Practical Task 2.2 P

- How classes are used to define objects.
- How methods, fields (attributes), and properties all work together when you create a class.
- How fields give knowledge to each object created from a class.
- How methods give capabilities to each object created from a class.

Classes in C# act as templates for building objects. The "Account" class in the program I wrote is specified with the attributes "_balance" and "_name," as well as the methods "Deposit," "Withdraw," "print," and "GetName." These establish the framework and How an `Account} object behaves An object uses the blueprint provided by the class to determine how much memory to reserve when it is instantiated from it. For instance, "Account Account1 = new Account("Alex Jones", 10000);" in the "Main{" function of the "TestAccount" class generates a new "Account" object with the name "Alex Jones" and an initial balance of $10,000.
Fields are used to display an object's data or state. The private fields "_balance" and "_name" of the "Account" class stand for the account's name and balance, respectively.

An object's behavior is represented via its methods. The methods "Deposit", "Withdraw", "Print", and "GetName" of the "Account" class define operations like making deposits and withdrawals, printing account details, and getting the account name.
Fields can be safely exposed and encapsulated using properties. For instance, "_name" and "_balance" could be made available as properties in this case to provide regulated access to these values from outside the class. Each object's data or state is stored in a field. An object inheriting from a class has a unique set of attributes that are assigned particular values. For example, upon creation, "Account1" receives unique "_balance" and "_name" properties that set it apart from all other {Account} objects.

What can be done by an object is defined by its methods. The methods defined in a class are accessible to all objects created from it. For instance, "Account1" can carry out operations like making deposits, taking withdrawals, and printing account details by calling methods like "Deposit," "Withdraw," and "Print." In conclusion, classes provide the structure (fields), behavior (methods), and optional properties that make up an object's blueprint. When an object is instantiated from one of these classes, it has its own state (fields) and is able to use the class's defined methods for operations. This encapsulation facilitates the creation of modular and sustainably written programming.