**1. Vehicles and Cars**

**Objective:** Introduce students to inheritance, constructors, and overriding ToString().

**Instructions:**

1. Create a **base class** called Vehicle with:
   * Properties for Make, Model, and Year, using get and set.
   * A constructor that initializes these properties.
   * An overridden ToString() method that returns a string in the format: "Year Make Model".
2. Create two subclasses, Car and Motorcycle, that inherit from Vehicle.
   * Car should have an additional property NumberOfDoors.
   * Motorcycle should have an additional property HasSidecar.
   * Each subclass should have a constructor that calls base() to initialize Make, Model, and Year.
   * Override ToString() in each subclass so that Car returns "Year Make Model with X doors" and Motorcycle returns "Year Make Model (Has Sidecar: true/false)".

**2. Employees and Specializations**

**Objective:** Reinforce constructor chaining and overriding ToString().

**Instructions:**

1. Create a **base class** called Employee with:
   * Properties for Name and Salary.
   * A constructor that initializes these properties.
   * An overridden ToString() method that returns "Employee: Name, Salary: $Salary".
2. Create two subclasses, Manager and Engineer, that inherit from Employee.
   * Manager should have an additional property NumberOfEmployeesManaged.
   * Engineer should have an additional property Specialization.
   * Each subclass should have a constructor that calls base() to initialize Name and Salary.
   * Override ToString() so that Manager includes the number of employees managed, and Engineer includes the specialization.

**3. Shapes and Areas**

**Objective:** Teach students how to define and override methods in subclasses.

**Instructions:**

1. Create an **abstract base class** called Shape with:
   * A Color property.
   * A constructor to initialize Color.
   * An **abstract method** GetArea(), which will be implemented by subclasses.
   * An overridden ToString() method that returns "Color Shape".
2. Create two subclasses, Rectangle and Circle, that inherit from Shape.
   * Rectangle should have properties for Width and Height, and its GetArea() method should return Width \* Height.
   * Circle should have a Radius property, and its GetArea() method should return Math.PI \* Radius \* Radius.
   * Each subclass should override ToString() to also include the shape’s dimensions and calculated area.

**4. Animals and Sounds**

**Objective:** Reinforce method overriding in subclasses.

**Instructions:**

1. Create a **base class** called Animal with:
   * A Name property.
   * A constructor to initialize Name.
   * A **virtual method** MakeSound(), which returns "Some generic animal sound".
   * An overridden ToString() method that returns "Animal: Name, Sound: MakeSound()".
2. Create two subclasses, Dog and Cat, that inherit from Animal.
   * Dog should override MakeSound() to return "Bark".
   * Cat should override MakeSound() to return "Meow".
   * Ensure that ToString() still correctly calls MakeSound() when printing the object.

**5. Library System**

**Objective:** Combine inheritance, encapsulation, and method overriding.

**Instructions:**

1. Create a **base class** called Book with:
   * Properties for Title, Author, and YearPublished.
   * A constructor to initialize these properties.
   * An overridden ToString() method that returns "Title by Author (YearPublished)".
2. Create two subclasses, EBook and PrintedBook, that inherit from Book.
   * EBook should have an additional property FileSizeMB.
   * PrintedBook should have an additional property PageCount.
   * Each subclass should override ToString() so that EBook includes the file size and PrintedBook includes the number of pages.