Abstract Classes

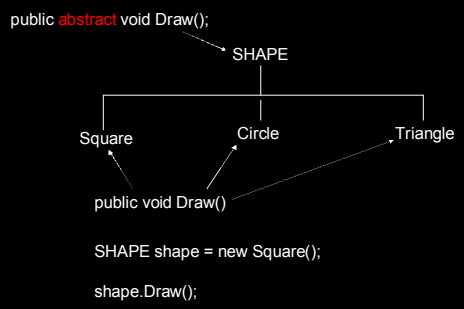
Abstract Classes

- Give us the ability to implement polymorphism in our objects.
- Polymorphism is the ability of different objects to respond uniquely (i.e., in a way appropriate to them) to the same message when the type of object is not known until run time.
- Run-time binding (aka Dynamic binding)

Abstract Classes

- Non virtual methods employ compile time binding (aka Static binding)
- Method to execute is fixed at compile time

Class Hierarchy



Polymorphism

- Define all methods you want to behave polymorphically in the (abstract) base class.
- No implementation in abstract base class.
- Implement each method (e.g., Draw()) differently depending on the type of derived object.

Abstract Classes

- Base class must be marked with keyword **abstract**.
- Each method to be implemented polymorphically must also be marked with keyword **abstract**.
- public abstract void Draw();

Abstract

- Can be used on classes, methods, and properties
- An abstract class
  - cannot be instantiated
  - may contain abstract methods and properties
  - cannot be sealed
- Derived types must implement all methods and properties that are abstract.

Abstract Methods

- They are virtual
- Only permitted in abstract classes
- Cannot have virtual, static, or override on them
- Contain no implementation and no braces{} following signature
  - public abstract void MyMethod();

Example

- www.seasite.niu.edu/zerwekh/abstractExample.htm