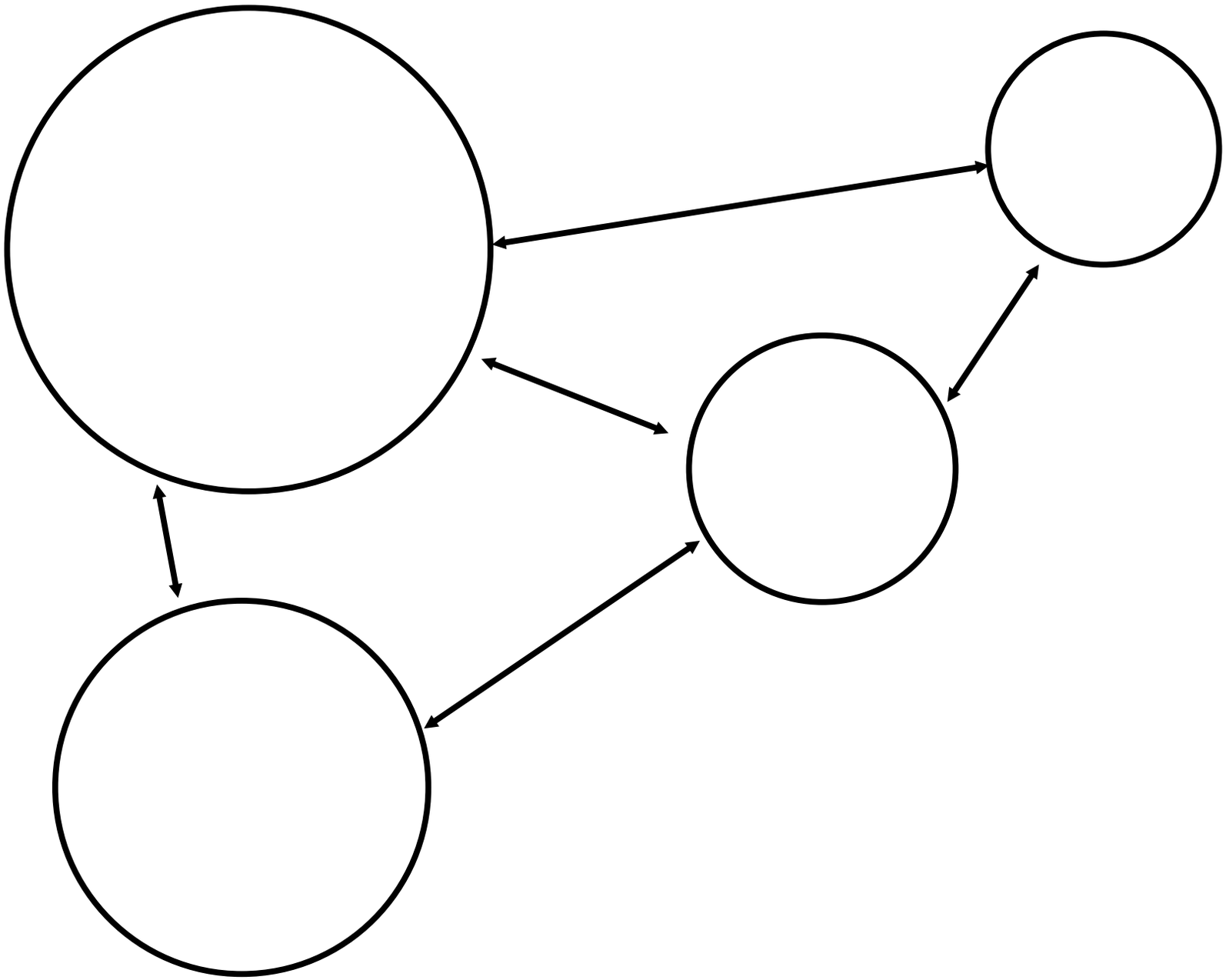
Object Oriented Programming

Procedural (Linear) Programming


```
namespace FunctionExample;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, World!");
        int a = 4;
        Console.WriteLine($"a = {a}");
        int ans = fn1(ref a);
        Console.Write($"a = {a}\nans = {ans}");
        Console.Write("\n\nPress the any key to continue...");
        Console.ReadKey();
    }
    static int fn1(ref int a)
    {
        a = a * 3;
        int b = a - 7;
        return b;
    }
}
```

Object Oriented Programming

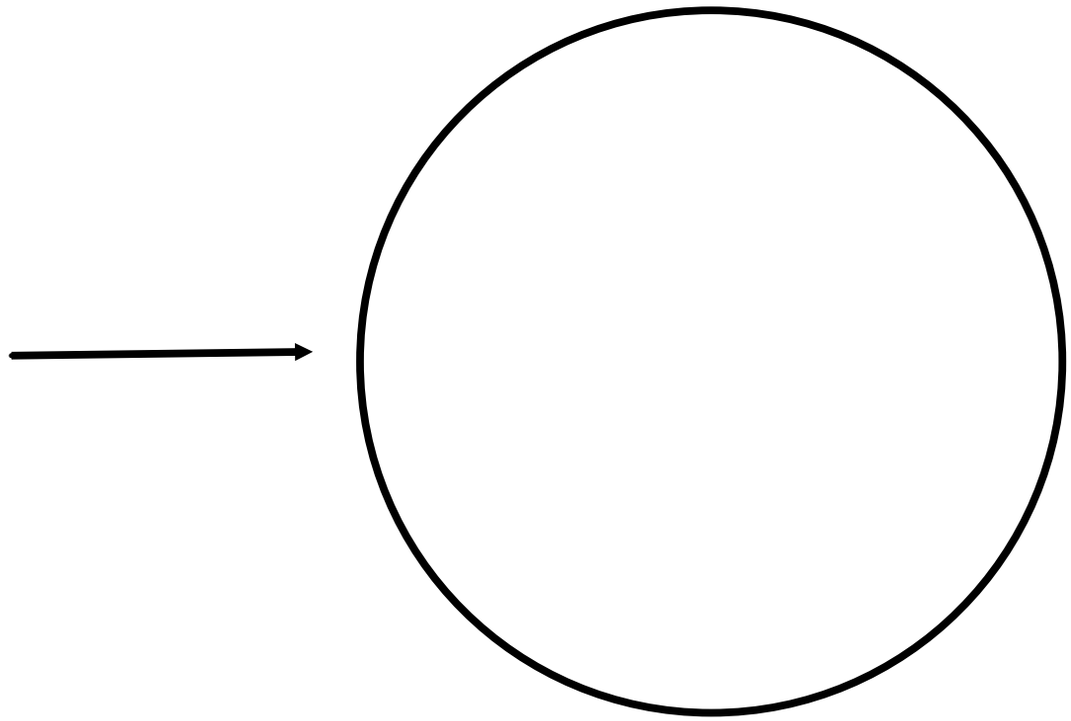


Advantages of OOP

- Encapsulation - everything you need is inside
- Inheritance - you can make objects from other objects (You will explore this more in AP CS.)
- Polymorphism - (Another thing you should explore in AP CS.)
- Encapsulation helps so that you can work in large groups. Each object can be developed on its own.
- Transportability - use a class in other programs. Write it once.

Class

```
using System;

namespace ShapeTester
{
    public class Circle
    {
        private double radius;

        public double Radius
        {
            get { return radius; }
            set { radius = value; }
        }

        public Circle()
        {
            radius = 0;
        }

        public Circle(double radius)
        {
            this.radius = radius;
        }

        public Circle(Circle C)
        {
            radius = C.radius;
        }

        public double Circumference()
        {
            return Math.PI * radius * 2.0;
        }

        public double Area()
        {
            return Math.PI * radius * radius;
        }

        public override String ToString()
        {
            String output = String.Format("Radius:    {0, 8:0.000}\n", radius);
            output += String.Format("Perimeter: {0, 8:0.000}\n", Circumference());
            output += String.Format("Area:      {0, 8:0.000}\n", this.Area());
            return output;
        }
    }
}
```



Class - property

```
using System;

namespace ShapeTester
{
    public class Circle
    {
        private double radius;

        public double Radius
        {
            get { return radius; }
            set { radius = value; }
        }
    }
}
```

Class - Constructor(s)

```
namespace ShapeTester
{
    public class Circle
    {
        public Circle()
        {
            radius = 0;
        }

        public Circle(double radius)
        {
            this.radius = radius;
        }

        public Circle(Circle C)
        {
            radius = C.radius;
        }
    }
}
```

Class - Methods

```
using System;

namespace ShapeTester
{
    public class Circle
    {
        public double Circumference()
        {
            return Math.PI * radius * 2.0;
        }

        public double Area()
        {
            return Math.PI * radius * radius;
        }
    }
}
```

Class - special method ToString()

```
using System;

namespace ShapeTester
{
    public class Circle
    {
        public override String ToString()
        {
            String output = String.Format("Radius:    {0, 8:0.000}\n",
radius);
            output += String.Format("Perimeter: {0, 8:0.000}\n",
Circumference());
            output += String.Format("Area:      {0, 8:0.000}\n",
this.Area());
            return output;
        }
    }
}
```

Class - special method ToString()

```
using System;

namespace ShapeTester
{
    public class Circle
    {
        public override String ToString()
        {
            String output = $"Radius:     {radius, 8:0.000}\n";
            output += String.Format("Perimeter:  {0,8:0.000}\n", Circumference());
            output += $"Area:       {Area(), 8:0.000}\n";
            return output;
        }
    }
}
```

Class - Main()

```
using System;
namespace ShapeTester
{
    internal class ShapeTester
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Shape Tester");
            Circle c = new Circle();
            c.Radius = 10.0;
            Console.WriteLine(c);
            Circle c1 = new Circle(c);
            Console.WriteLine(c1);
            c1.Radius = 5.0;
           Console.WriteLine("c1's new radius = " + c1.Radius + "\n");
            Circle c2 = new Circle(15.0);
            Console.WriteLine(c2);
            //Square s = new Square();
        }
    }
}
```

Class - Output

```
Shape Tester
Radius:       10.000
Perimeter:    62.832
Area:        314.159

Radius:       10.000
Perimeter:    62.832
Area:        314.159

c1's new radius =    5.000

Radius:       15.000
Perimeter:    94.248
Area:        706.858

Press the any key to continue...
```