>
</tr>
<tr>
<td>Be able to identify and draw classes and instances using the UML notation</td>
<td></td>
</tr>
<tr>
<td>Be able to explain the relationship between models, classes, attributes, instances, and schemas, tables, fields and records.</td>
<td></td>
</tr>
<tr>
<td>Be able to explain data types, enumerated types and default values are additional characteristics of attributes</td>
<td></td>
</tr>
<tr>
<td>Be able to describe the varying amount of detail that can be displayed in a class diagram</td>
<td></td>
</tr>
<tr>
<td>Be able to discuss the concept of “view” as it relates to class diagrams</td>
<td></td>
</tr>
</tbody>
</table>
3 Background

In the past 30 years the profession of systems analyst (modeller) has come into being and developed rapidly. The first modelling techniques focused on designing databases, and for an enthralling description of the development of the first computer system, along with its database for Lyons teashops in the 1950’s see Georgina Ferry’s excellent book, A computer called Leo.

In the 1970’s Peter Chen, who is very much still alive and continues to produce important research, (http://www.csc.lsu.edu/~chen/chen.html) suggested a method to represent a high level diagrammatic description of the structure of any database. The diagram was called an Entity Relationship Diagram (ERD) and is still used today, you may have come across it when using Microsoft Access and it is an essential component of any Database administration System (DBMS). The ERD below shows the structure of a database used to collect research data from those who suffer from diabetes and are also pregnant. It represents the data at a high level of abstraction meaning that it is not necessary to show details of the various fields or indexes, just the table names (i.e. entity types) and optionally the field names.

For more information about ERDs see: www.robin-beaumont.co.uk/virtualclassroom/chap11/s9/erds_1.pdf

While ERDs are useful they are limited and a more modern approach is to use Class diagrams, this more modern approach is what we will concentrate on in the rest of this chapter.

3.1 Why learn about UML Class Diagrams?

This chapter, along with and several subsequent ones, focus on getting you to learn what class diagrams are and how to produce them. You may ask why, so here are a few reasons I believe it is important for you to do so:

- These diagrams form the basis of most database design methods. If you ever are involved in database design (i.e. data modelling) you will therefore need to understand them because if they are wrong (i.e. the blueprint), the database that is built from them will also be wrong!

- These diagrams can be transformed to the more traditional ERD to create databases.

- These diagrams provide a method of analysing a situation that forces you to adopt a stance of a typical database modeller. By using them you begin to realise how their minds work and also begin to appreciate why some databases are so problematic.

These are only a few of the reasons that I personally believe this skill is so important.

Even if you do not intend to become a programmer or systems analyst/modeller you may want to become the type of person who is able to provide a bridge between your professional group – such as doctor, vet or solicitor – and those who develop or manage information systems. Such people are vitally important in developing usable information systems and there is a great shortage of them. Such a person is often referred to as a ‘Domain Expert’ or subject matter expert (SME). Before you start to learn what class diagrams are and how to create them yourself there are some Multiple Choice Questions (MCQs) on the next page to see how much you have taken in so far.
Exercise 1. MCQs
1. From the list below choose two reasons why it is important for a ‘domain expert’/subject matter expert (SME) such as you to learn about the Class diagramming method:

   a. Provides insight into the mindset of database developers
   b. Is the only method available to describe the data requirements
   c. Provides credibility to IT personnel
   d. Forms the basis of most data modelling techniques
   e. Has been proved to be the most cost effective method of specifying data requirements

2. From the list below choose the one option that describes the most desirable ‘domain expert’ from the medical profession:

   a. Someone who has developed several databases but knows little of database modelling or current issues in medicine
   b. Someone who has little interest in how information may help the department
   c. Someone who has problems working in a collaborative environment
   d. Someone who has previously managed IT projects
   e. Someone who has knowledge of data modelling techniques and currently works in the appropriate situation

3. ERDs provide a description of (one correct answer):

   a. The processes that occur in the model
   b. The various entity types and their relationships in the model
   c. The entity types in the model
   d. The processes and data requirements of a model
   e. The User Interface aspects of the model

4. ERDs provide a (one correct answer):

   a. A detailed description of the data within a data model
   b. A detailed description of the proposed uses of a database
   c. A guide to the training costs
   d. A high level description of the data within a data model
   e. A graphical picture that is of little use in developing a database

4 Introduction

The process of designing any system, whether it be computer, paper or person based or a mixture of all three, consists of specifying two important aspects, the what (data - structure) and the how (what it does - behaviour). Considering the How aspect, a system is worse than useless if it either is difficult to use (e.g. for entering or retrieving information) or hinders rather than facilitates working practices. For example, if a system is planned to be used in a medical consultation it should be easier rather than more difficult to prescribe and allow the users to collect data in the way they would find natural. While this How aspect is as important as the data side of things, we will concentrate on the ‘what’ side of things in this document.

You may be familiar with ERDs, a graphical technique for specifying the what. However in this document we will be looking at a more advanced diagramming technique to produce diagrams explaining the What, namely UML Class diagrams. These diagrams are one aspect of ‘object oriented’ modelling techniques that have developed over the last few decades. Basically the Object oriented approach allows more complex models to be developed than previously possible. Before we consider the UML Class diagram in detail we will take a quick look at object oriented modelling techniques in general and UML.
5 Object Oriented Approaches and UML

In the early 1990s Rumbaugh as well as M Blaha et al developed a modelling process called OMT (Object Modelling Technique) which was documented in a book by them called Object-Oriented Modelling and Design, Prentice Hall 1991. This book has taken on the status of a classic and is still available although they have published what can be considered to be an update to this in 2005 – Object Oriented Modelling with UML.

So what are Object Oriented Modelling techniques and how do they relate to this thing called UML? Object oriented modelling relies heavily upon the following five ideas (concepts), which allow us to model aspects of the world in a logically consistent manner. Notice this is much wider than just database modelling:

- Classes and Objects
- Association
- Inheritance
- Encapsulation
- Polymorphism

In this chapter we will concentrate on the first aspect, although we will have a glance at the others in passing. So how does UML relate to Object Oriented Modelling? The answer is basically historical.

In 1998, Rumbaugh joined forces with Grady Booch and Ivar Jacobson, who also have their own Object Oriented Modelling languages, to create a Unified Modelling Language (UML). That year saw a burst of activity with several books being published describing UML (Fowler & Scott 1998, Blaha & Premerlani 1998), UML not only subsuming Rumbaugh’s OMT but also expanding it with new diagrams. Since this time UML has gone through a number of revisions with the most recent being version 2.4 (Around May 2011). I have therefore decided not to discuss OMT and the earlier versions of UML. The specification of the latest version of UML can be found at http://www.omg.org/.

UML is definitely the "in thing" and if you look in any computing/modelling/system development section of most large bookshops you will find a row of books about UML. Three that I would recommend are:


You can get second hand copies of the above books for far less by looking on www.addall.com, www.Amazon.com and www.amazon.co.uk Only the Bennett books talks about uml2 in detail.

UML provides a standard set of tools (most of us would call these diagrams – but the UML people get upset if you reduce them to such things!) for analysing and developing a model using an Object Oriented Approach. UML is not a method, you are free to use which tools (diagrams) you feel are appropriate and at the appropriate points you decide during the process. However this does not mean there is no relationship between the various diagrams; one of the most important aspects of a CASE TOOL is how it manages the relationships that exist between them as specified in the UML. Few if any modellers use all the diagrams; it depends upon the specific project and the personal preferences of the modeller.

Fowler 2004, p.12 has a nice diagram showing the various diagrams in UML (version 2) taken from the formal UML specification document; the greyed out boxes represent the classification of the diagrams. In UML version 2 there was 13 different diagrams, and now in UML2.4 14 diagrams:

In this and the following chapter we will focus on the Class diagram, probably the most important diagram in UML. But to help you understand what classes are we will look briefly at class instances first.
6 Class instances

So what is a class instance in the context of 'object oriented modelling'?

A class instance (often called an object in UML but I avoid this term) is best thought of as a definable thing with a number of facets that results in us perceiving it as something that is distinct.

What are these facets? well in biology people often talk about something having form and function, we can think of the 'descriptive' side of things as being data (e.g. a person having a name, age, hair colour and address etc) and the 'function' side of things as being activities (e.g. washing hair, giving birth, going shopping etc). Moreover, the data (structure) and activities (behaviour) are inextricably linked in our conceptualisation of the instance. If the instance should change either its activities or characteristics it exhibits, we feel distinctly uneasy. In other words, when we conceptualise an instance in the world, we do not naturally divide it out into these two aspects. To do this requires skill.

For example, when we see something we like, we tend to think of both actions as well as its characteristics (data), when one thinks of a banana or a bowl of soup, not only do both have pleasant characteristics such as colour and smell but they also have pleasant actions associated with them such as the soup being made (created, the default operation all instances possess) and eaten, and the banana growing and ripening, thinking of something more complex such as a Person, instances of this would be far more complex.

From the above we can say that we have two instances of class, Person and one each of class banana and soup. So we can have any number of instances for a particular class.

In the context of 'instance modelling' a class instance is therefore something in the investigator's mind that usually possesses a recognisable number of both characteristics (data consisting of data items) and actions. As in all areas of computer science, one word is never enough; and UML uses the word 'attribute' instead of data or characteristic.

The diagram above shows how class instances are drawn in UML. Each class instance is divided into two sections:

- The top box gives the name of the class instance along with its class name (more about that below).
- The bottom section lists each attribute along with its value, where each line is called a slot.

All the above examples of class instances indicate that the attributes have values suggesting that the class instance exists in some way and because of this it often means class instances are called 'real world things' but this can be misleading.

If you know about ERD diagrams you may be thinking that the concept of a class instance is very similar to that of an Entity Instance. In fact you would not be far from wrong; however, the class instance concept also has the concept of Operations within it, they are just not shown in the instance diagram.
In the above section we have discussed instances of Classes now we will investigate the actual Class concept.

### 7 Classes

The best way to think of a class is to think of the blueprint idea. A class is often called a concept, although philosophers argue about the differences. Many theorists have provided complex definitions of a class or entity type, which is a similar idea, but none are complete or infallible. Hopefully by the end of this section you will be able to conceptualise what a class is in this context even if you still won’t be able to provide a definition for it.

Chen, one of the early writers (circa 1970), felt that the task of class identification (he was talking about 'entity types' at the time, as instance modelling had not been developed) was best left to the individual who was within, and who knew most about, the system to be modelled. This suggests that classes are partially dependent upon the person who is identifying them, and this will become clear as you work through this section. The domain expert / subject matter expert (SME) plays a vital function between the end user and IT/programmers defining and correctly modelling classes.

What then is a class? A class can be thought of as an abstraction from the instance. Notice that in the examples concerning instances a single banana was described. The instance 'mybanana' is said to be an instance of class banana. Similarly John is said to be an instance of class Person and “mysoup” an instance of class soup. Notice that while the class person could exist without any instances, the reverse is not true. We say that an 'instance' is an instantiation of a class. This may appear rather pedantic but it is very useful in modelling as it provides a method of classifying groups of things often in very useful ways.

Basically:

- **Class = Template**
- **Instance = Example of a particular class**

All instances of a particular class must have the same structure as the class

The rest of this section will be concerned with identifying and organising Classes.

---

**Exercise 2. Introductory video**

Please watch my introduction to UML at: [http://www.youtube.com/watch?v=5d2bHVH3h8](http://www.youtube.com/watch?v=5d2bHVH3h8)

Also OPTIONALLY if you want a quick overview of UML and the various diagrams see:


Please note that both these sites by providing a very superficial overview make many technical errors - but they do provide a starting point.
7.1 Defined by context

Below are some examples of classes and instances for three different contexts:

A typical general practitioner planning on collecting information about patients:

<table>
<thead>
<tr>
<th>Class</th>
<th>Some possible instance(s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doctor</td>
<td>Name=Angus Wallace. Residence= living in the UK</td>
</tr>
<tr>
<td>Patient</td>
<td>Name=Joe Bloggs, Town=Newcastle, Name=Alan Smith, Town=London</td>
</tr>
<tr>
<td>Visit</td>
<td>Time=12/06/2001/4pm location+Prospect House Newcastle</td>
</tr>
<tr>
<td>Prescription</td>
<td>Drug=Valium, dosage=5mg. Frequency= eight times a day</td>
</tr>
<tr>
<td>Advice leaflet</td>
<td>Title=Chronic back pain leaflet</td>
</tr>
<tr>
<td></td>
<td>Title=UTI self management</td>
</tr>
</tbody>
</table>

The university administrator thinking about developing a course database:

<table>
<thead>
<tr>
<th>Class</th>
<th>Some possible instance(s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Coordinator</td>
<td>Name=Robin Beaumont, Name=Joe Brand</td>
</tr>
<tr>
<td>Student</td>
<td>Name=Paul Whatling</td>
</tr>
<tr>
<td></td>
<td>Name=Mary Brown</td>
</tr>
<tr>
<td></td>
<td>Name=Anne Smith</td>
</tr>
<tr>
<td>Tutor</td>
<td>Name=Ruth Brown</td>
</tr>
<tr>
<td>Module</td>
<td>Title=Introduction to Informatics, year=2003</td>
</tr>
</tbody>
</table>

A garage collecting detailed information about car components:

<table>
<thead>
<tr>
<th>Class</th>
<th>Some possible instance(s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark Plug</td>
<td>Make=Rover, model=49532</td>
</tr>
<tr>
<td>Engine</td>
<td>Escription=122 Brake Horsepower</td>
</tr>
<tr>
<td>Cooling system</td>
<td>Capacity=17 pints</td>
</tr>
<tr>
<td>Gearbox</td>
<td>Second gear ratio= 7.55:1</td>
</tr>
<tr>
<td>Propeller shaft</td>
<td>Type=Hardy Spicer</td>
</tr>
</tbody>
</table>

Are the above examples correct?

While the above examples probably list a selection of the important classes for each of the contexts, there is no absolute way of being able to say they are correct. Rumbaugh et al (1991, p21) makes this important point when discussing class identification:

“We define a class as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. Classes serve two purposes: they promote understanding of the real world and provide a practical basis for computer implementation. Decomposition of a problem into classes depends on judgment and the nature of the problem. There is no one correct presentation.” [edited slightly]

This is clear from considering the last example concerning car component details. Assume that another garage owner who was less concerned with collecting detailed information about each component only wanted one item of information recorded for each component. In this instance we would probably have CAR as the class and each of the classes listed above relegated to attributes of the CAR class. There is no definitive correct answer; we only know by talking to the actual garage owner. **Classes are largely defined by the context.**
7.2 Presentation in UML

Computer scientists, whenever possible, use diagrams or symbols rather than words. This is because words are considered to be the most ambiguous form of communication or more probably because most computer scientists are notoriously poor at English. The diagram below shows how UML would possibly represent several of the classes described previously:

In the diagram each box represents a class. The box is divided into three sections. The top section of the box provides the name of the class, usually a noun. The next section down provides details of the data (attributes or data items) in the class while the last and final section provides names of the activities (methods) that make up the class. While the first two examples -- Person and Banana -- are relatively easy to define in this way, the third is more difficult. This is probably because Soup has a large number of attributes but has very few activities. The only one I could think of, besides the generic ‘create’ which all classes have, how else would you create instances? is the process of decomposing when it's left too long, possibly 'heat up' might be another. Not all classes have actions (other than the generic ‘create’ and ‘destroy’), but all must have a name and something that uniquely distinguishes them from another class. Because instances are just instances of a particular class, these rules apply to them as well.

In this chapter we will concentrate on class diagrams rather than instance diagrams; these being more general, they tend to be more useful. However, instance diagrams are useful if you want to investigate a particular situation.

If you know about ERD diagrams you may be thinking that classes and Instances (instances) are the same as Entity Types and Entity Instances however Classes and instances are far more complex because they possess operations as well as attributes.
We will now take a specific example, assume that we have decided to propose that a valid class would be one named Doctor, firstly we need to consider if this is a valid class, again context comes into play if we were creating a model for a general purpose employment agency we would probably have a more generic class such as client, with thewre profession as an attribute in the class, however if we were modelling a GP practice or hospital we would have Doctor as a specific class. Assuming we have the latter situation:

<table>
<thead>
<tr>
<th>Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>surname</td>
</tr>
<tr>
<td>forename</td>
</tr>
<tr>
<td>dob</td>
</tr>
<tr>
<td>GMC_no</td>
</tr>
<tr>
<td>salary</td>
</tr>
<tr>
<td>gender</td>
</tr>
<tr>
<td>examine</td>
</tr>
<tr>
<td>question</td>
</tr>
<tr>
<td>diagnose</td>
</tr>
<tr>
<td>request_test</td>
</tr>
</tbody>
</table>

**Why is this a class?**

Because it as a name
Because it has a set of unique attributes i.e. surname, forename, date of birth, GMC number, salary, gender etc.

And a unique set of operations i.e. examine, question, diagnose, request test etc.

**Why is this a class rather than an instance?**

Because it possesses these attributes and operations but they do not have specific values

'Unique' here means different from any other class in the model

<table>
<thead>
<tr>
<th>Sarah_smith:Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>surname=smith</td>
</tr>
<tr>
<td>forename=sarah</td>
</tr>
<tr>
<td>dob=19/12/1956</td>
</tr>
<tr>
<td>GMC_no=39200456</td>
</tr>
<tr>
<td>salary=89K</td>
</tr>
<tr>
<td>gender=m</td>
</tr>
</tbody>
</table>

**Why is this an instance of the Doctor Class?**

Because it has the same set of attributes as the doctor class i.e. surname, forename, date of birth, GMC number, salary, age etc.

And each attribute has a specific value

Because it has the same set of operations as the doctor class (not shown in instance/object diagram)

Another way of thinking about the bond between a class and instance is to think of the class as being the column headings (= attributes + operation names) for a number of rows each of which is an instance of the class. For example

<table>
<thead>
<tr>
<th>Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>surname</td>
</tr>
<tr>
<td>smith</td>
</tr>
<tr>
<td>Coates</td>
</tr>
<tr>
<td>Worsley</td>
</tr>
</tbody>
</table>

**Exercise 3. MCQs**

1. Which one of the following is an example of a class:
   a. Robin Beaumont
   b. Carrot
   c. Date’s book An Introduction to Database Systems on my shelf at home
   d. Spike from the TV series "Buffy the Vampire Slayer"
   e. My backup disk (in the second drawer of my desk)

2. Which one of the following is an example of an instance of a class:
   a. Course Manager
   b. Health Informatics courses
   c. Women
   d. Tony Blair
   e. Religion
This is how I arrived at the answers for the exercise on the previous page

1. Which one of the following is an example of a class:

   a. Robin Beaumont could be:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>First_name</th>
<th>surname</th>
<th>dob</th>
<th>Mobile</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robin</td>
<td>Robin</td>
<td>Beaumont</td>
<td>12/06/87</td>
<td>07986750664</td>
<td><a href="mailto:you@me.com">you@me.com</a></td>
</tr>
<tr>
<td>Tony</td>
<td>Tony</td>
<td>Milner</td>
<td>05/09/1967</td>
<td>07986750664</td>
<td><a href="mailto:you@me.com">you@me.com</a></td>
</tr>
</tbody>
</table>
```

b. Carrot could be:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Variety</th>
<th>Germination_date</th>
<th>Seeding_position</th>
<th>Manufacturer</th>
<th>Taste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>Old red</td>
<td>04/09/2011</td>
<td>Plot4</td>
<td>Thomsons</td>
<td>Honey sweat</td>
</tr>
<tr>
<td>Tony</td>
<td>Tonys special</td>
<td>04/09/2011</td>
<td>Plot2</td>
<td>Thomsons</td>
<td>Sharp</td>
</tr>
</tbody>
</table>
```

c. Date's book An Introduction to Database Systems on my shelf at home

```
<table>
<thead>
<tr>
<th>Name</th>
<th>author</th>
<th>publisher</th>
<th>date</th>
<th>etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML for beginners</td>
<td>Beaumont</td>
<td>Brewester Brown</td>
<td>01/02/2016</td>
<td></td>
</tr>
<tr>
<td>An intro to databases</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

d. Spike from the TV series "Buffy the Vampire Slayer"

```
<table>
<thead>
<tr>
<th>Name</th>
<th>character</th>
<th>First appearance</th>
<th>Last appearance</th>
<th>etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Marsters</td>
<td>spike</td>
<td>1997</td>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>Seth Benjamin Gesshel-Green</td>
<td>Seth Benjamin Gesshel-Green</td>
<td>1998</td>
<td>2001</td>
<td></td>
</tr>
</tbody>
</table>
```

e. My backup disk (in the second drawer of my desk)

```
<table>
<thead>
<tr>
<th>Name</th>
<th>size</th>
<th>manufacturer</th>
<th>date</th>
<th>etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup1</td>
<td>5.25</td>
<td>imac</td>
<td>02/09/2010</td>
<td></td>
</tr>
<tr>
<td>My backup disk</td>
<td>3.25</td>
<td>dualik</td>
<td>10/09/2006</td>
<td></td>
</tr>
<tr>
<td>Backup3</td>
<td>5.25</td>
<td>imac</td>
<td>02/09/2010</td>
<td></td>
</tr>
<tr>
<td>Backup4</td>
<td>3.25</td>
<td>dualik</td>
<td>10/09/2006</td>
<td></td>
</tr>
</tbody>
</table>
```

So from the above only the carrot is at the class rather than instance level, the relevant entries are shaded yellow. A similar approach can be taken to the second exercise

2. Which one of the following is an example of an instance of a class:

   a. Course Manager - This is probably a class with a name attribute.
   b. Health Informatics courses - This is probably a class with attributes such as title, start_date etc
   c. Women - This is probably a class with attributes such as gender, surname dob etc
   d. Tony Blair - This is probably an instance of Human or Prime minister
   e. Religion - This is probably a class with attributes such as name, leader, estimated number of followers etc

While the table format is nothing to do with UML it does help to understand the link between the class and its zero or more instances

All instances of a particular class must have the same structure as the class
### Exercise 4. Classes and Instances

Spend five minutes filling in the tables below. List some possible classes along with examples of their instances. Some of your classes may have no or several instances. The important thing is that each instance possesses the same set of attributes as the entity type.

<table>
<thead>
<tr>
<th>Class name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes names:</td>
</tr>
<tr>
<td>Attributes values:</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes names:</td>
</tr>
<tr>
<td>Attributes values:</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes names:</td>
</tr>
<tr>
<td>Attributes values:</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Exercise 5. UML Classes and Instances

#### Part A

Consider your job to be a class. If you do not have a paid job, you may be a student for example, then use that instead. List two of your (clinical) activities along with the information about them which you feel would be useful to collect. If it helps, consider it in the context of a learning log book of some sort.

#### Part B

Draw a diagram that represents the following classes: community nurse, GP, clinical manager in a hospital, day care coordinator, director of finance in an acute trust (hospital outside the UK), patient, yourself and your boss.

#### Part C

Draw an instance diagram that represents an instance of some of the classes you described in the above tasks.
Exercise 6. MCQs

1. Which of the following provides the best description of a Class (select one)?
   a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
   b. A value given to a particular attribute (e.g. height - 230 cm)
   c. A thing that we wish to collect data about where one or more, real world examples of it exist
   d. A template for a group of things with the same set of characteristics that may exist in the real world
   e. An undefined concept that needs further clarification

2. Which of the following provides the best description of a class instance (select one)?
   a. A specific concrete object with a defined set of processes (e.g. John Brown with diabetes)
   b. A value given to a particular attribute (e.g. height - 230 cm)
   c. A thing that we wish to collect data about where one or more, real world examples of it exist
   d. A template for a group of things with the same set of characteristics that may exist in the real world
   e. An undefined concept that needs further clarification

3. From the following list, select two class instances:
   a. Steinway piano model D design template
   b. McGill type forceps as used in surgical operations
   c. Tony Blair (British prime minister)
   d. PIII type chip that is found in many modern PCs
   e. An advice leaflet for chronic back pain issued to Mrs Smith on 02/10/2002

4. Which one of the following statements is true (select one)?
   a. A UML class diagram can only display class names.
   b. Attributes are a rarely considered aspect of classes.
   c. Attributes must always be displayed in UML class diagrams.
   d. Attributes equate to table names in a database.
   e. Attributes can optionally be displayed in UML class diagrams.

5. Select from the following the best list of attributes for the class SHOE for someone working in a shoe shop (select one answer):
   a. 1, 2, 6
   b. 1, 2, 6, 3
   c. 1, 2
   d. 1, 6
   e. 1, 2, 3, 4, 5, 6

6. Select from the following the best list of attributes for the class RECIPE for someone following a recipe at home (select one):
   a. 4, 5, 7, 8
   b. 4, 5
   c. 4, 5, 6
   d. 4, 5, 8
   e. 4, 5, 6, 8

1=size
2=number of lace holes
3=propulsion
4=cooking time
5=number of portions
6=material colour
7=volume (loudness)
8=cooking temperature
7.3 Identifying Classes

While there is much written concerning UML and Class diagramming theory, far less has been written about actually developing the models. Two of the most common ways of identifying classes for developing UML class diagrams are:

- Narrative descriptions of requirements
- Workshops

Both of the above methods rely to some extent on expert domain knowledge to identify and classify classes. We will now look at the first method, using a narrative description of the system requirements on which to base the development of a UML class diagram.

Rather than give a brief description of each stage and then visit each in turn, I will just take you on a journey and then reflect upon what we will have done in a summary at the end. So now let’s start with the narrative description.

7.3.1 Analyzing a Narrative Description to identify candidate classes

Frequently the first thing to be produced when someone has the idea of developing a model (also called “the system” or “database” below) is a paper document describing what he or she wants. This often includes something like the following:

We wish to develop an information system for our hospital in which:

"Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two. Patients either pay for their treatment directly or through an insurance company. Healthcare assistants also attend to the patients, and a number of these are associated with each ward.

Initially the system will be concerned solely with drug treatment. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time.

The system must record details concerning patient treatment and staff payment. Some staff are paid part time, and doctors and care assistants work varying amounts of overtime at varying rates (subject to grade).

The system will also need to track what treatments are required for which patients and when, and it should be capable of calculating the cost of treatment per week for each patient.

When users use the system they will be able to print out as well as view on screen the results. (Although it is currently unclear to what use this information will be put.)"

(taken from http://www.umsl.edu/~sauter/analysis/er/er_intro.html and expanded)

The first stage is to pick out all the nouns (names) in the above passage; this provides a good baseline from which to consider possible entity types.

Exercise 7. Marking Nouns

Mark in the above narrative all the nouns (names) you see.
You can see in bold below the possible set of nouns (names) that I have come up with:

“Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two. Patients either pay for their treatment directly or through an insurance company.

Healthcare assistants also attend to the patients, a number of these are associated with each ward.

Initially the system will be concerned solely with drug treatment. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time.

The system must record details concerning patient treatment and staff payment. Some staff are paid part time and doctors and care assistants work varying amounts of overtime at varying rates (subject to grade).

The system will also need to track what treatments are required for which patients and when and it should be capable of calculating the cost of treatment per week for each patient.

When the users use the system they will be able to print out as well as view on screen the results. (Although it is currently unclear to what use this information will be put.)”

(taken from http://www.umsl.edu/~sauter/analysis/er/er_intro.html and expanded)

Gathering together all the above nouns produces the following list, not in any specific order:

<table>
<thead>
<tr>
<th>Candidate classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
</tr>
<tr>
<td>Doctors</td>
</tr>
<tr>
<td>Healthcare assistants</td>
</tr>
<tr>
<td>Care assistants</td>
</tr>
<tr>
<td>Ward</td>
</tr>
<tr>
<td>Drug treatment</td>
</tr>
<tr>
<td>Drugs</td>
</tr>
<tr>
<td>Patient treatment</td>
</tr>
<tr>
<td>Treatment</td>
</tr>
<tr>
<td>Staff payment</td>
</tr>
<tr>
<td>Staff</td>
</tr>
<tr>
<td>Insurance company</td>
</tr>
<tr>
<td>Screen</td>
</tr>
<tr>
<td>Use</td>
</tr>
<tr>
<td>Information</td>
</tr>
</tbody>
</table>

Each of the above nouns can be considered to be a ‘candidate class’. The next stage is to consider which of these are appropriate classes to include in the UML class diagram. To do this we consider Rumbaugh et al (1991 p152-3, repeated in Blaha & Rumbaugh 2005 p 185 -6) criteria.
7.3.2 Developing candidate Classes

Rumbaugh et al (1991 p152-3, repeated in Blaha & Rumbaugh 2005 p 185-6) provides some guidance as to how to identify appropriate classes. He suggests that you consider the following issues:

- **Redundant class types** Two class types may express the same information (i.e. two names for the same concept, or synonyms). In the above example DOCTORS and HEALTHCARE ASSISTANTS (HCA) might be considered to be just class instances of the class called STAFF; however, this is unlikely in the above example because we know from our own expert domain knowledge that doctors are concerned with prescribing whereas healthcare assistants are not, therefore they have different activities, furthermore some of the attributes will be different, doctors possess a GMC number whereas HCA’s do not. In addition the information we plan to collect is specifically concerned with prescribing. We therefore consider the class type STAFF to be redundant, at least for the time being; possibly when the system is developed further such considerations can be re-visited.

- **Irrelevant class types** If an class type has little or nothing to do with the problem, it should possibly be left out. In the above example one would need to clarify with the person requesting the system if they need information about INSURANCE COMPANIES. Also if the system is initially only concerned with drug treatments, is there any point collecting information about other treatments?

- **Vague class types** In a narrative description, often words are used indiscriminately. In the above example the description states that “Initially the system will be concerned solely with drug treatment” yet the following paragraphs refer to PATIENT TREATMENT and also TREATMENT. Are these three different class types or not? Again we can only find out by discussing this issue with the person who requested the database. We would probably suggest that we have two class types called DRUG_TREATMENT and NON_DRUG_TREATMENT where NON_DRUG_TREATMENT is concerned with information about any non-drug intervention and probably not included in the initial class diagram.

- **Class types that are really attributes** Often an initial class type is an attribute. In the above, COST has been classified as a possible class yet it is probably an attribute of NON_DRUG_TREATMENT or DRUG_TREATMENT or PATIENT (renaming it BILL).

- **Multiple roles become class types** "The name of a class type should reflect its intrinsic nature and not a role that it plays. For example, OWNER would be a poor name for a class type in a car manufacturer’s database. What if a list of drivers is added latter? What about persons who lease cars? The proper class is PERSON (or possibly CUSTOMER), which assumes various different roles, such as owner, driver, lessee.” (Blaha & Rumbaugh 2005 p 186). Similarly a doctor is not just a prescriber but a treatment giver and a reviewer of results.

- **Implementation information** The Class diagram is concerned with defining the data that needs to be stored. It is not concerned with the hardware or processing of the data. Therefore, in the above example SYSTEM, SCREEN and USER can be ignored. Similarly RESULTS are an outcome of processing data. Perhaps from your knowledge of Access or other databases, you will realise that such things as RESULTS and reports are just a process of using a query or report tool to interrogate the model.

Another type of problem results from the existence of homonyms. This is where the same noun in the narrative actually means different things in different places.

For example the phrases "take a variety of drugs" and “Initially the system will be concerned solely with drug treatment” both contain the word DRUG but when you think about this where the word appears in the first phrase it is probably referring to the actual bottle of pills (i.e. 29 Ampicillin capsules prescribed to Mr smith on 29/05/2014) or a specific bag of intravenous fluid that is given to a specific patient however in the second phrase it is probably referring to all Ampicillin given to all patients or even possibly all the ampicillin allocated to a specific hospital ward but not necessarily consumed. The word DRUG_TREATMENT seems to describe the first situation and DRUG the second, but this would need further investigation. Similarly as we have discussed above TREATMENT has several meanings and it may be necessary to add one or more additional classes to express the different concepts more clearly (e.g. DRUG_TREATMENT, DIETARY_PRESCRIPTION and ART_THERAPY etc).

So what have we ended up with after all the above deliberations? The following is the revised list of classes, for a very simple first attempt which we will obviously expand:

PATIENT, DOCTOR, DRUG_TREATMENT, DRUG

By concentrating on the drug aspect of treatment and considering each of Rumbaugh’s guidelines we have reduced the list from 27 to 4 for a possible initial UML class diagram. However I’m sure that in future developments a large number of other classes with be added or reinstated!
Exercise 8. Identifying Classes from a Narrative

**Time:** 60 minutes

Look through the Radio station scenario, found in the scenarios handout, and:
1. Mark all the nouns and produce a list of them.
2. Using Rumbaugh’s guidelines revise your list to identify candidate classes.

7.3.3 Applying naming standards to Classes

While the previous page was concerned with identifying classes, we will now look at one of the many informal naming standards suggested, this one is by Reingruber & Gregory 1994 (p65-77). While they disused this in the context of ERD modelling I feel that the criteria can be readily adapted to UML class modelling.

**Class names:**
- Should be **unique** for the particular model. (This does not usually apply to attribute names which only need to be unique to a particular class)
- Should be a **singular noun** (e.g. PATIENT not PATIENTS).
- Should be **self-explanatory** to those reading the UML class diagram.
- Should follow any naming conventions locally defined by the modellers. For example, two common constraints are that:
  - They should be written in **UPPER CASE**.
  - Possible **spaces** should be represented by the _ character (e.g. DRUG_TREATMENT rather than DRUG TREATMENT).
- Should **not** be a name of an **individual instance** (e.g. Freeman Hospital Newcastle upon Tyne).
- Should **not** express **more than one concept** (e.g. EQUIPMENT_BED, PATIENT_DOCTOR_NAME).
- Should be **documented** in addition to the diagram The description for each class should be clear, unambiguous and supplemented with examples of instances.

When carrying out this process it is a good idea to draw up a table similar to the one below to make sure you carry out the process, the initial proposed class type name on the left and the final class type name on the right.

<table>
<thead>
<tr>
<th>Initial class type name</th>
<th>Unique</th>
<th>Singular noun</th>
<th>Self-explanatory</th>
<th>UPPER CASE</th>
<th>The _ character</th>
<th>Not a individual instance</th>
<th>Not more than one concept</th>
<th>Documented</th>
<th>Final class type name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doctors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drug treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drugs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exercise 9. Applying standards to Classes

**Time:** 60 minutes

1. Looking back at the hospital example, consider the following four possible classes. Use the above guidelines to edit them appropriately:
   - Patients
   - Doctors
   - Drug treatment
   - Drugs
2. Revisit the initial set of classes you defined for the Radio station scenario and edit them accordingly. It may help to draw up a table similar to the one above.
7.3.4 Workshops

The alternative to using a paper document to partially derive a set of classes is to hold a series of interactive workshops. This requires commitment, including the willingness to learn, from the person(s) who have requested the system (i.e. database). They need to understand to a limited extent what you are learning now!

Often a mixed approach can be taken, using a paper document as the starting point, to a workshop. Running workshops requires good group working skills, and must be carried out in a non-threatening manner. Often the modellers need to show considerable tact when people get hung up on what the modellers themselves know to be irrelevant issues (e.g. font size/colour, type of screen or computer etc) for the particular task at hand.

Now that we have looked at how to create an initial list of classes we will consider the actual mechanics of drawing UML class diagrams

7.4 Drawing UML class diagrams - Case tools

It must be remembered that systems modelling, of which UML is one particular aspect, is a discipline that has only been in existence for about a quarter of a century. I was recently looking at a book on systems analysis written in 1984 (Daniels & Yeates) and only 2 of the 300 pages was dedicated to a similar topic of databases, and within those two pages no mention was made of diagramming.

To begin with learning to draw UML class diagrams is usually a pen and paper exercise and I would strongly encourage you to practice using this simple technique at the start - be warned you'll also need a large rubber! Drawing informal UML class diagrams and discussing them is a very good way of developing them in a measured and appropriate manner, flip charts, or electronic white boards are excellent in this respect.

Other documents on my web site provide detailed tutorials on how to use specialised software (called case tools) to draw UML class diagrams. http://www.robin-beaumont.co.uk/virtualclassroom/contents.html

7.5 Class Diagrams and Amount of Detail Shown

In a UML class diagram, varying degrees of detail of a class can be shown. Sometimes all that is shown is the class name. Alternatively the class name and attributes are shown, or alternatively both the attributes and operations are shown.

7.6 Views

At the beginning of this document it was stated that the identification and description of Classes/instances was partly a matter of individual interpretation. This is clear when one considers the examples given above. Now considering the NHS, a patient as perceived by a hospital chaplain is a very different instance from that perceived by a director of finance. If you asked each to draw a Class diagram like those given on the previous pages, each would come up with a different set of data and activities.

- **A model is always context specific.**
- **It is never purely a reflection of reality - whatever that is!**

It is interesting to note that a large amount of effort in the modelling process is directed towards getting these differing views consolidated in some way. The potential problem is tackled in different ways depending upon the particular software development lifecycle chosen. Frequently a stakeholder or cultural/political analysis is carried out to see how important each of the views are, thus allowing a way to prioritise one view over another. Another method is used during the actual software development phase of the lifecycle where different views, or user interfaces, to the data can be developed for different user types. These are just two of a whole host of techniques available to tackle this problem. This concept of a view is similar to that of a 'semantic purpose', which is described on page 22 of Rumbaugh (both editions same page).

The importance of being aware of potential problems with differing views in the healthcare sector is discussed in:

Exercise 10. MCQs

1. Which of the following types of software (applications) is most suitable for developing UML class diagrams?
   a. Desktop drawing package (e.g. Smartdraw)
   b. Desktop publishing application to ease the use of textual explanations
   c. Anything with drawing capabilities (e.g. Microsoft Word)
   d. A CASE tool
   e. A spreadsheet

Exercise 11. Different viewpoints

Describe what might be the main differences in viewpoints between some of the following:

- community nurse
- GP
- clinical manager in a hospital
- anaesthetist
- day care coordinator
- paediatrician
- psychiatrist
- microbiologist
- director of finance in an acute trust
- patient

You may want to devise some type of table to help organise your thoughts.

7.7 Warning about the ‘is a Kind of’ misunderstanding

To start with, people are often confused about the connection between classes and instances thinking that an instance “is a kind of” class. An example will help to make this clearer. Someone might say that for the class ANIMAL instances in this case might be BIRD or HUMAN. This is not usually the case. The reason for this is because thinking about the structure, that is what attributes and behaviour you might want to collect about BIRDS and HUMANs would probably be different in several respects for each of them. Would you really want to collect information about the size of wingspan for a HUMAN or the IQ of a particular bird? In other words the set of attributes and operations for each is different.

Unfortunately it is not always as simple as this. Here is the argument for keeping BIRD and HUMAN as instances of HUMAN. Suppose you were an antique dealer and just wished to collect information about the instances of ANIMAL statue you had. In this case you would probably just want to record the number of ANIMAL statues you had rather than be animal specific, information in which the amount and type would vary for each. In this situation you are happy about keeping the same type of information about each of the statues regardless of what they represent, probably just species name would do for your attribute name. **The context dictates what class types you will find.**

Lest consider another example. This time assume we have a class called ??? with the following 5 instances with some of the attributes shown. You will notice that you can divide up the instances into two groups based upon the empty attributes. It look like we need actually two classes here one for men and one for women? However we do have a certain level of commonality between them with the First_name, Date_of_birth and waist_size attributes.

We will learn latter that this ‘is a kind of’ situation can be very effectively modelled by using a concept called inheritance in UML class diagrams. Much more about this latter.

<table>
<thead>
<tr>
<th>Class = ???</th>
</tr>
</thead>
<tbody>
<tr>
<td>First_name</td>
</tr>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Philip</td>
</tr>
<tr>
<td>Anne</td>
</tr>
</tbody>
</table>

For now the take home message is each class in a UML diagram: **Has a DIFFERENT STRUCTURE** (i.e. Different attributes and operations)
7.8 Tight coupling

From the above sections you will realise that each class has a distinct set of attributes and operations, not only is it vitally important that each class is distinct from others in the model but the class must be a closely knit structure for example a patient class should not have leakage from other classes in the model such as GP or APPOINTMENT in the model below.

Exercise 12. Inappropriate attributes

Consider the following classes and mark the attributes you feel are inappropriate in each class. Remember that the attributes in a class are those that define the class not aspects that may relate to it.

Model 1 - GP surgery

Model 2 Travel agency

7.9 Class diagrams and Databases

UML is often, BUT NOT ALWAYS used to develop computer systems and frequently these are databases. When people first start developing databases they always make two fundamental errors. Firstly, they rush to a computer to play with the DBMS (e.g. Access). Secondly, they believe they can create a perfect database by specifying a model without creating a prototype of some sort. It is vitally important to get the design right, but this involves many revisions resulting partly from feedback from people using the actual database. For each iteration the Class diagram(s) allow you to create a blueprint (model, specify, design, etc) of the database structure, and the DBMS allows you to create (implement, develop, etc) the actual database structure.

In the diagram to the left, the data dictionary is shown as a half-way house between the ‘high level’ class model(s) and the actual database. Depending upon your inclination it is often not necessary to produce a data dictionary as a detailed description of each attribute is adequate; it is basically up to you and whomever you may be working with. (See Blaha & Premerlani 1998 for more detail.)

Again I must reiterate Class diagrams are not only used to develop databases!

7.9.1 The Relationship between Classes/Instances to Databases

If we accept the object oriented view of the world, databases themselves are also simply a collection of classes/instances. More specifically a table is simply a way of allowing the computer, via a piece of software called a DBMS (Database Management System, e.g. Access), to store these classes/instances. This all sounds rather abstract, so I will now provide an example.

The diagram opposite demonstrates the relationship between the class Patient and a DBMS table structure.

The attributes in the class diagram become fields in a table where the table name is the same as the class name.

Taking this one step further, a class instance is equivalent to a record in a table in the database. Also at the top most level the complete ERD is called a schema which is comparable to the complete UML class model.
Exercise 13. Linking Classes and Instances to Databases

Spend a few minutes taking some of the classes / instances you have identified in the previous exercises indicate how they might relate to tables, fields and records. It might help to create a picture similar to the ones on the previous page.

7.9.2 Specifying data types

Fields in databases besides having a name also specify what type of data they store, for example it might be a number of characters (called a string), a whole number (called an integer) or a true/false value (called a Boolean) similarly in a UML class diagram you can specify the data type of individual attributes providing an even greater degree of linkage between the UML class diagram and a database structure.

7.9.3 Derived Attributes and Default values

Often in databases field (attribute) values are derived from other field (attribute) values and you can show this in UML classes by using the forward slash (/) before the name, for example say we have a derived attribute in the Patient class called bmi (Body Mass Index) which is derived from two other attributes in the class height and weight, the situation opposite. Optionally you can add a note to the UML class diagram to provide additional details, but I personally prefer to add the detail to the narrative accompanying the diagram as the diagram gets very cluttered with lots of notes. The derived attribute does not necessarily use attributes from within the class but can make use of attributes from others.

When a new record (instance) is created the actual value for each of the attributes is empty, however it is often useful to set a default value for some of the attributes. In the example below we have two more attributes in the patient class called dateRegistered data type date, and isHypertensive with data type Boolean, for the former attribute the default value is the date the instance was created i.e. today and for the latter attribute it is given the value, False indicating that they are not hypertensive.

Obviously the default values can be overwritten, and reasons for a default value should always be considered carefully, for example in the above, is it sensible for all new registered patients to be considered to be normotensive, probably a better value would be empty or not known.

7.9.4 Enumerated types

From the above example it appears that what we need is really a isHypertensive field/attribute that can take one of three values {yes, no, not known} you can achieve this by defining what is known as an enumerated type. An enumerated type allows us to specify a limited number of options that a particular attribute/field can take and you can think of it as a pick list. In the example opposite I have defined a new enumerated type called hyp_options, and set its default value to ‘unknown’. The valid options for this new enumerated type are either specified within the documentation accompanying the diagram or alternatively using a special enumerate class with the name hyp_options in this example, personally I prefer the first option as often you have a large number of enumerated types in a class diagram and it can get very messy. We will discuss this again in the next chapter.
7.10 Operations

So far we have not discussed the third compartment in each class in UML class diagrams. This compartment specifies the behaviour of the class, its dynamic aspects. An example is given opposite. Although clearly this aspect is very important, discussion of it will be deferred to the chapters concerned with dynamic modelling as there are specific UML diagrams that help with the specification of classes. However having said that there is nothing stopping a modeller directly adding operations (i.e. behaviours) when considering the UML class diagram.

7.11 Summary

In this chapter we have focused on one particular aspect of UML class diagrams, the classes by considering:

- historical aspects considering ERD’s - the forerunner to UML class diagrams, + the development of the Object Oriented Modelling approach accumulating in the creation of UML.
- various stages one can go through to create a list of initial (candidate) classes, then applying various criteria and naming standardisation. Consideration was also given to common problems related to class identification along with the ‘type of class’ misunderstanding.
- How classes and instances are inextricably linked
- How classes relate to database structure. and the further specification of attributes in UML

I have reproduced the class identification/development method below for easy reference:

1. Identifying nouns and drawing up a list
2. Removing:
   a. Redundant classes (e.g. synonyms)
   b. Irrelevant classes
   c. Vague classes
   d. Classes that are really attributes
   e. Multiple Roles amalgamated into a single class
   f. Implementation information
3. Adding classes due to homonyms
4. Considering Reingruber & Gregory 1994’s guidelines creating a table with the following columns:
   a. Initial class type name
   b. Unique
   c. Singular noun
   d. Self-explanatory
   e. UPPER CASE
   f. The _ character
   g. Not an individual instance
   h. (Not more than one concept)
   i. Documented
   j. Final class type name

We could have also used a more interactive approach with clients in a workshop environment, and developed UML class diagrams with them. After a few MCQs we will move onto the second aspect of UML class diagrams, the relationships. If you feel like one this would be a good time to take a break.

We have now completed our investigations into UML classes, however clearly these classes must relate to one another is a way analogous to the way tables do in a database structure, which will be investigated in the next chapter.

For now go back to the start, look at the mind map on the front page and also the list of learning outcomes, see how much you have managed to learnt. Also remember there are my Youtube videos and practical Case tool tutorials to reinforce this knowledge.
Exercise 14. MCQs

1. Which one of the following best describes the technique used to identify classes in a narrative?

   a. Identification of verbs, the application of various criteria and standards (e.g. naming conventions etc) to refine the list and then the creation of a list of appropriately named classes
   b. Identification of nouns then the application of ISO standards to create a list of appropriately named classes
   c. Identification of verbs then the application of standards (e.g. naming conventions etc) to create a list of appropriately named classes
   d. Identification of nouns, the application of various criteria (e.g. Reingruber & Gregory 1994’s) and standards (e.g. naming conventions etc) to refine the list and then the creation of a list of appropriately named classes
   e. Identification of a few important nouns then the application of standards (e.g. naming conventions etc) to create a list of appropriately named classes

2. Which of the following criteria are true (choose three)?

   a. Attribute names must always be unique for a particular model.
   b. Class names must be unique for a particular model.
   c. Attribute names must be unique for a particular class within a model.
   d. A class should only express one concept.
   e. Class names should be either singular or plural nouns.

8 References

Alavi M Wetherbe J C 1991 Mixed prototyping and data modelling for information system design. IEEE software May 87 – 91. One of very few articles which uses an experimental design in place of the purely non experimental case study, to assess various methodologies.


Blaha M Premerlani W 1998 Instance-oriented Modelling and design for Database applications; Includes UML. Prentice Hall


Checkland P Scholes J 1990 Soft Systems Methodology in action John Wiley. This book includes a case study from the NHS.

Date C J. 1995 (6th ed.) An introduction to database systems

Fowler M Scott K 1998 UML distilled: Applying the standard instance language. Addison Wesley

Newman M. Rosenberg D. 1985 Systems analysts and the politics of organisational control OMEGA int. j. of mgmt sci., 13 (5) 393 - 406

Open University 1993 Relational Database Systems M866 (Five books; Introduction to database technology, The Relational Modal, Normalisation, Using SQL, SQL database management) [These are excellent resources with many exercises and clear concise explanations].

Pender A Thomas 2002 UML weekend crash course. Wiley publishing inc.


Introduction to UML
part 1 - classes and instances


Joubert M The Use of Conceptual Graphs for Knowledge Bases, Customisation and Actual Data Organisation, Unpublished project working paper (AIM NUCLEUS)

9 Web Links

Past student's recommendations:

   Class and sequence diagram guidelines and more:
   http://www.agilemodeling.com/style/classDiagram.htm
   http://www.agilemodeling.com/artifacts/sequenceDiagram.htm