High Level Programing Paradigms

What the Specification Says
Identify a variety of programming paradigms (low-level, object-oriented, declarative and procedural);
Explain, with examples, the terms object oriented, declarative and procedural as applied to high-level languages, showing an understanding of typical uses;
Discuss the concepts and, using examples, show an understanding of data encapsulation, classes and derived classes, and inheritance when referring to object-oriented languages;
Understand the purpose of the Unified Modelling Language (UML);
Interpret class, object, use case, state, sequence, activity and communication diagrams;
Create class, object, and use case and communication diagrams;
Discuss the concepts and, using examples, show an understanding of backtracking, instantiation, predicate logic and satisfying goals when referring to declarative languages.

Notes

Programing Paradigms
A programing paradigm is just a method, or *a way* of programing. This unit will cover the 4 main paradigms(low-level, object-orientated, declarative and procedural)

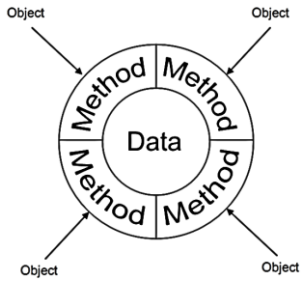
Low Level Paradigms
- Assembly language is a low level language.
- Assembly language was developed to make early programing easier, by replacing machine code functions with mnemonics and addresses with labels.
- It is a second generation language. Because it came after using strait machine code.

Procedural Languages
- Third generation, also called high level languages
- Problem orientated
- They describe exactly step by step what procedure should be followed to solve a problem.
- They use the constructs sequence, selection and repetition.
- May use functions and procedure, but they always specify the order instructions must be executed in, in order to solve the problem.
- Can be difficult to reuse code

Object Orientated Languages
- Where data and the methods for manipulating the data are kept in a single unit, called an object.
- The only way the user can access the data in the object is through the objcts methods
- Once an object is fully working, it cannot be corrupted by the user – because they cannot change any of its parameters.
- Also the internal workings of the object may be changed without affecting the workings of the object.
- A class is a construct that is used as a blueprint or template for an object.
- An object is an entity that can be manipulated by the commands of the class.

Declarative Programing
- The computer is told what the problem is, not how to solve it.
- It searches for a solution from a database
- Very useful for solving problems in the real world, like medical diagnostics.
- The user inputs a query to the search engine, which searches a database for the answers, and returns them to the user.
- The programmer does not have to tell the computer how to answer the query, the system just consist of a search engine of facts and rules.
- Backtracking is going back to a previously found successful match in order to continue a search.
- Instantiation is giving a variable in a statement a value.
- Predicate logic is a branch of mathematics that manipulates logical statements that can be either True or False.
- A goal is a statement that we are trying to prove whether or not it is True or False

Encapsulation



- Data encapsulation is the concept that data can only be accessed via methods provided by the class.
- This ensures data integrity.
- Data encapsulation is the combining together of the variables and the methods that can operate on the variables so that the methods are the only ways of using the variables.

Inheritance

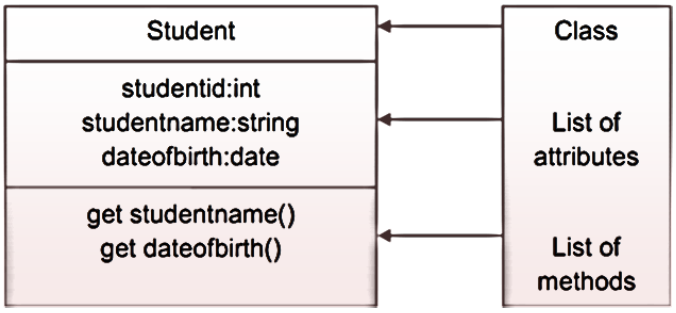
- Inheritance allows the reuse of code and the facility to extend the data and methods without affecting the original code.
- A derived class can be create which inherits its structure from the super class, but has added data.
- The derived class can be accessed through its methods, or the superclass's methods.
- A class describes the variables and methods appropriate to some real-world entity.
- An object is an instance of a class and is an actual real-world entity.
- Inheritance is the ability of a class to use the variables and methods of a class from which the new class is derived.

Unified Modelling Language (UML)

The Unified Modelling Language consists of a set of descriptive diagrammatic representations to describe the stages to produce effective object oriented programs. For the exam, we need to know 7 types.

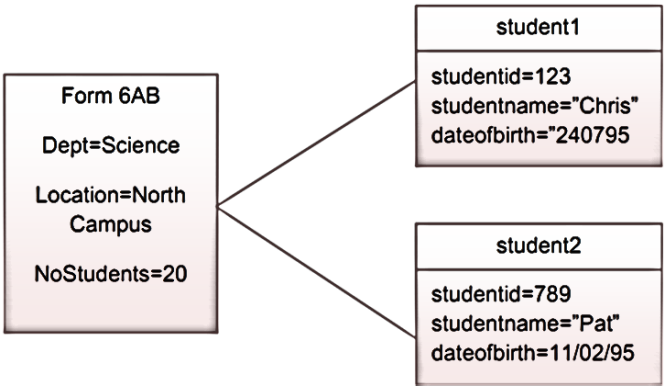
Class Diagrams

- Object oriented programming uses classes.
- A class is 'an entity of a given system that provides an encapsulation of the functionality of a given entity.'
- A class diagram is a rectangle, divided into 3 parts. The first gives the name of the class, the second gives the facts that should be known about any element belonging to that class and the third gives methods that can be used to look at the facts that are stored in the class.



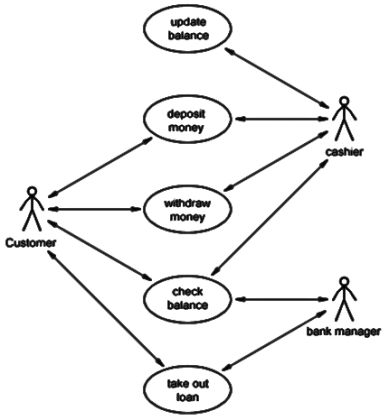
Object Diagrams

- If classes are groups of real things, with each class representing at least one entity
- Each object will have specific data to match each of the attributes.



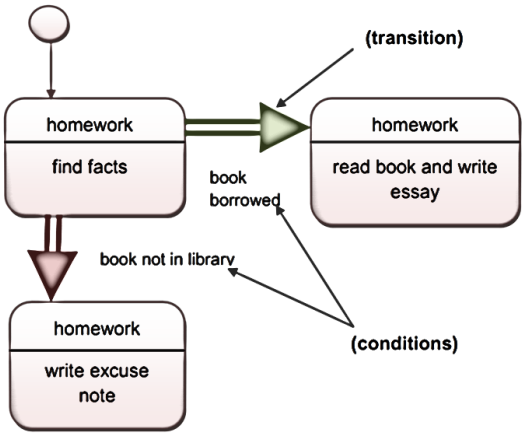
Use Case Diagrams

- Like business-orientated diagram show what should be going on in a system.
- Shows the users, or people.



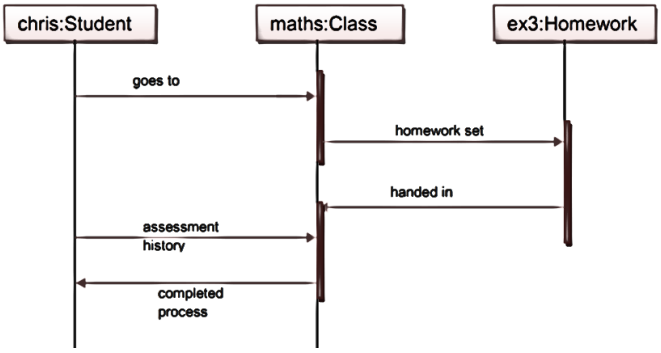
State Diagram

- A state diagram shows how an object may behave through the various processes of a system. It is a bit like a cross between a DFD and a flow diagram.
- It starts with a shaded in circle to show the initial state before anything has happened. Arrows are used to show the flow of activity to different outcomes for the object as it goes through the system



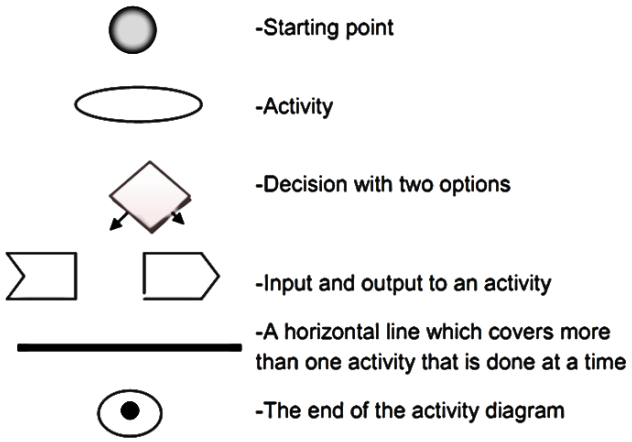
Sequence Diagrams

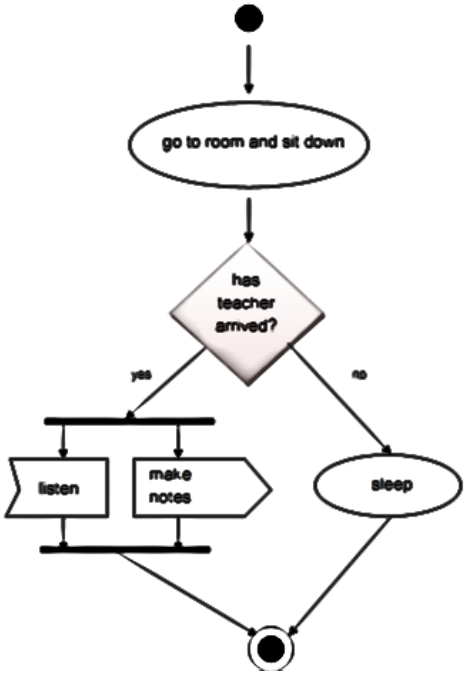
- The show how the objects in a class interact with one another.
- The classes go along the top of the diagram, with a dotted line underneath it which shows how long that object is said to exist, called it's lifeline. Where the dotted lines sometimes turn into thin boxes is where the classes methods have been called to do something.



Activity Diagram

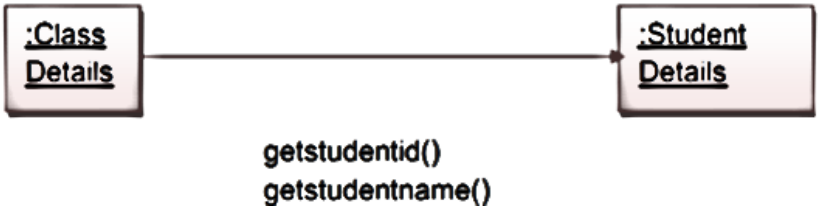
- Similar to a flow chart in procedural programing, although it shows the activities necessary to get an object into a particular state, rather than the development of the logic behind the code.





Communication Diagrams

- Used to show how different objects combine to pursue a common purpose.
- Each object is contained in a rectangle and the data that they share with one another is placed beside arrows showing in which direction the data is flowing.



Key Words

- **Programing Paradigm** – method of programing (e.g. low-level, procedural, declarative or object-orientated)
- **Low Level Language** – 2nd generation language with a one-to-one relationship with assembly code, makes use of mnemonics that represents assembly operations and labels that represent data locations.
- **Procedural Language** – where instructions are comprehensive, sequential and specify how to solve a problem.
- **Declarative Languages** – a problem-orientated language where the computer is given a set of facts and a goal, which it uses to interrogate a data base and find the solution. The computer is not told how to solve the problem, just what the problem is.
- **Goal** – this is that is trying to be found (in declarative programing)
- **Instance** – an existence of a piece of data (in declarative programing)
- **Backtracking** – where the computer goes back and takes another route while trying to find solutions to the problem (in declarative programing)
- **Predicate Logic** – is the process of different facts being applied to rules to produce a result that can be considered true or false.
- **Object-Orientated Programing** – paradigm that relies on objects in the real world being classified.
- **Class** – a group of data (in OOP)
- **Data Encapsulation** –data can only be accessed through methods in the class
- **Derived Classes** – classes that have inherited data from a super class.
- **Unified Modelling Language (UML)** - descriptive diagrammatic representation that describe stages required to produce object orientated programs.
- **Object Diagram** – shows the attributes for a specific object from a class.
- **Use Case Diagram** – shows what is happening in a system, rather than how it is done.
- **State Diagram** – shows the state of an object through the process.
- **Activity Diagram** – show how the logic behind the program was developed.
- **Communication Diagram** - shows how different objects combine to pursue a common purpose.
- **Sequence Diagram** - shows how the objects in a class interact with one another.
- **Class Diagram** – represents the classes, showing their methods and data.

Declarative Programing

- The computer is given a **set of facts and a goal**.
- It does not need a set of instructions - it is capable of **deciding how to solve the problem** itself.
- When it is run, the computer **applies the definitions to the facts** and supplies the result.
- The **goal** is what is being searched for.
- The value found is an **instance**
- When it has either found a result, or failed to find a result on one path, it will go back and check the other possible routes. This is called **backtracking**.
- Declrative languages use **predicate logic**.

Object Orientated Programing

- This is another programing paradigm, also called OOP.
- It relies on objects in the real world being classified.
- It can show information about a group of things that must have the same characteristics - the group is called a **class**
- The objects can provide data, which can only be accessed through the objects methods, this is called **data encapsulation**.
- **Derived classses** have **inherited** data from a **superclass**
- The methedology for planning and developing object orientated code is called **Unified Modeling Language (UML)**
- UML consists of a number descriptive diagramtic representations that decribe the stages required to produce OO programs.

Past Exam Questions and Answers

Explain backtracking

after finding a solution (to a goal)
go back and follow an alternative path…
…to attempt to find another solution

What is the need for BNF?

to unambiguously
define the syntax of a computer language

Why is UML used?

a standard way to present (information)…
…the design of a system…
…which is visual, so easy to understand
allows systems analysts, programmers and clients to communicate
makes system maintenance easier…
…when modifying a system

Describe the features of a procedural language

imperative language
uses sequence, selection & iteration
program states what to do…
… & how to do it
program statements are in blocks
each block is a procedure or function
logic of program is given as a series of procedure calls