WJEC A2 Computing Unit 3.4->3.5 Cheat Sheet

by AlexHoratio (Horatio) via cheatography.com/21127/cs/16034/

Programming Paradigms

Imperative	Directly telling the computer what to do via a series of instructins. **Procedural** and **object-oriented** programming are both subsets of imperative programming.
Declar-ative	This is where the programmer can "**declare**" what the solution should look like, and the language will obtain the results without the programmer explicitly telling it how. A common example of this is **SQL**.
Procedural	This is where a program is given a sequence of instru-ctions - a **procedure** - to solve a problem or perform a task. For example, **C** and **Pascal** are both procedural languages.
Logic	The language is given a set of facts and rules about what is happening, and the computer can then extrapolate data. If the computer is told that B is the child of A, it would deduce that A is the father of B. An example is **Prolog**
Functional	This is a language that uses **functions** to obtain and calculate values. For instance, **double(4)** would return the value 8. Examples of this are **C++** or **Python**.

Low Level Programming

Low level code involves working very closely with the physical inner-workings of the computer, for example directly pushing to **registers** and the **accumulator**.

Low level languages are usually:

1. Difficult for programmers to use since even simple tasks require a lot of code.
2. Used to develop performance-int-ensive programs or tasks.
3. Very small in terms of file size
4. Usually hard-coded onto ROM chips for small jobs

Types of Changeover

Direct Changeover

The old system is taken away and the new system is implemented completely overnight. This is **risky**, since if the new program has errors it could be catast-rophic.

Parallel Running

The old system is run alongside the new system for a while. This involves duplic-ating all the work done on both systems, but if there are errors in the new system then things are not as catastrophic.

Piloting

The new system can be tested out by one branch of the company at a time so that if anything fails, the entire company is not in trouble.

Phased Changeover

Part of the new system is used, then once all the problems have been worked out then another part can be used.

Types of Documentation

Technical Manual

This would include technical details for use by a programmer. For example it might involve **data flow diagrams**, **flowcharts**, **variable lists** or **data dictio-naries**.

User Manual

This is written for use by the end user. This might involve instructions for starting and installing the program, clerical procedures, how to use the program, or fixes for common errors.

Object-Oriented Programming

What is it?

Object-Oriented languages are languages which deal with "objects". Objects usually have a set of attributes and methods which are specific to that type. Objects are usually instances of **Classes**.

What's a class?

A class is a "template" for an object to be generated from. This would outline the member variables(or **attributes**) and methods for use later in the program.

What's inheritance?

A class can inherit from another class. This is commonly used for creating a very broad definition(e.g, Animal) and then extending it for more specific functi-onality(e.g Dog). A class which inherits from another class will have all the attributes and methods of the class it inherits from.

High Level Languages

High level languages are languages which create a great deal of abstraction between the programmer and the CPU itself. Examples of this are **C**, **Java** and **Python**.

High level languages are usually:

1. Easier for humans to understand
2. Problem-oriented
3. Used to develop programs which can run on different processors

System Life Cycle

1. Problem Definition

State the **aims** and **objectives** for the program.

2. Feasibility Study

Determine whether the project is economically/technically viable. This might involve **legality**, **time expense**, **monetary expense** or **technological complexity**.

3. Investigation

Carry out **questionnaires**, **interviews**, **observations** and **study existing documents** to determine some of the requirements for the program.

4. Design

Create a detailed design for the new system, involving **database modelling**, **screen designs** and map out what **processes** will need to be carries out on the data.

5. Development

The program is actually created. Alongside this, the program will be tested via either **alpha**, **beta** or **acceptance** testing. **Alpha testing** is performed by people employed by the company who have not worked on the project. **Beta testing** involves giving the program to people outside the company for testing. **Acceptance testing** involves giving the program to the client for feedback.

System Life Cycle (cont)

6. Implementation/Changeover

Changing from the old system to the new system. There are many different approaches to this, such as **parallel running**, **phased changeover**, etc.

7. Maintenance/Review

The system is maintained via either **adaptive**, **corrective** or **perfective** maintenance.

Types of Languages

Event Driven Languages	These are languages which can wait for an event to occur, such as a key press. When an event occurs it is handled by an **event handler**.
Markup Languages	These are a set of instructions to denote how an item should look, or how data is to be organized. Examples of this include **HTML** and **XML**.
Visual Languages	A visual language uses graphical representations of code in order to make it more intuitive for inexperienced programmers. An example of this is **Scratch**

Polymorphism

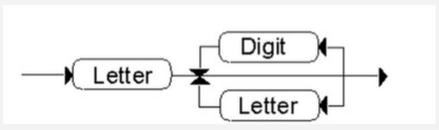
Polymorphism is when a single operator(for example, **+**) has multiple meanings depending on data type. + could mean either integer addition or string contatenation, based on the context its used.

Encapsulation

Encapsulation involves the usage of "**getters**" and "**setters**" in order to prevent things from changing without consequences.

For example, if the weight of a plane was updated, the fuel per hour would also need to be updated. This would be calculated and set appropriately via a **setter**.

Syntax Diagrams



Backus Naur Form

BNF is a standard for writing the rules of synta language.

BNF uses three symbols to define syntax:

::=

"is defined by"

|

"OR"

< >

Define a variable

An example of this is the following:

1. `<di git> ::= 0|1|2|...|9`
2. `<in teg er> ::= <di git> | <di teg er>`

Types of Maintenance

Perfective

This involves improving the system to run faster or perform better for its intended objective.

Adaptive

If there are changes in the company structure, requirements, etc then the system may need to be **adapted** to fit the new specification.

Types of Maintenance (cont)

Corrective

> This involves correcting problems or bugs in the program that were not noticed or fixed during testing.

Waterfall Approach

See above, also the waterfall approach is gay

Agile Approach

The Agile Approach is designed to be a **more flexible** alternative to the **Waterfall Approach**. Agile involves working closely with the client. Unfortunately, Agile usually yields **less detailed documentation** and **less specific deadlines** than the Waterfall Approach. There are two well known implementations of the Agile Approach.

Extreme Programming

> This emphasizes four areas of the Agile approach which are **communication**, **simplicity**, **feedback** and **courage**. The developers are frequently given opportunities for regular feedback from other developers and the client. Also, they often program in pairs known as **pair programming**.

Scrum

> **Scrum focuses on regular iterations to create prototypes quickly**. Scrum usually involves having iterations called **"sprints"**, where at the end of each sprint the team will reconvene and update each other on the progress that has been made.