4. Non-Primitive Types

5. Enums
6. Value Types
7. Reference Types

Enum	A set of name/value pairs (constants).

Use Enums where we need to define related constants.

```
const int RegularAirMail = 1;
const int RegisteredAirMail = 2;
const int Express = 3;


public enum ShippingMethod
{
    RegularAirMail = 1,
    RegisteredAirMail = 2,
    Express = 3;
}
```

```
const int RegularAirMail = 1;
const int RegisteredAirMail = 2;
const int Express = 3;


public enum ShippingMethod
{
    RegularAirMail = 1,
    RegisteredAirMail = 2,
    Express = 3;
}


var method = ShippingMethod.Express;
```

```
const int RegularAirMail = 1;
const int RegisteredAirMail = 2;
const int Express = 3;


public enum ShippingMethod : byte
{
    RegularAirMail = 1,
    RegisteredAirMail = 2,
    Express = 3;
}


var method = ShippingMethod.Express;
```

By default enums are integers

```
namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail,
        RegisteredAirMail,
        Express
    }

    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

As enum is a new kind of user define data type – we need to define it at namespace level.

If we do not assign any value to enum at the time of Declaration – first one be 1 and rest will be incremented by one.
by one.

Program.cs

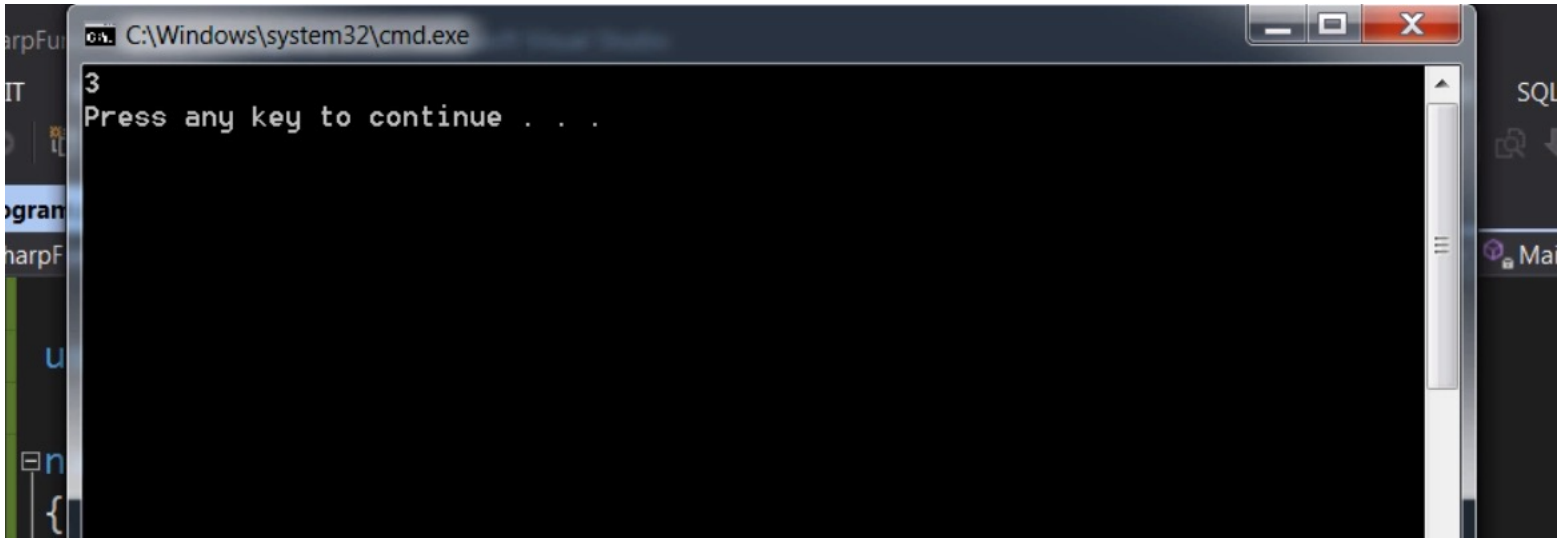
CSharpFundamentals.Program

```
namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;

        }
    }
}
```

Program.cs

CSharpFundamentals.Program

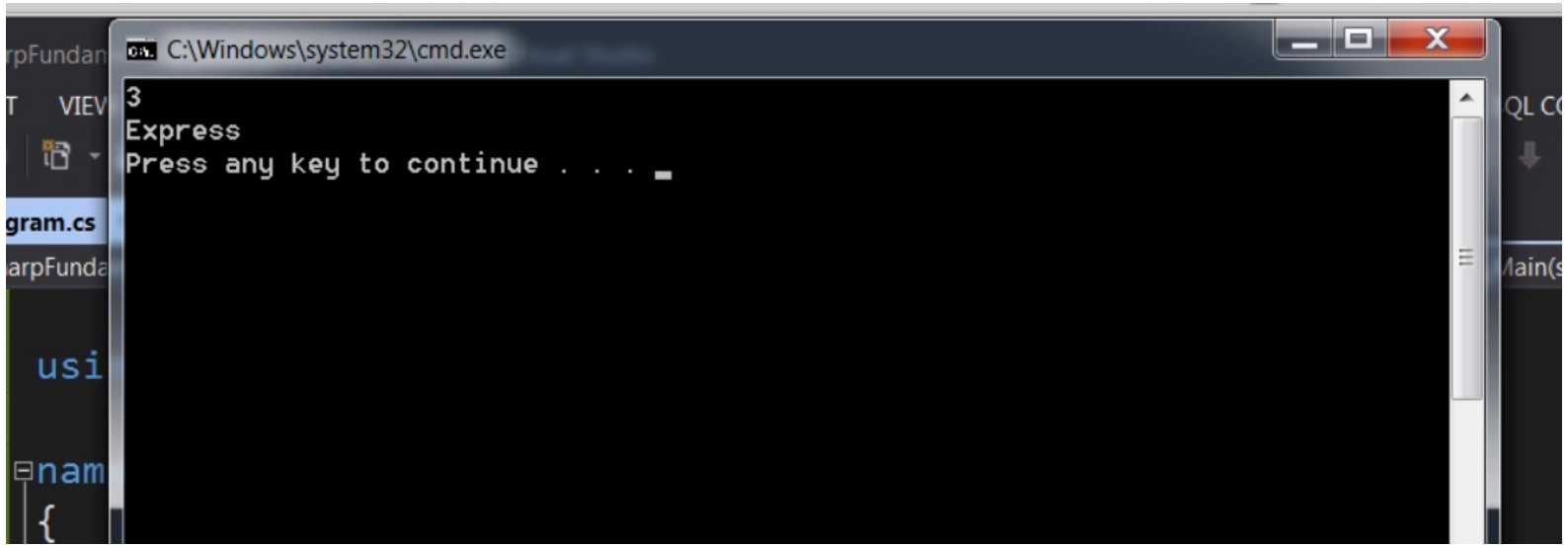
```
using System;

namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;
            Console.WriteLine((int)method);
        }
    }
}
```


C:\Windows\system32\cmd.exe
3
Press any key to continue . . .

Program.cs

CSharpFundamentals.Program

```
using System;

namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;
            Console.WriteLine((int)method);

            var methodId = 3;
            Console.WriteLine((ShippingMethod)methodId);
        }
    }
}
```


C:\Windows\system32\cmd.exe
3
Express
Press any key to continue . . .

Program.cs

CSharpFundamentals.Program

```
using System;

namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;
            Console.WriteLine((int)method);

            var methodId = 3;
            Console.WriteLine((ShippingMethod)methodId);

            Console.WriteLine(method.ToString());
        }
    }
}
```

Program.cs

CSharpFundamentals.Program

```
using System;

namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;
            Console.WriteLine((int)method);

            var methodId = 3;
            Console.WriteLine((ShippingMethod)methodId);

            Console.WriteLine(method);
        }
    }
}
```

```
ace CSharpFundamentals

blic enum ShippingMethod

  RegularAirMail = 1,
  RegisteredAirMail = 2,
  Express = 3


ass Program

  static void Ma
  {
      var method
      Console.Wri

      var methodI
      Console.Wri

      Console.Wri

      var methodN
      Enum.Parse()
```

([NotNull] Type enumType, [NotNull] string value):object

Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

enumType: An enumeration type.

([NotNull] Type enumType, [NotNull] string value, bool ignoreCase):object

global
ignoreCase:
int
long
method
methodId
methodName
new
null
object
out

Local variable string methodName

```
ace CSharpFundamentals

blic enum ShippingMethod

  RegularAirMail = 1,
  RegisteredAirMail = 2,
  Express = 3


ass Program

  static void Ma
  {
      var method = ShippingMethod.Express
      Console.WriteLine((int)method);

      var methodId = 3;
      Console.WriteLine((ShippingMethod)m

      Console.WriteLine(method.ToString()

      var methodName = "Express";
      Enum.Parse(typeof(ShippingMethod),
  }
```

([NotNull] Type enumType, **[NotNull] string value**):object

Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

value: A string containing the name or value to convert.

([NotNull] Type enumType, **[NotNull] string value**, bool ignoreCase):object

- global
- ignoreCase:
- **int**
- **long**
- method
- methodId
- methodName
- **new**
- **null**
- **object**
- **out**

Local variable string methodName

```
ace CSharpFundamentals

blic enum ShippingMethod

  RegularAirMail = 1,
  RegisteredAirMail = 2,
  Express = 3


ass Program

  static void Main(string[] args)
  {
      var method = ShippingMethod.Express;
      Console.WriteLine((int)method);

      var methodId = 3;
      Console.WriteLine((ShippingMethod)methodId);

ype enumType, [NotNull] string value):object
e string representation of the name or numeric value of one or
erated constants to an equivalent enumerated object.
ring containing the name or value to convert.
                                                  String());
ype enumType, [NotNull] string value, bool ignoreCase):object
      Enum.Parse(typeof(ShippingMethod), methodName)
```

```
ace CSharpFundamentals

blic enum ShippingMethod

  RegularAirMail = 1,
  RegisteredAirMail = 2,
  Express = 3


ass Program

  static void Main(string[] args)
  {
      var method = ShippingMethod.Express;
      Console.WriteLine((int)method);

      var methodId = 3;
      Console.WriteLine((ShippingMethod)methodId);

      Console.WriteLine(method.ToString());

      var methodName = "Express";
      (ShippingMethod)Enum.Parse(typeof(ShippingMethod), methodName)
```

```
  RegularAirMail = 1,
  RegisteredAirMail = 2,
  Express = 3


ass Program

  static void Main(string[] args)
  {
      var method = ShippingMethod.Express;
      Console.WriteLine((int)method);

      var methodId = 3;
      Console.WriteLine((ShippingMethod)methodId);

      Console.WriteLine(method.ToString());

      var methodName = "Express";
      var shippingMethod = (ShippingMethod) Enum.Parse(typeof (ShippingMethod), methodName);
      enum CSharpFundamentals.ShippingMethod

  }
```

Types

- int
- char
- float
- bool
- classes
- structures
- arrays (System.Array)
- strings (System.String)

Types

Structures

- Primitive types
- Custom structures

Classes

- Arrays
- Strings
- Custom classes

Value Types

Structures

- Allocated on stack
- Memory allocation done automatically
- Immediately removed when out of scope

Reference Types

Classes

- You need to allocate memory
- Memory allocated on heap
- Garbage collected by CLR

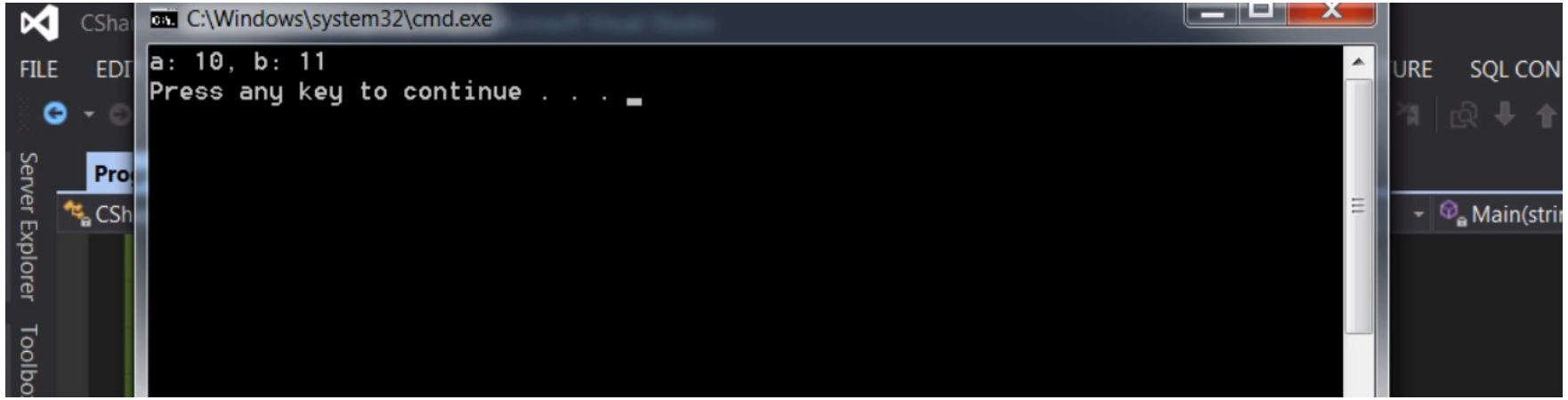
Copying value types and Reference Types

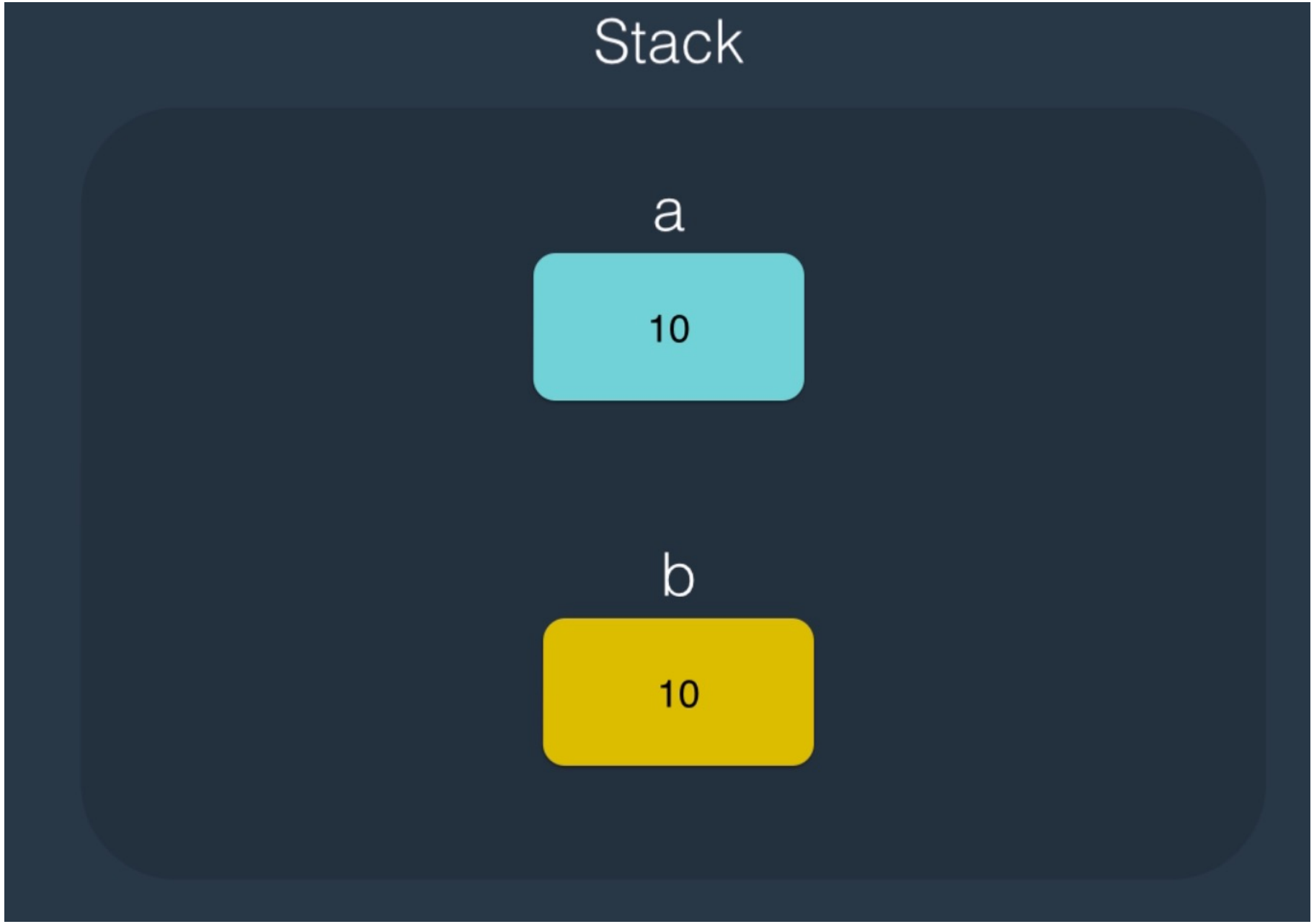
```
var anotherObject = someObject;
```

Program.cs

CSharpFundamentals.Program Main(string[] args

```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var a = 10;
            var b = a;
            b++;
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));
        }
    }
}
```

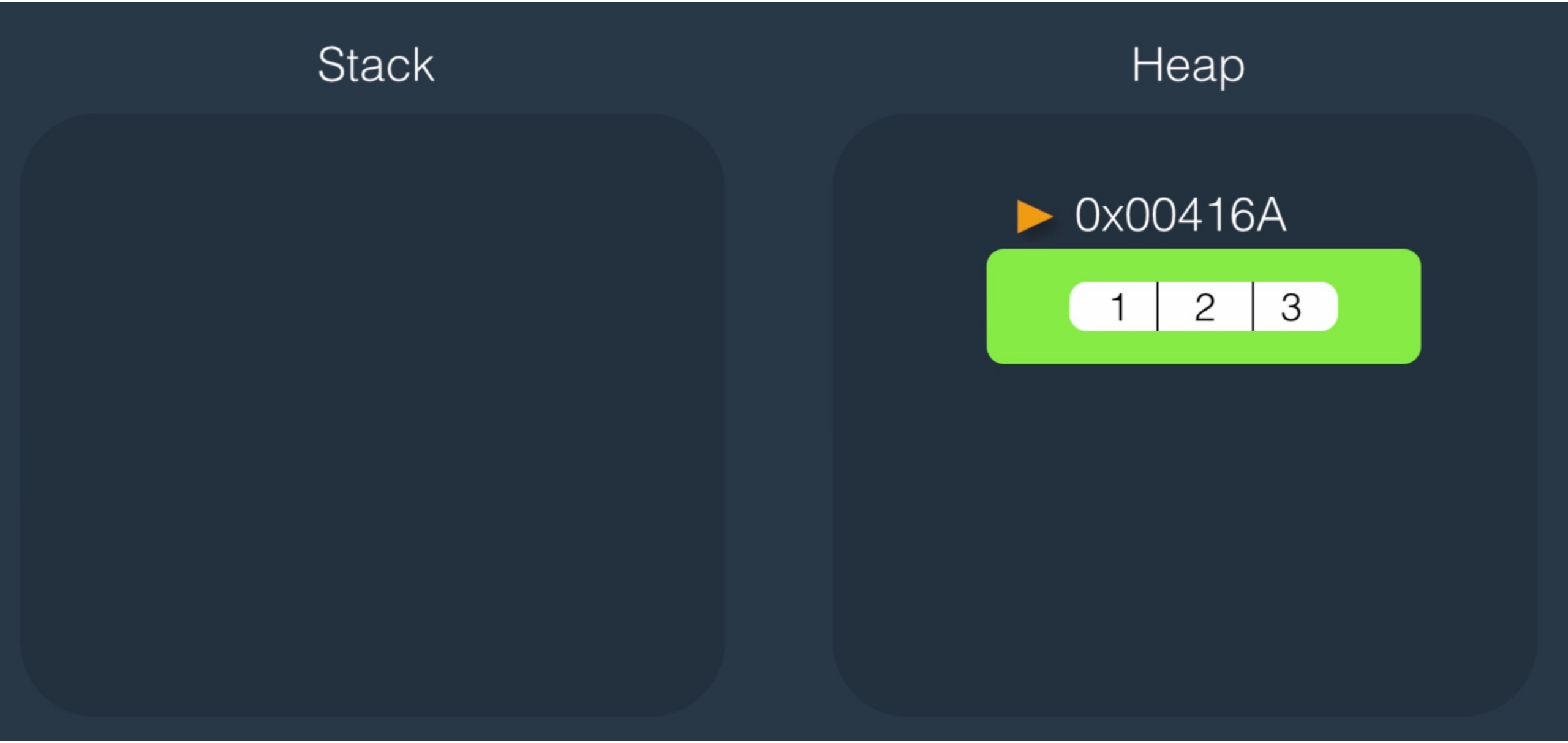

C:\Windows\system32\cmd.exe
a: 10, b: 11
Press any key to continue . . .
FILE
SQL CON
Server Explorer
Main(stri

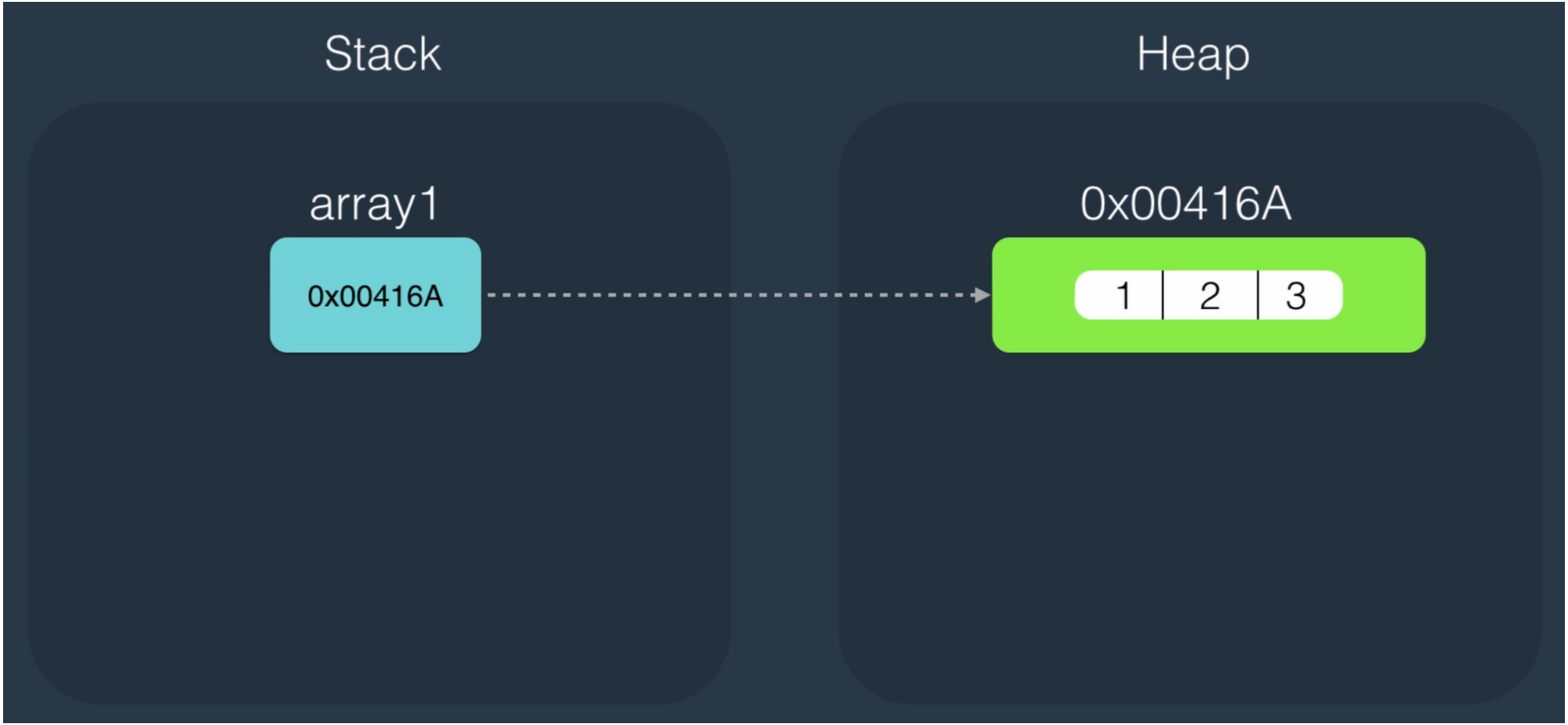

Stack
a
10
b
10

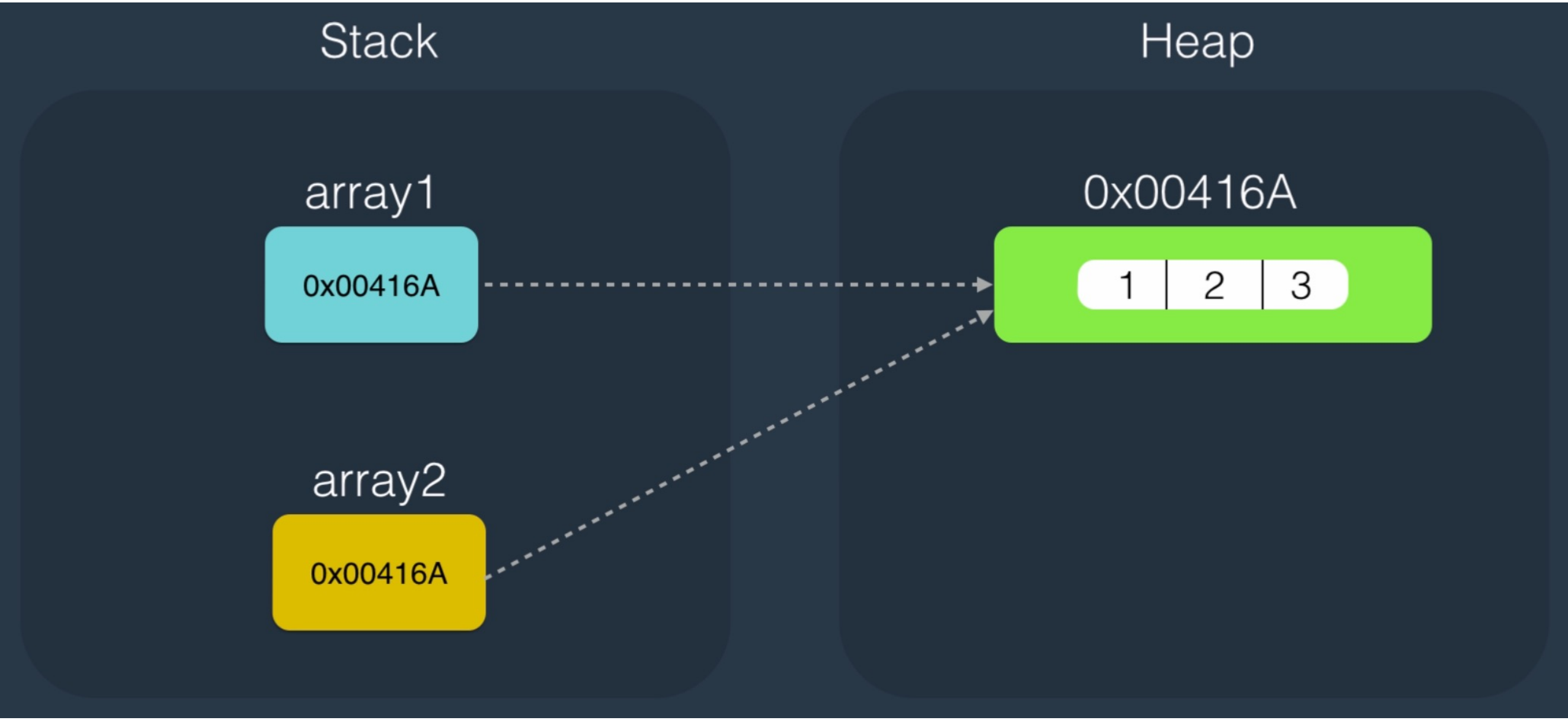
Program.cs

CSharpFundamentals.Program Main(string[] args)

```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var a = 10;
            var b = a;
            b++;
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));

            var array1 = new int[3] {1, 2, 3};
            var array2 = array1;
            array2[0] = 0;

        }
    }
}
```


Stack
Heap
0x00416A
1
2
3

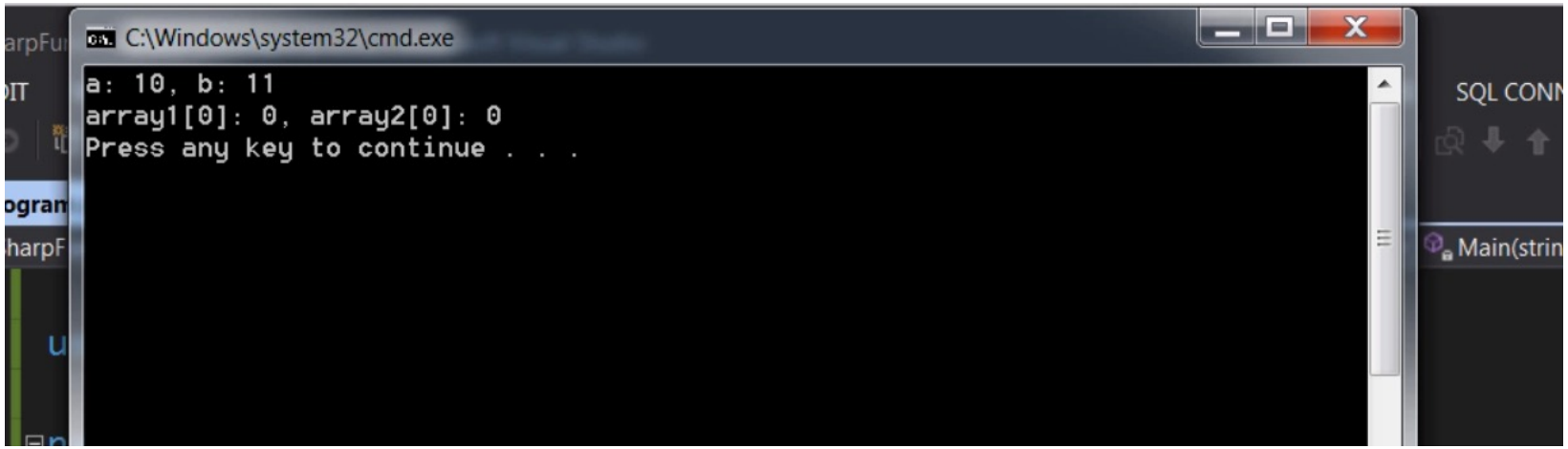

Stack
Heap
array1
0x00416A
0x00416A
1
2
3


Stack
Heap
array1
0x00416A
array2
0x00416A
0x00416A
1
2
3

Program.cs

CSharpFundamentals.Program Main(string[] args)

```
using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var a = 10;
            var b = a;
            b++;
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));

            var array1 = new int[3] {1, 2, 3};
            var array2 = array1;
            array2[0] = 0;
            Console.WriteLine(string.Format("array1[0]: {0}, array2[0]: {1}", array1[0], array2[0]));
        }
    }
}
```


C:\Windows\system32\cmd.exe
a: 10, b: 11
array1[0]: 0, array2[0]: 0
Press any key to continue . . .
SQL CONN
Main(strin

Program.cs

CSharpFundamentals.Program

```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
}
```

Program.cs

CSharpFundamentals.Program

```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
}
```

Program.cs

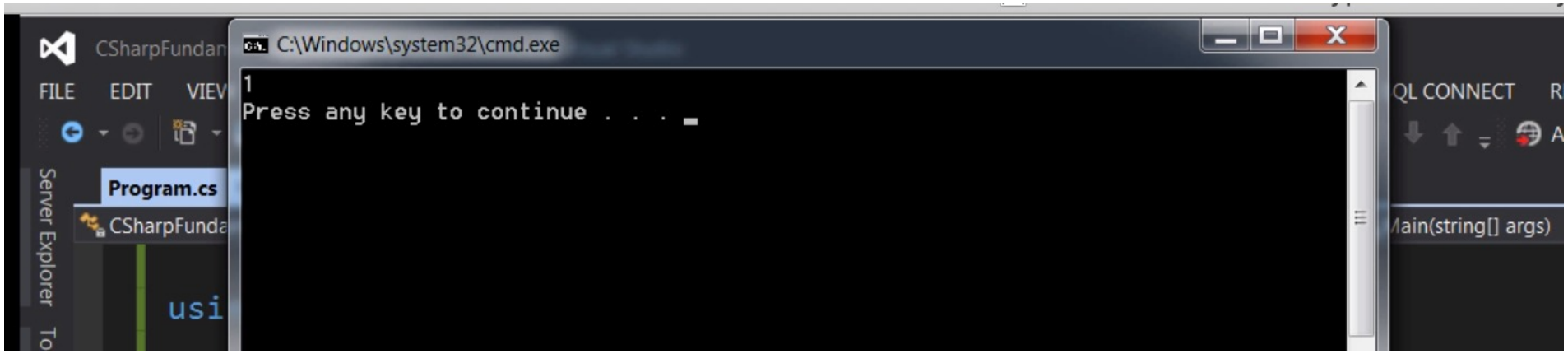
CSharpFundamentals.Program

```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);

        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
```

Program.cs

CSharpFundamentals.Program

```
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
            Console.WriteLine(number);
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
```


C:\Windows\system32\cmd.exe
1
Press any key to continue . . .
CSharpFundan
FILE
EDIT
VIEW
Program.cs
CSharpFunda
usi
Server Explorer
QL CONNECT
Main(string[] args)

Program.cs

CSharpFundamentals.Program

```
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
            Console.WriteLine(number);

            var person = new Person() {Age = 20};

        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }  Program
}
```

Program.cs

CSharpFundamentals.Program

```
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
            Console.WriteLine(number);

            var person = new Person() {Age = 20};

        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    } Program
}
```

Program.cs

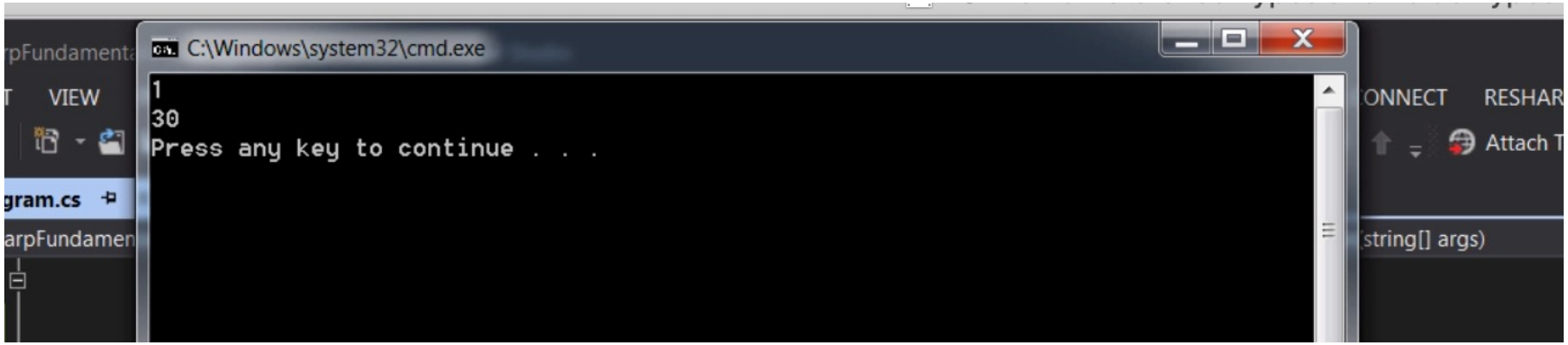
CSharpFundamentals.Program

```
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
            Console.WriteLine(number);

            var person = new Person() {Age = 20};

        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    } Program
}
```

Program.cs

CSharpFundamentals.Program

```
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
            Console.WriteLine(number);

            var person = new Person() {Age = 20};
            MakeOld(person);
            Console.WriteLine(person.Age);
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    } Program
}
```


C:\Windows\system32\cmd.exe
1
30
Press any key to continue . . .
VIEW
gram.cs
ONNECT
RESHAR
Attach T
string[] args)

Classes

Strings
Arrays
Custom Classes

Structures

Primitive Types
Custom Structures