Tutorial 2: C# OOP Principles and ADO.NET Tutorial

This tutorial introduces students to Object-Oriented Programming (OOP) principles in C#, explains key .NET OOP keywords, and provides a basic introduction to ADO.NET using the AdventureWorks2022 database.

OOP Principles in C#

Encapsulation

Encapsulation restricts access to internal object data and exposes only necessary parts through public members.

Example:

```
public class BankAccount {
    private double balance;

    public double Balance {
        get { return balance; }
        set {
            if (value >= 0) balance = value;
        }
    }
}
```

Inheritance

Inheritance allows a class to inherit members from another class, promoting code reuse.

Example:

```
public class Animal {
    public void Eat() {
        Console.WriteLine("Eating...");
```

```
    }
}

public class Dog : Animal {
    public void Bark() {
        Console.WriteLine("Barking...");
    }
}
```

Polymorphism

Polymorphism allows methods to behave differently based on the object that invokes them.

Example:

```
public class Animal {
    public virtual void Speak() {
        Console.WriteLine("Animal sound");
    }
}

public class Cat : Animal {
    public override void Speak() {
        Console.WriteLine("Meow");
    }
}
```

Abstraction

Abstraction hides implementation details and shows only essential features.

Example:

```
public abstract class Shape {
    public abstract double GetArea();
}

public class Circle : Shape {
    public double Radius;
    public override double GetArea() => Math.PI * Radius * Radius;
}
```

.NET OOP Keywords

Keyword	Explanation
class	Defines a class type.
interface	Defines a contract that classes can implement.
abstract	Specifies that a class or member must be implemented in derived classes.
virtual	Allows a method to be overridden in a derived class.
override	Overrides a base class method.
new	Hides a member inherited from a base class.
public	Accessible from any other class.
private	Accessible only within the containing class.
protected	Accessible within the containing class and derived classes.

Introduction to ADO.NET

ADO.NET is a data access technology in .NET that enables communication between applications and databases.

Key ADO.NET Objects

- **SqlConnection**: Establishes a connection to the database.
- **SqlCommand**: Executes SQL queries and commands.
- **SqlDataReader**: Reads data from the database in a forward-only stream.
- **SqlDataAdapter**: Fills DataSet and updates the database.

Using AdventureWorks Database

Example: SELECT Query

```
using System;
using System.Data.SqlClient;

class Program {
    static void Main() {
        string connStr = "Server=localhost;Database=<db_name>; User Id=sa;Password=<your_password>;";
        using (SqlConnection conn = new SqlConnection(connStr)) {
            conn.Open();
            SqlCommand cmd = new SqlCommand("SELECT FirstName, LastName FROM Person.Person", conn);
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read()) {
                Console.WriteLine(reader["FirstName"] + " " + reader["LastName"]);
            }
        }
    }
}
```

Example: JOIN Query

```
SqlCommand cmd = new SqlCommand(
    "SELECT p.FirstName, p.LastName, e.JobTitle FROM HumanResources.Employee e " +
    "JOIN Person.Person p ON e.BusinessEntityID = p.BusinessEntityID", conn);
SqlDataReader reader = cmd.ExecuteReader();
while (reader.Read()) {
    Console.WriteLine(reader["FirstName"] + " " + reader["LastName"] + " - " + reader["JobTitle"]);
}
```

Example: Scalar Function

```
SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM Person.Person", conn);
int count = (int)cmd.ExecuteScalar();
Console.WriteLine("Total Persons: " + count);
```

Example: Aggregate Function

```
SqlCommand cmd = new SqlCommand("SELECT AVG(Rate) FROM HumanResources.EmployeePayHistory", conn);
double avgRate = Convert.ToDouble(cmd.ExecuteScalar());
Console.WriteLine("Average Pay Rate: " + avgRate);
```