

Abstract Classes



ABSTRACT CLASSES



- **Objectives**

- Improve design by using abstract classes



- **Contents**

- The problem if we have no abstract classes
- Abstract classes with abstract members
- Polymorphism and abstract classes

- **Hands-on labs**

- Creating abstract classes



WHAT ARE ABSTRACT CLASSES



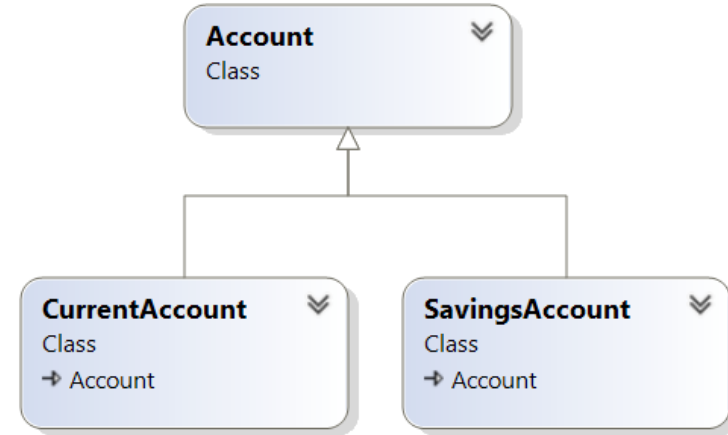
- **You would not create and draw a shape**
 - You draw a derived type like a rectangle, circle
- **You would never open an account**
 - You open a type of Account like saving account
- **Some classes are not meant to be instantiated**
 - These are marked as **abstract**
- **abstract classes follow these rules:**
 - They can hold **fields, methods, constructors**
 - May have zero or more **methods** marked as **abstract**
 - Cannot be instantiated
 - Are a base for inheritance
 - **e.g.** all shapes have in common (x, y, w, h...)

Problem with a concrete base class

```
class Account
{
    double balance;
}

class CurrentAccount : Account
{
    public void Withdraw(int amount)
    {
        Console.WriteLine("CurrentAccount withdraw");
    }
}

class SavingsAccount : Account
{
    public void Withdraw(int amount)
    {
        Console.WriteLine("SavingsAccount withdraw");
    }
}
```



```
Account[] accounts = { new CurrentAccount(), new SavingsAccount() };

foreach (Account acc in accounts)
{
    acc.Withdraw(50);
}
```

Will this code compile?

Problem with a concrete base class...

```
class Account
{
    double balance;
    public virtual void Withdraw(int amount) {}
}

class CurrentAccount : Account
{
    public override void Withdraw(int amount)
    {
        Console.WriteLine("CurrentAccount withdraw");
    }
}

class SavingsAccount : Account
{
    public override void Withdraw(int amount)
    {
        Console.WriteLine("SavingsAccount withdraw");
    }
}
```

Add a dummy
Withdraw() method

```
Account[] accounts = {
    new CurrentAccount(),
    new SavingsAccount()
};

foreach (Account acc in accounts)
{
    acc.Withdraw(50);
}
```

What is the
problem now?


You can override
but don't have to.

Using an abstract class – Problem solved!

```
abstract class Account
{
    double balance;
    public abstract void withdraw(int amt);
}

class CurrentAccount : Account
{
    public override void Withdraw(int amount)
    {
        Console.WriteLine("CurrentAccount withdraw");
    }
}

class SavingsAccount : Account
{
    public override void Withdraw(int amount)
    {
        Console.WriteLine("SavingsAccount withdraw");
    }
}
```



```
Account[] accounts = {
    new CurrentAccount(),
    new SavingsAccount()
};

foreach (Account acc in accounts)
{
    acc.Withdraw(50);
}
```

Cannot create an instance of **abstract** Account

abstract methods live in **abstract** classes.

Have no code, as they cannot meaningfully be implemented

abstract methods must be overridden in every derived class

Polymorphism with abstract classes

```
abstract class Shape
{
    public abstract void Draw();
}
```

Shape.Draw()
is implemented by the derived type.
**In future, could handle as yet unwritten
derived types**

```
class Rectangle : Shape
{
    public override void Draw()
    {
        // code to draw a rectangle
    }
}
class Circle : Shape
{
    public override void Draw()
    {
        // code to draw a circle
    }
}
```

```
Shape[] shapes = { new Rectangle(), new Circle() };

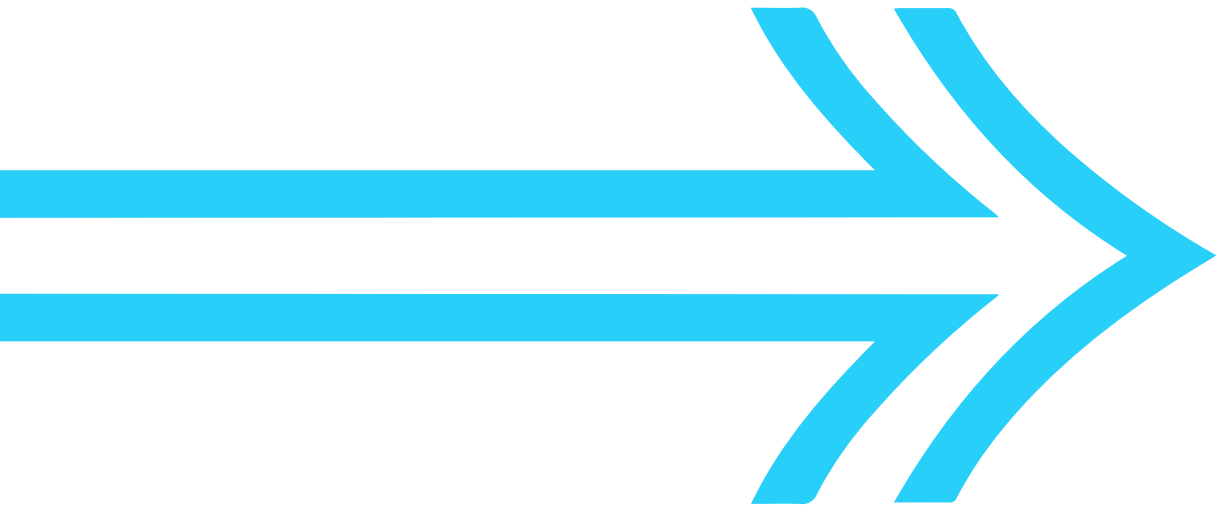
foreach (Shape shape in shapes)
{
    Draw(shape);
}

...

void Draw(Shape shape)
{
    shape.Draw();
}
```



- **In this chapter you learned**
 - What abstract classes are
 - How to use an abstract class
 - Improve design by using abstract classes
 - Polymorphism and abstract classes



LABS



Abstract classes



Duration 30 minutes