The four principles of Programming with Classes are Abstraction, Encapsulation, Inheritance, and Polymorphism. These four principles enable powerful programming for a myriad of applications.

The first principle is Abstraction. Abstraction takes complex ideas and turns them into something simple, reducing repetitive tasks and making it easier to implement changes efficiently. In the goal tracking project, the abstraction CreateGoal asked a user for information about a goal, created the goal, and added it to a list. The programming for this simple step would be cumbersome to include every time it was needed. By creating an object with the programming needed for this step and then simply calling on the abstraction, we eliminate the need to do all that work over and over again. This also means that any changes to the code only need to be made in one place.

The second principle is Encapsulation. Encapsulation is about hiding certain data in the code in a way that protects it within a class and makes it impossible for that data to be manipulated from other parts of the program. Encapsulation makes it easier to fix or update something because you only need to check the class, not the whole program. In the online ordering program, the product class held the product name, product ID, price, and quantity and how those variables would be handled. By doing this, the other parts of the program couldn’t directly manipulate them.

The third principle is Inheritance. This principle allows derived classes to “inherit” the properties of the base class. This means that if different classes, which are all part of a larger category, need to do similar things, they can share a base class to handle those tasks. Each class can then focus on what makes it unique, avoiding repeated code and making updates easier. In the mindfulness project there were multiple types of activities that each did specific things, but they all had common traits as activities. The different types of activities: Breathing, Listing, Reflecting, were all “activities’, so it made sense to have an Activity class to handle common tasks. Inheritance made it possible for each individual activity to access the attributes and methods of the base. This property also prevents unnecessary duplication of code that would need to be tracked down when a change needs to be made.

The fourth principle is Polymorphism. This principle takes Inheritance to a new level by making it possible to execute an action in different ways, depending on the behavior that is required. In the exercise tracking program, running, swimming, and cycling are all exercise activities but the way their metrics are measured is different. Polymorphism allows the needed technique to override the basic concept. Getting the distance for swimming required data for time and laps to make the calculation. For cycling, time and speed were used. All activities used GetDistance from the base class, but the derived classes were able to override with the process needed. This principle keeps the information that is needed only where it is needed, facilitating simple access to making changes in the future.