FACTFILE: GCSE DIGITAL TECHNOLOGY

Unit 4

DIGITAL AUTHORING CONCEPTS

Rewarding Learning

Contemporary Trends in Software Development

Learning Outcomes

Students should be able to:

- describe the following programming paradigms;
  - Procedural programming
  - Object-oriented programming

- explain the significance of the following aspects of software development environments;
  - Editing features
  - High level code translation and execution

Programming Paradigms

Procedural Programming

In procedural programming, the programmer sets out, step by step, what the computer needs to do. They do this in the form of instructions which are written in source code.

A procedural language is *imperative*, meaning that it gives orders or instructions.

A procedural language is *sequential*, meaning that instructions carried out one after another. These steps form the *program flow*.

Object Oriented Programming

Object Oriented Programming (OOP)uses the concept of self-contained objects, which contain both program routines and the data being processed. An object-orientated program is designed as a collection of objects which interact by sending messages to each other.

An object is an element of a program that can perform actions and interact with other elements of the program. When designing a program or application objects are considered first. During the design process objects are defined formally.

Once all of the objects have been defined, the building of an application can begin. Objects occur in the real world, for example a person could be described as an object. The person would have certain attributes such as age, hair colour, height, eye colour and so on. In OOP the attributes that describe objects care called *properties*.

An object can perform actions. In OOP these actions are called methods. Methods allow the objects to do things and allow the properties of the object to be manipulated.

Object classes may be used in other programs.

These programs are more reliable because the objects are self-contained, simple and so easier to program.

Software Development Environments

Program code can be created using a text or code editor. The code (source code) is then translated into machine code. Source code cannot be executed by the computer until it has been translated into machine code.

If a text editor is used, the program will have to be compiled using program software. More advanced software known as an Integrated Development Environment (IDE) provides programmers with all of the facilities required to complete the development of an application from coding to testing. This means that an application can be developed fully within one application.

Typically, an IDE consists of the following elements:

- Source code editor
- Compiler
- Debugger
- Graphical User Interface(GUI) builder with an associated toolbox of controls
- Solution Explorer

Editing Features

The basic window for entering source code provides many of the following code editors:

1. **Clipboard**: Programmers can copy items and choose one to paste into the current file.
2. **Code Outlining**: Programmers can collapse/ expand selected regions of code under their first lines. This means that long programs can be viewed in small logical sections.
3. **IntelliSense**: As you enter a function or statement in the Code Editor, its complete syntax and arguments are shown in a ToolTip. When items are needed to complete a statement, IntelliSense provides popup insertion lists of available functions, statements, constants, or values to choose from.
4. **Line Numbering**: Line numbers help programmers to distinguish between lines in lengthy coding sections
5. **Syntax error assistance**: As code is entered, the Code Editor will place 'wavy lines' beneath code that is incorrect or could cause a problem.

High level code translation and execution

High level code is code that allows the programmer to reflect the way they want to solve the problem and not have to worry about the internal computer processes. However a program called a translator is then needed to convert the high-level code into something the computer does understand (machine code). Translators can be **compilers** or interpreters

Compiler

- A compiler **translates the whole program into machine code before the program is run**.
- All bugs are reported after the program has been compiled so it is difficult to test individual lines of code.
- All errors must be removed before the code is fully compiled.
- The machine code is saved and stored separately to the high-level code.
- Compilation is slow but machine code can be executed quickly.

Examples of Compilers: Java and C++

Interpreter

- An interpreter **translates source code line by line**.
- The processor executes a line of code before proceeding to translate the next line of code.
- Errors are displayed as soon as they are found so it can be easier to debug than compiled code.
- Source code is generates each time it runs, therefore it can be slower to execute than compiled code.

Example of interpreted languages: Javascript, PHP, Phython

Sources:

BCS Glossary of Computing and ICT 13th edn.

AS Digital Technology Fact Files

http://www.bbc.co.uk/education/guides/zgmpr82/revision/2