Module 1 Day 11

Inheritance

Inheritance

- To receive from a parent or ancestor by genetic transmission
- A subclass *inherits* all the **data** and **behaviors** from its superclass
- “Is a” (*is* C# operator)
- “Programming differences”
- Generalization / specialization
- Multiple levels of inheritance are possible (grandparent/parent/child)
- Any class may derive from no more than 1 parent class

Superclass	Subclass
Parent	Child
Base	Derived

Extending Classes

- “Derive from” (“extend”) a base class using :
- Call parent class using the special base operator
- Implement constructor(s)
- New access modifier: protected
- If not specified, Object is parent

```
// Account is the superclass of all account types
public class Account
{
    // Property that holds the account balance
    public decimal Balance { get; protected set; }

    // Constructor
    public Account(decimal initialBalance)
    {
        this.Balance = initialBalance;
    }
```

```
// Saving account is a sub-class of Account
public class SavingsAccount : Account
{
    public double InterestRate { get; private set; }
    public SavingsAccount(decimal initialBalance) : base(initialBalance)
    {
        this.InterestRate = 0.009;
    }
```

Overriding Behavior

- Use virtual in superclass to “allow” override
- Use override in subclass to provide a new implementation

```
public class Account
{
    Property and Constructor definition hidden...
    // Remove money
    virtual public decimal Withdraw(decimal amount)
    {
        // The base behavior for a withdrawal
        this.Balance -= amount;
        return amount;
    }
```

```
public class SavingsAccount : Account
{
    Property and Constructor definition hidden...
    public override decimal Withdraw(decimal amount)
    {
        // Do not allow overdrawing an account!
        if (amount > this.Balance)
        {
            Console.WriteLine("Error! You cannot withdraw more than you have!");
            return 0.00M;
        }
        else
        {
            return base.Withdraw(amount);
        }
    }
```

Using Super- and Sub-classes

- If an object “is-a” subclass, then by definition, it “is-a” subclass’s superclass (a grandfather clock “is-a” clock)

```
List<Account> myAccounts = new List<Account>();

SavingsAccount savings = new SavingsAccount(500.00M);
CheckingAccount checking = new CheckingAccount(100.00M, 0.50M);

myAccounts.Add(savings);
myAccounts.Add(checking);

foreach (Account account in myAccounts)
{
    account.Withdraw(10.00M);
    Console.WriteLine($"The account balance is {account.Balance}");
}
```