Types of Programming Language (1.2.4)

Programming Paradigm: A specific way to write programs.

Procedural	Object-Orientated	Low Level (Assembly)
Statements grouped together in self- contained blocks called subroutines.	Program is split into smaller units, called objects.	Use mnemonics to represent specific sets of machine code
Subroutines have local variables.	Objects can be easily reused and inherited without having to rewrite code.	Specific to a processor
Logic of the program is expressed in a series of procedure calls.	Each object contains methods and attributes.	Has a one-to-one relationship with machine code

Name of Paradigm	Procedural
Features of Paradigm	Sequence, selection, and iteration. Data stored in local or global variables.
Advantages	Lots of user support Good level of control over CPU
Disadvantages	There are many procedural languages, so a programmer must be very good at one language to get work.

Assembly Language:

- Gives you close control of the CPU, and can be very efficient

Address Mode	Explanation	Advantages	Disadvantages
Immediate addressing	The operand is the needed data	Useful for short constants	
Direct addressing	The operand is the memory address of the needed data	It is still very fast.	Slower than Immediate Mode since the CPU must look up the data. Cannot have re-locatable code (code depends on specific memory locations). Address size is limited by operand size.
Indirect addressing	The operand is the address of the address of the data.	More space for addresses	Slower than direct addressing, since the CPU needs to look up two things
Indexed addressing	The address of the needed data is the sum of the operand and a base address.	Fast for manipulating arrays and lists. If the array is re-located in memory, then only the base address needs to be changed.	

Memory Address Modes: modes that determine the method used by a program to access data or the next instruction.

All classes have three main sections:

- The name of the class, its attributes (data), its methods (functionality)

Class: A blueprint used to define the attributes and methods of a certain type of object. Used to Instantiate objects. Instantiation: The process of creating an object using a class. Calls the Constructor method with the initial attributes: a special method that runs when an object is instantiated.

variable_name = NEW class_name(initial attributes)

Inheritance: A mechanism allowing you to derive a class from another class, sharing the superclass's attributes and methods with the subclass. Allows us to reuse code quickly and easily and extend a class's attributes and methods without affecting or rewriting the original code. Inheritance arrows are drawn from the subclass to the superclass.

Dynamic Polymorphism: Method Overriding; where a method in the subclass overrides a method in the superclass of the same name. Thus, a subclass can have a more detailed version of a method in a superclass.

CLASS subclass INHERITS superclass

Encapsulation: The bundling of attributes with the methods that operate on and restrict direct access to them. Encapsulated attributes should only be accessible via methods. This keeps attributes safe and keeps the programmer in control. An object's methods are normally public, so that they can be used to access the object's private attributes.

Access Modifier: These will specify which properties (attributes and methods) can be accessed from outside the class. Public Attribute: Can be directly accessed anywhere in the program. Private Attribute: Can only be accessed by the public methods in the same class. Protected Attributes: Can be accessed from within the class and from classes that inherit from this class (but from nowhere else). Static Method: A method that accomplishes an action that is relevant to the class rather than a specific object. The method will be used even when there are no instances of the class.

Data Hiding: Where only the class designer should understand the inner details and workings of the class (but no-one else).

Access Modifier	Symbol
Private	-
Public	+
Protected	#

Benefits of Encapsulation: Easier to debug code, Attributes can have access modifiers

Multiple Inheritance: Where a subclass inherits from multiple super-classes. The main problem with this is Method Overriding. For example, Class C inherits from both Classes A and B; both super-classes contain a method called Hello(). If an instance of Class C is asked to execute the method Hello(), and if it has not been overridden in class C, the computer will not know which method to execute.

Abstract Method: A method declared without implementation, but with the name, the parameters, and all data types. Abstract Class: A class that declares abstract methods. Abstract methods tell the programmer what functionality a class needs to implement without defining implementation that may not be useful.