Array/list of classes/structs

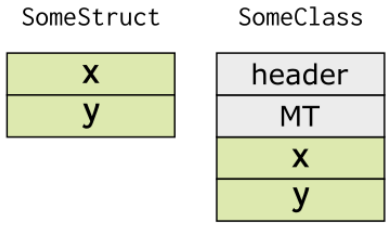
Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

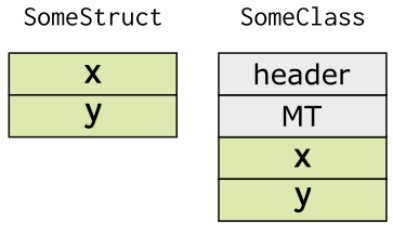
We know already the layout of those instances:



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

We know already the layout of those instances:

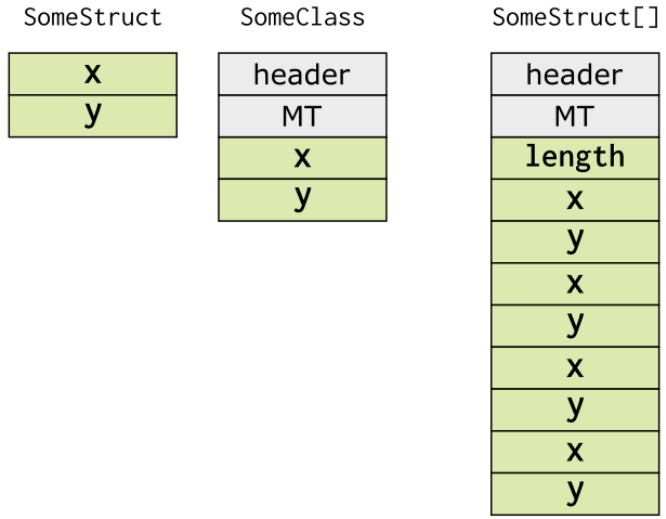


Value type fields of reference type instance are becoming part (inlined) of that instance.

Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

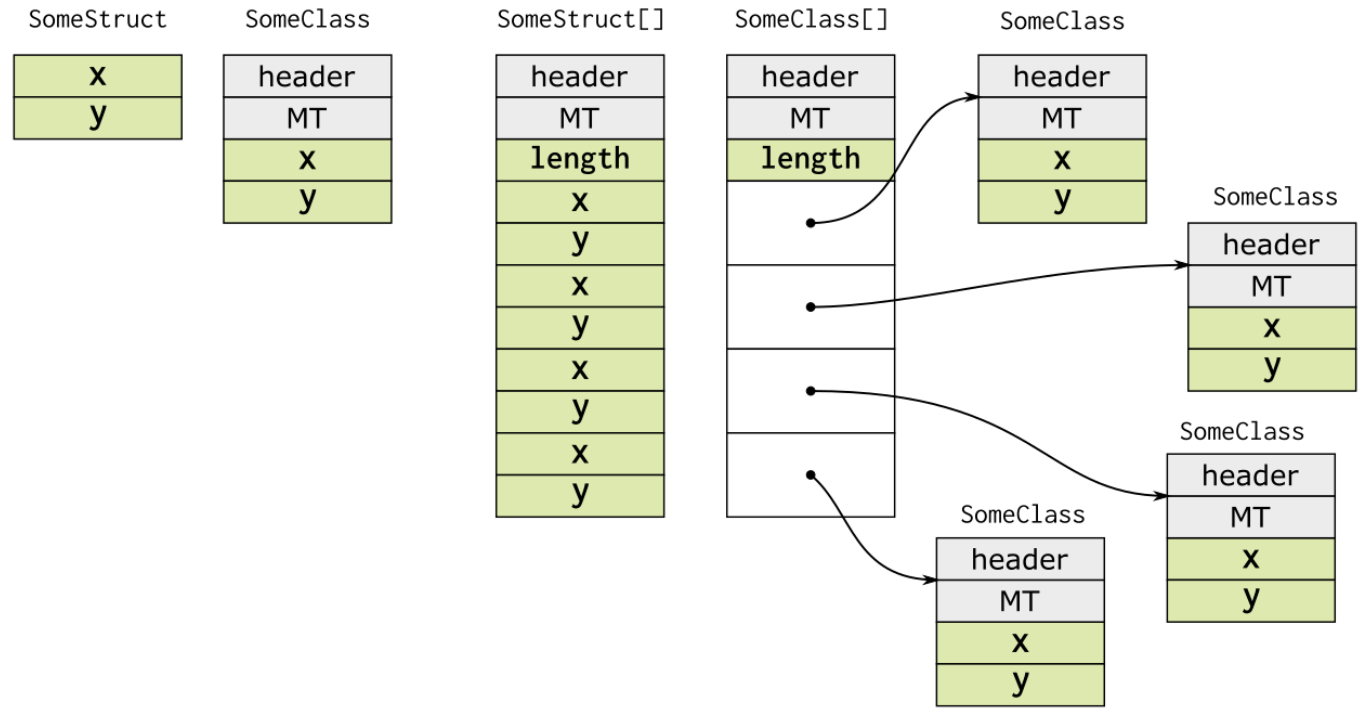
Array of structs - the elements are the structs themselves:



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

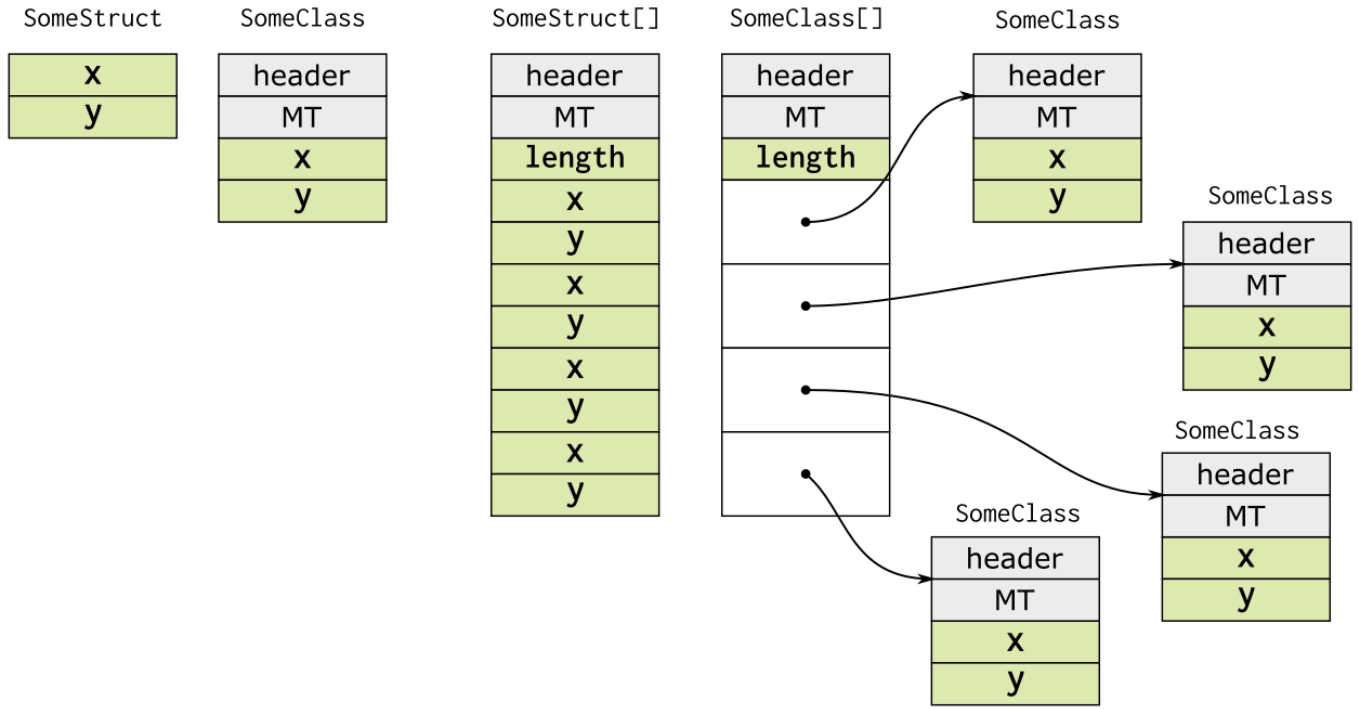
Array of structs - the elements are references to the classes instances:



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

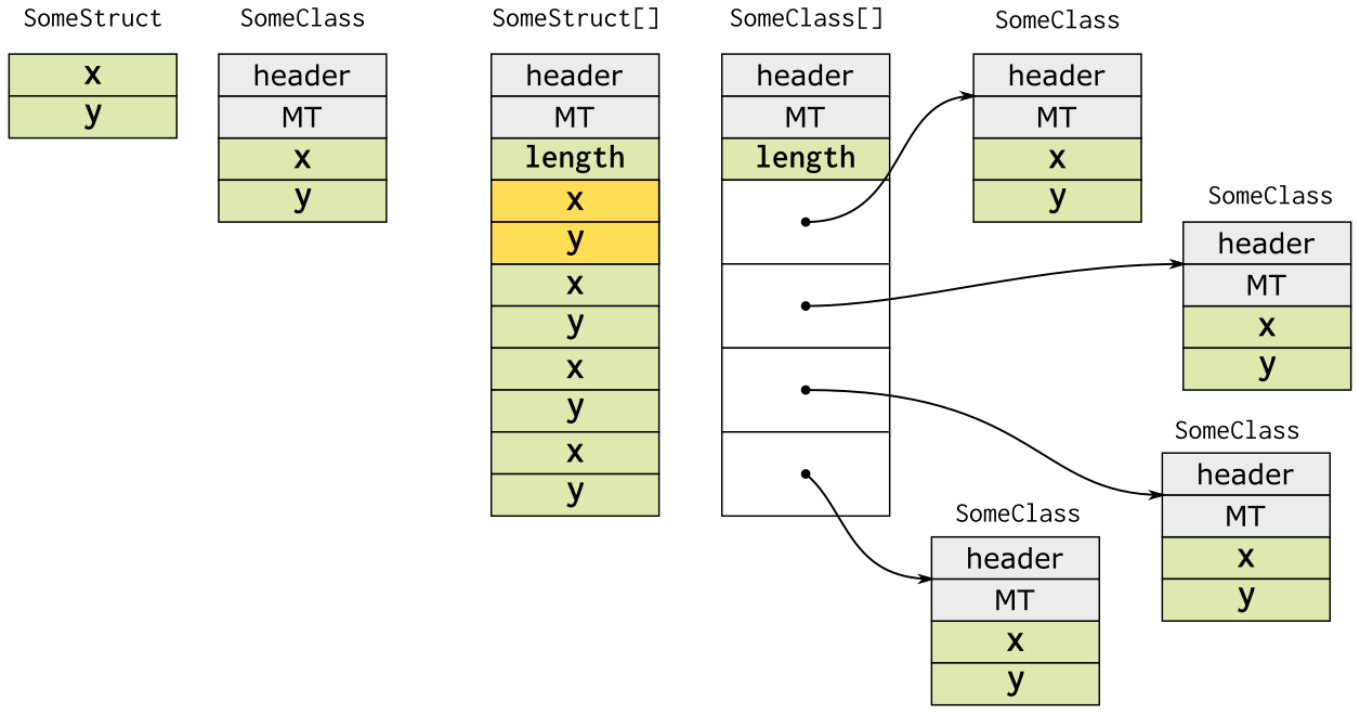
```
var a = new SomeStruct[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

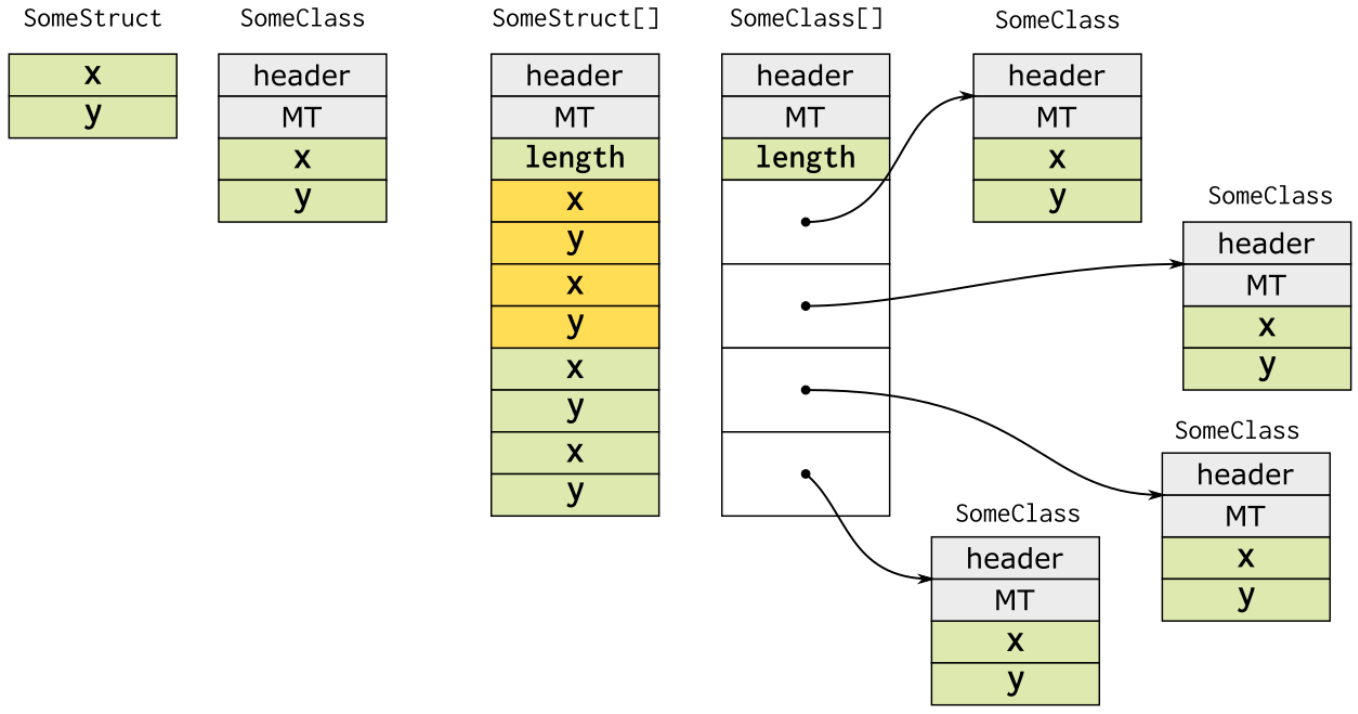
```
var a = new SomeStruct[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

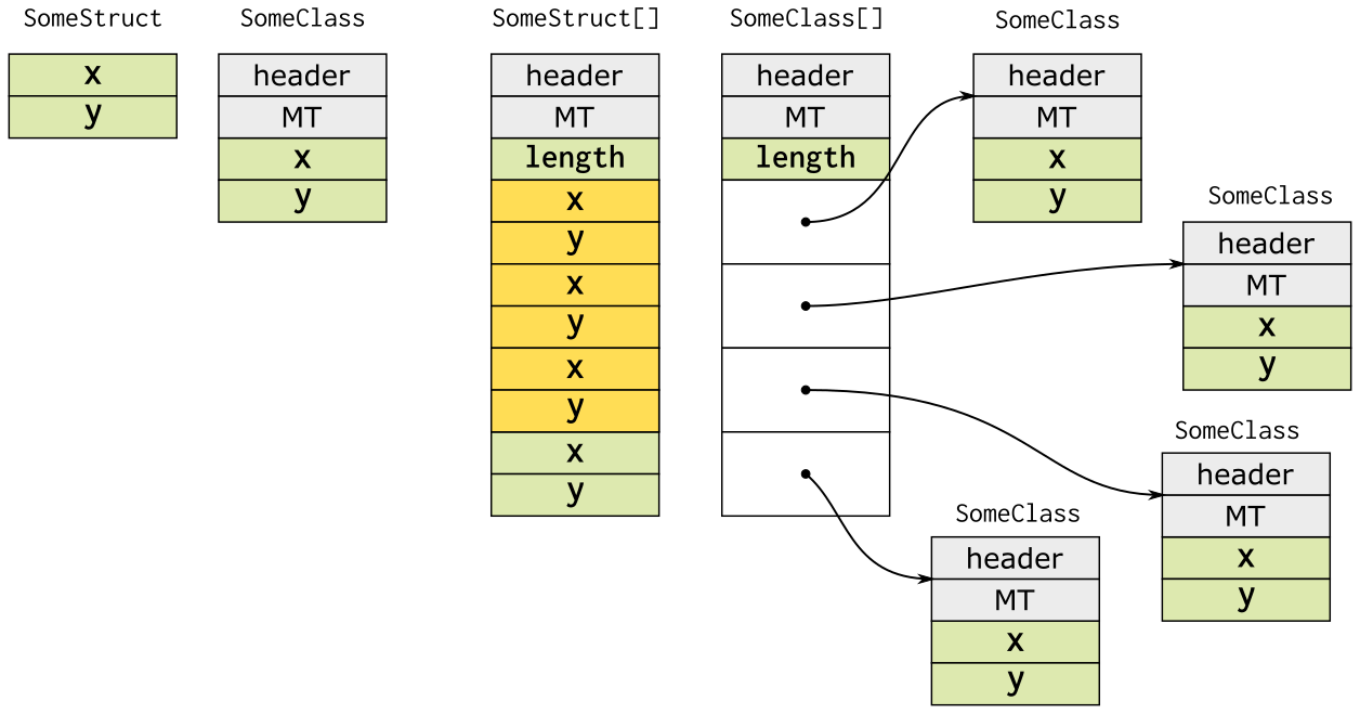
```
var a = new SomeStruct[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

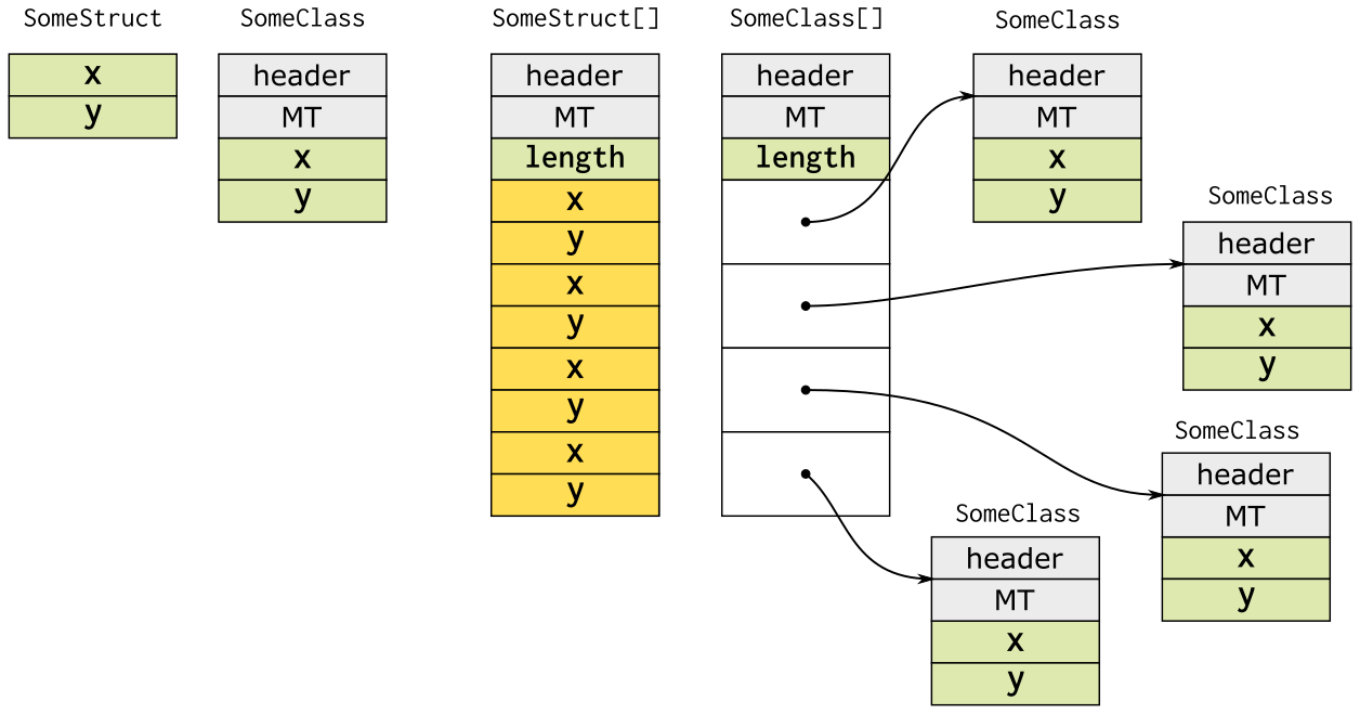
```
var a = new SomeStruct[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

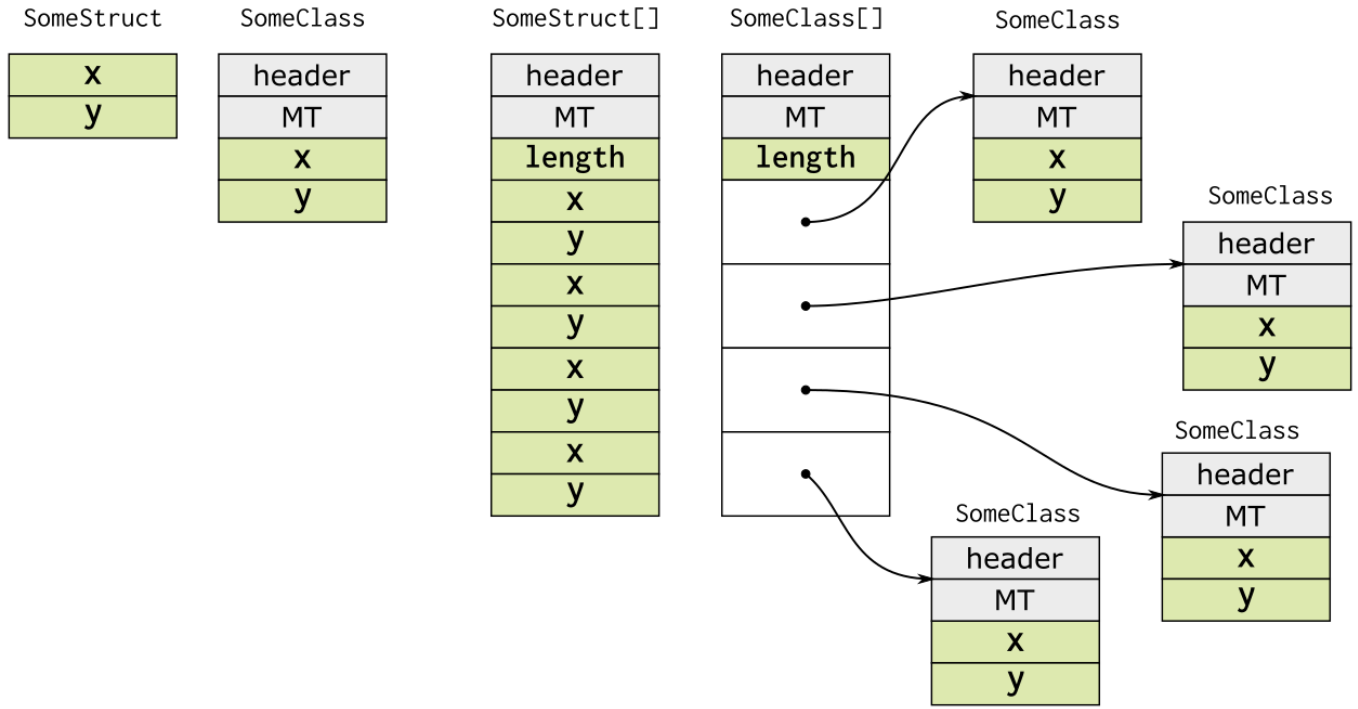
```
var a = new SomeStruct[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

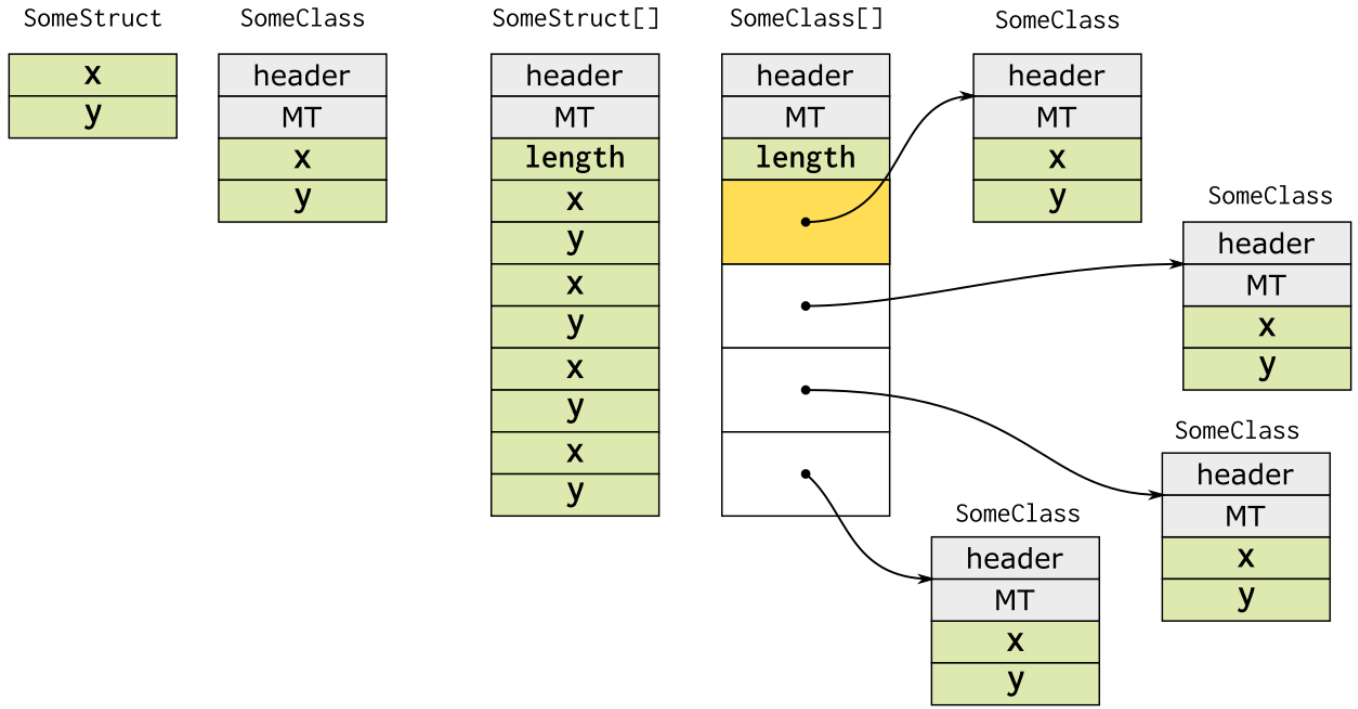
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

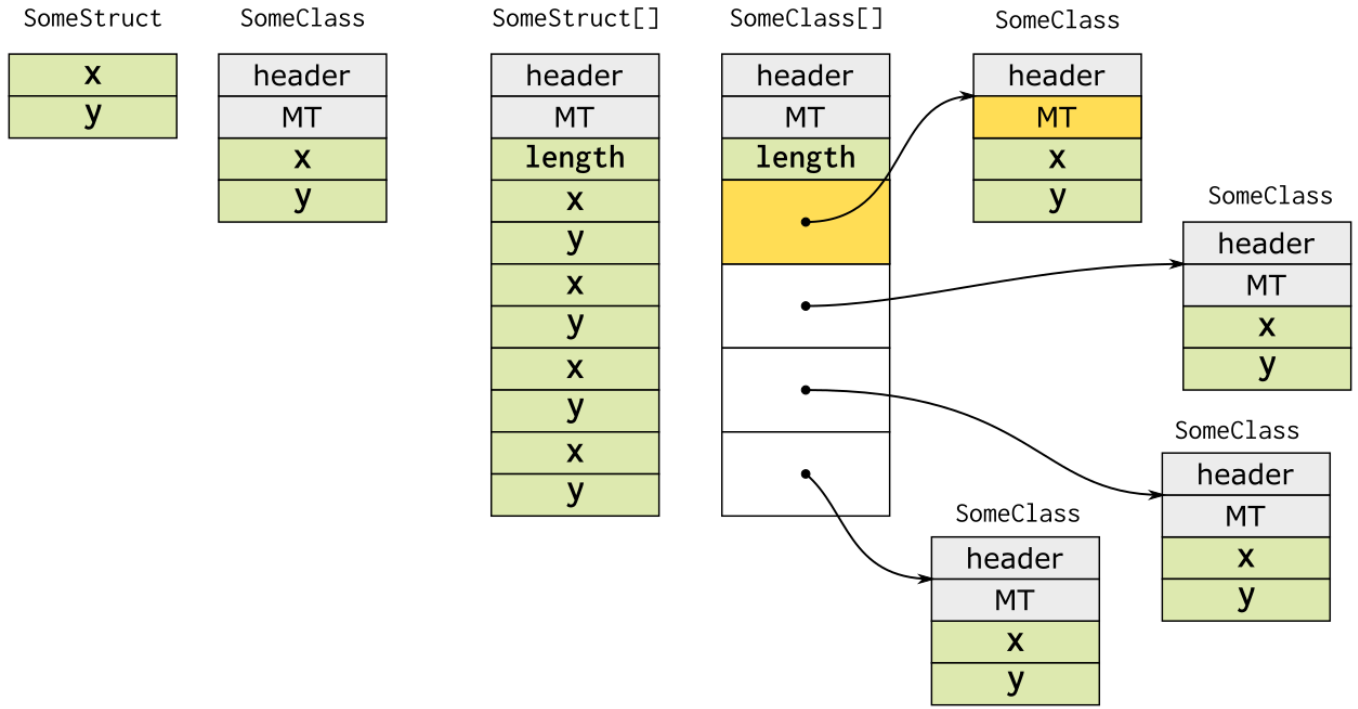
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

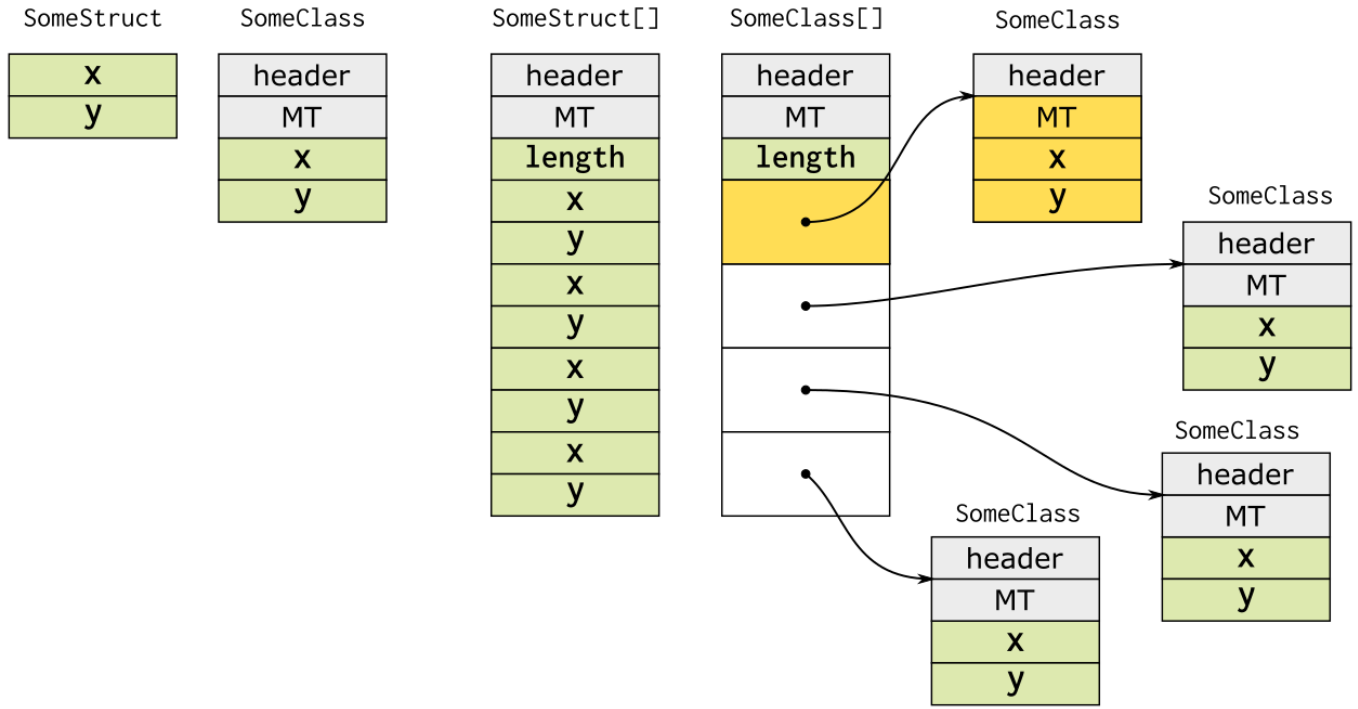
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

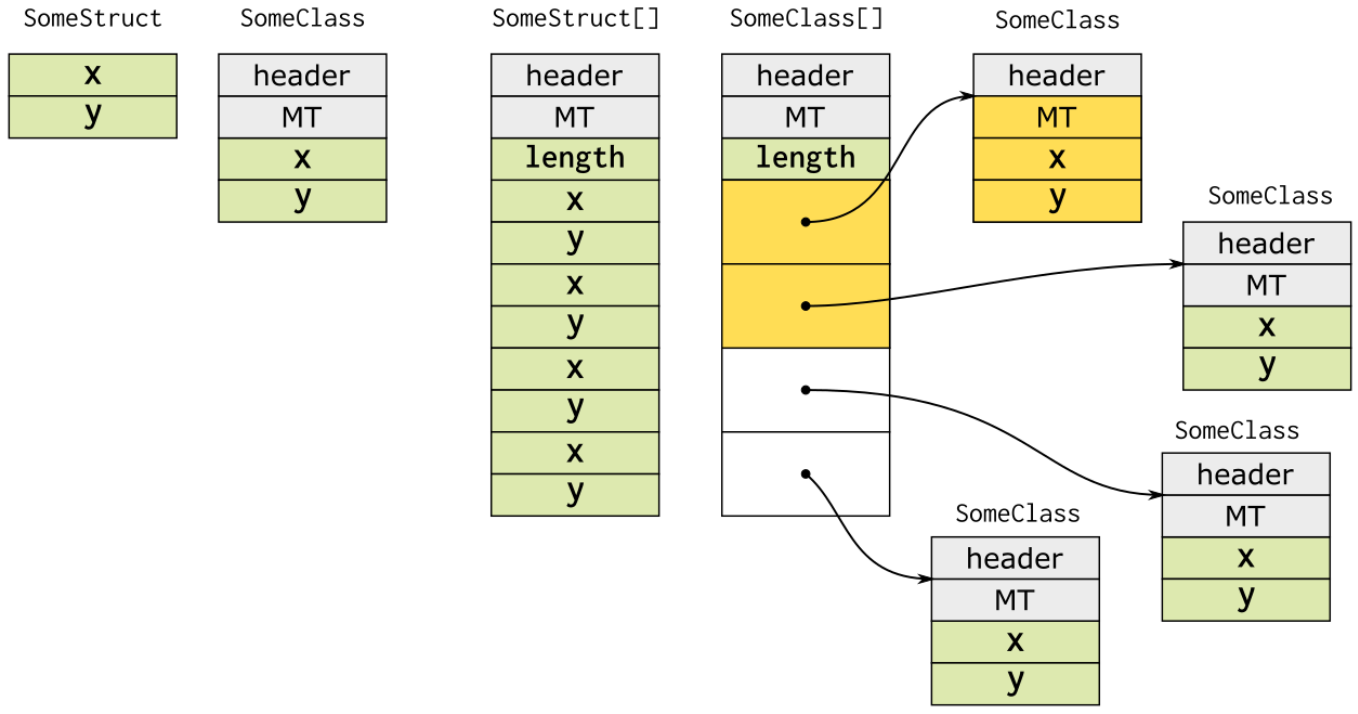
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

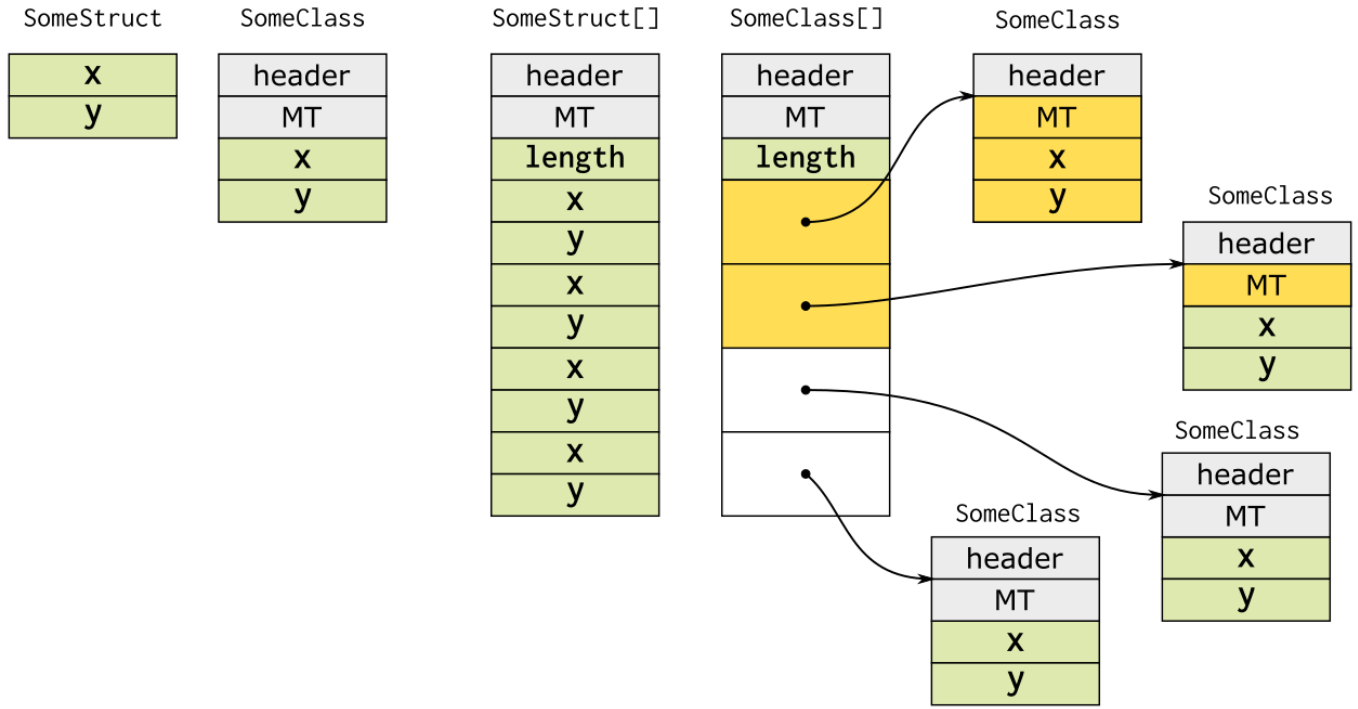
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

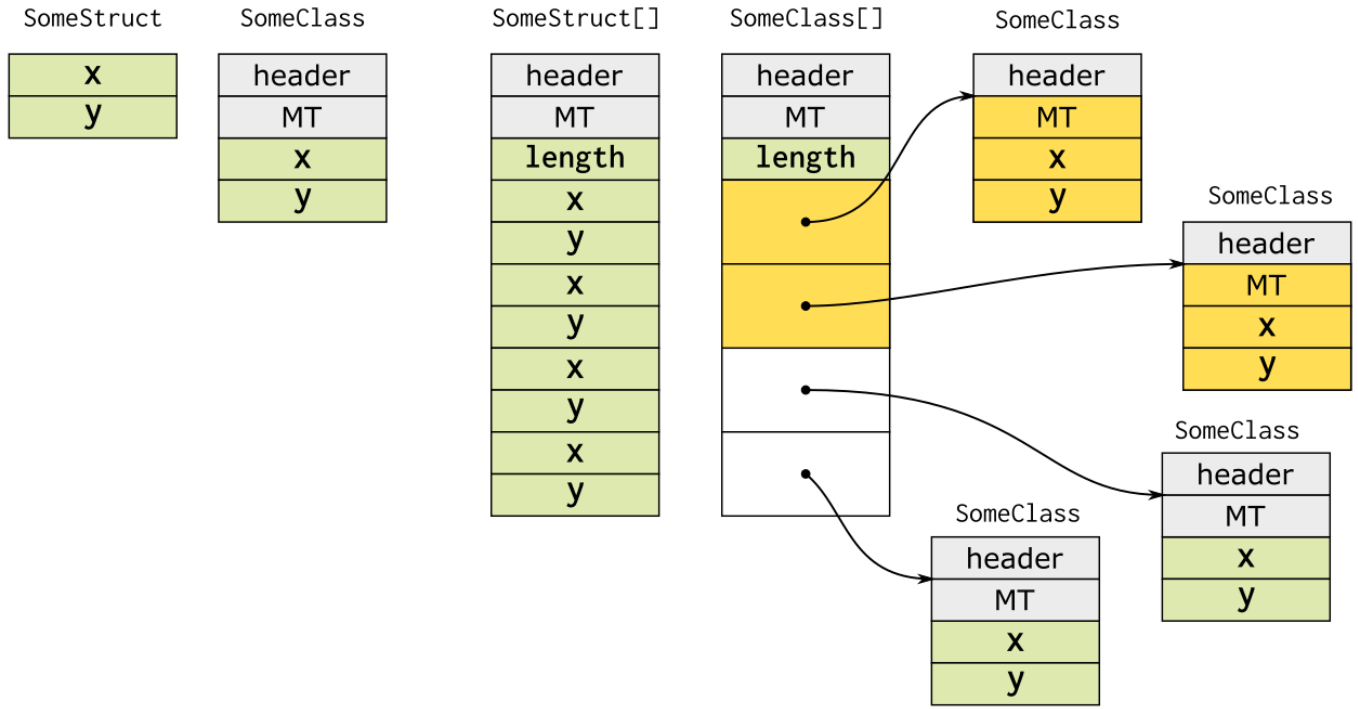
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

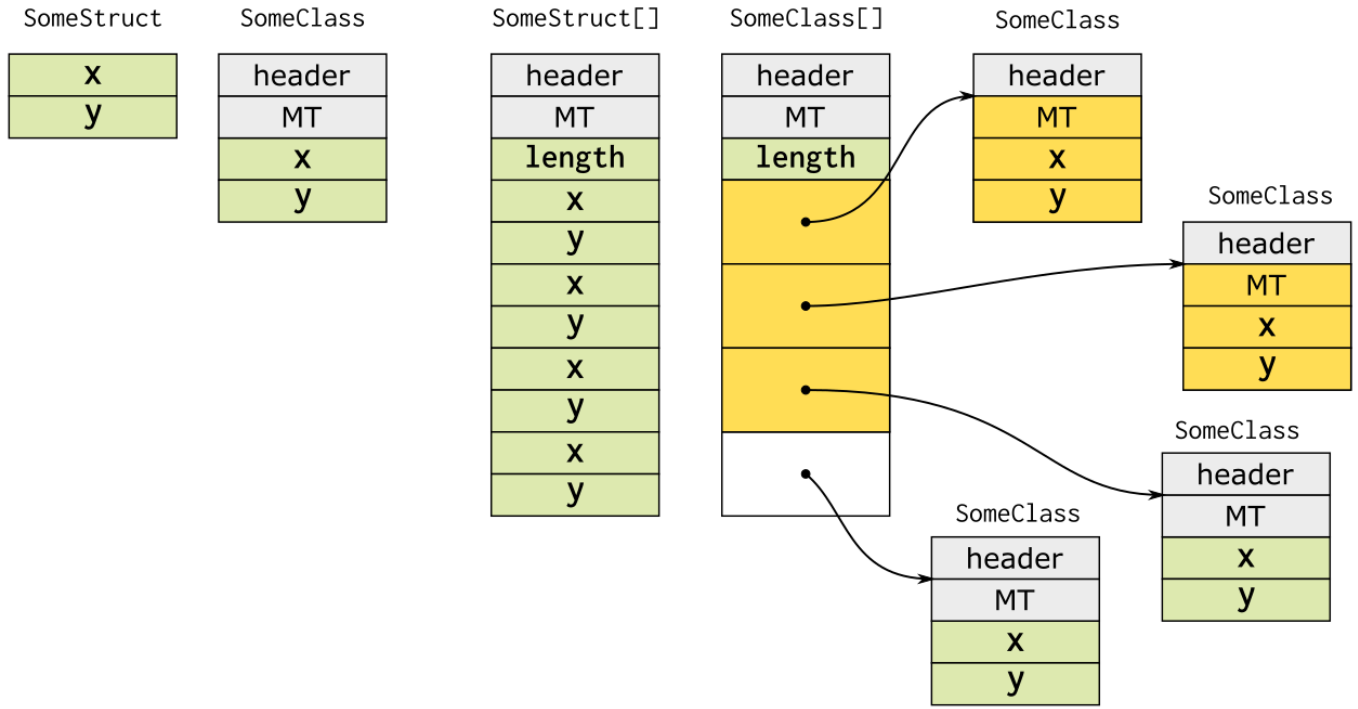
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

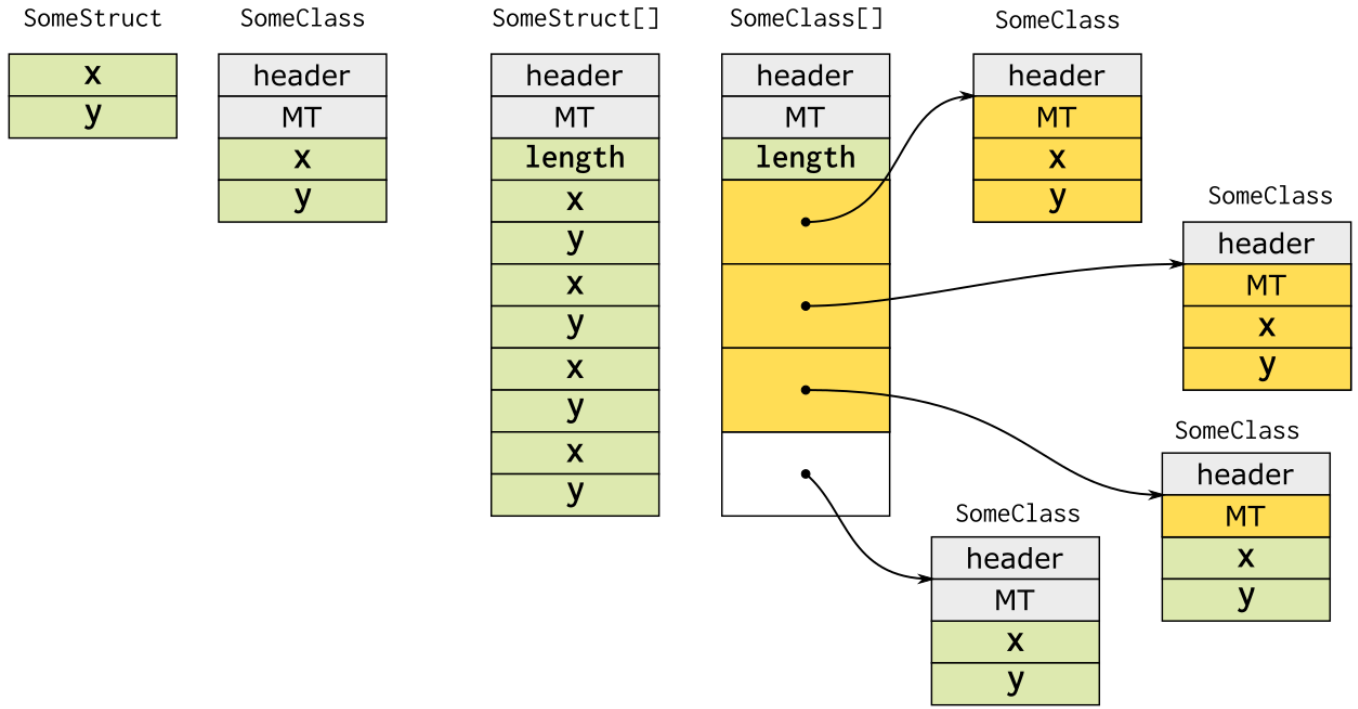
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

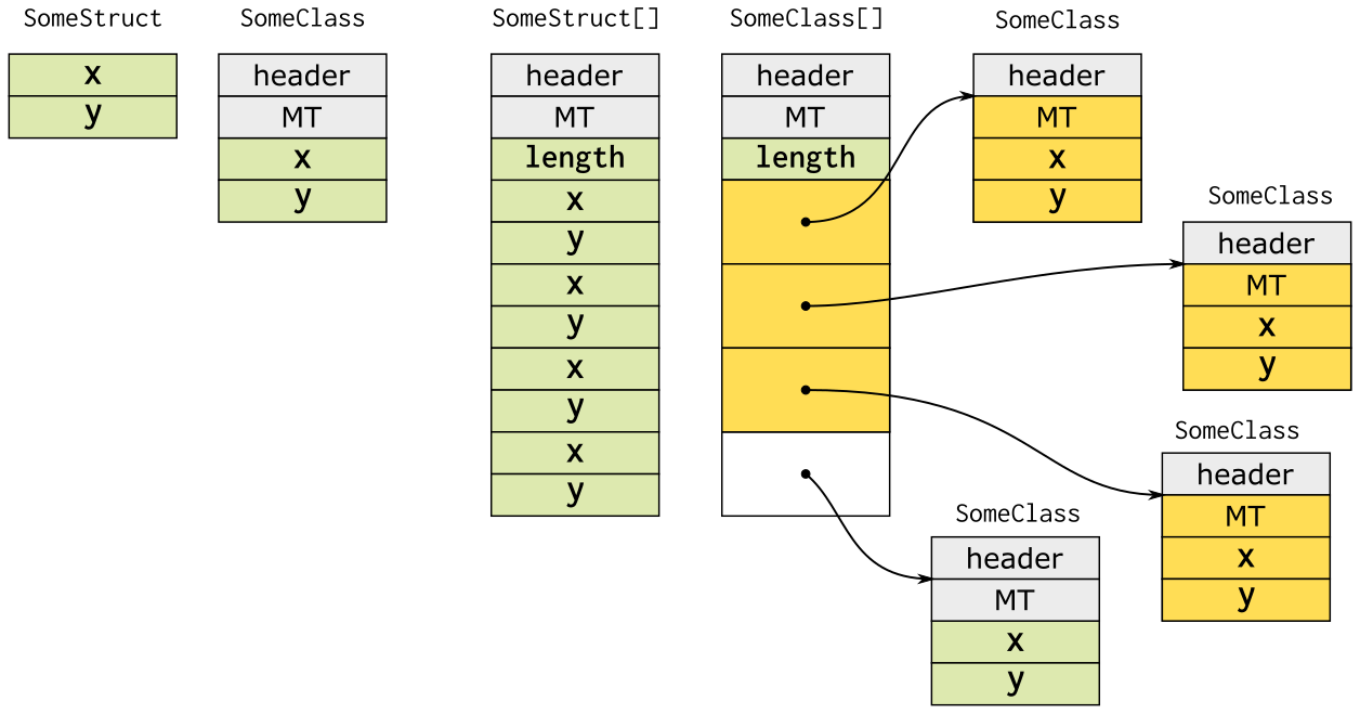
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

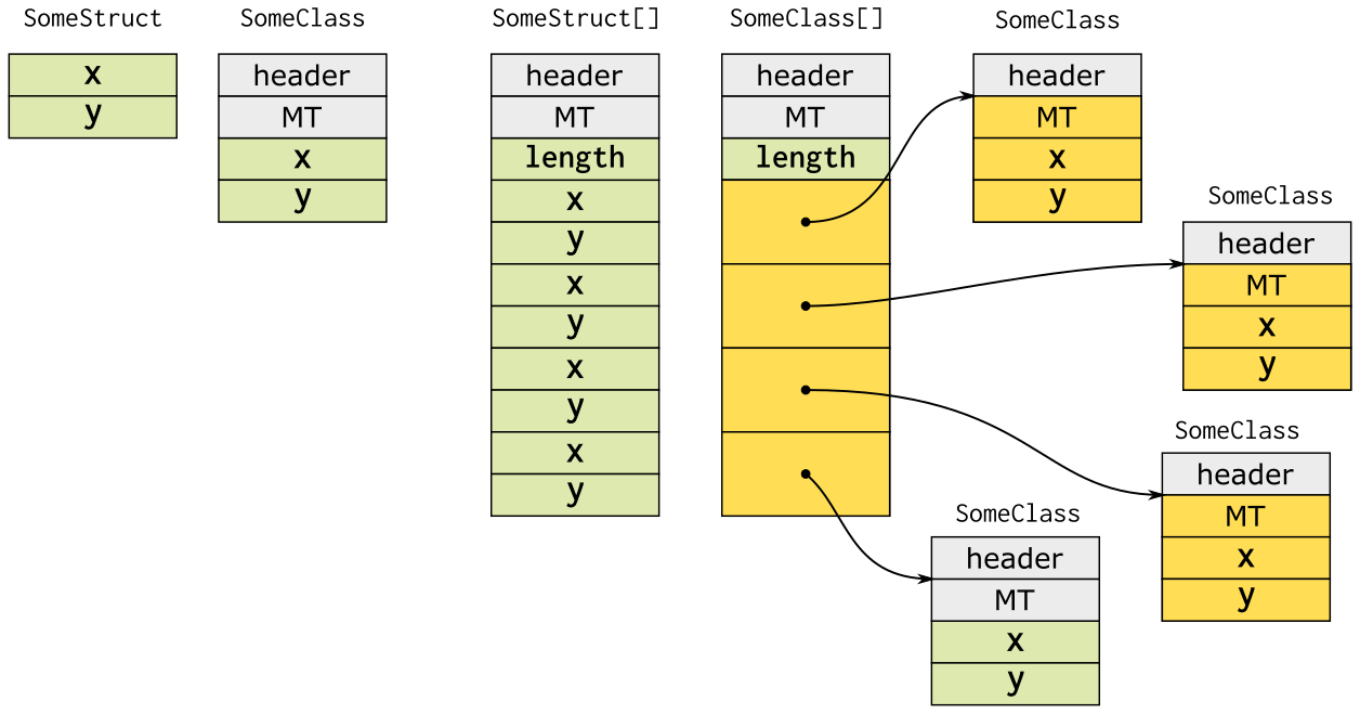
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

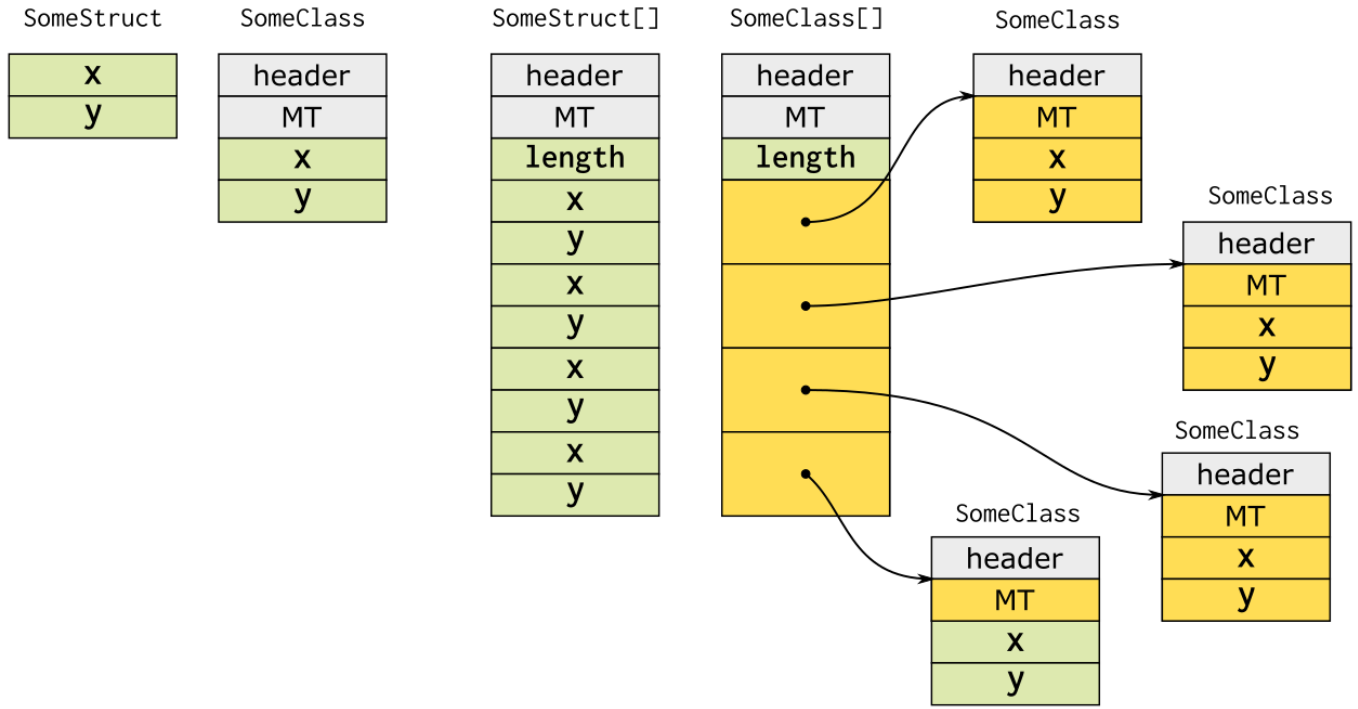
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

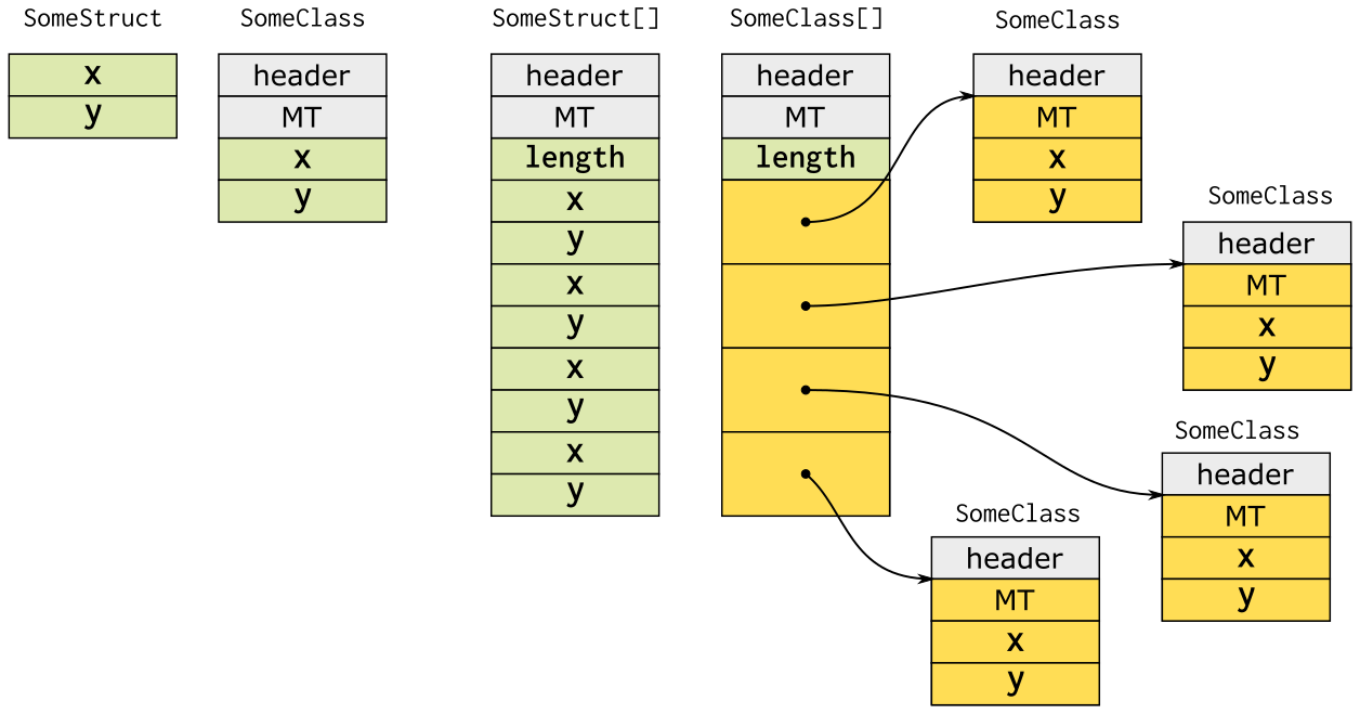
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



Array of structs vs array of classes

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```

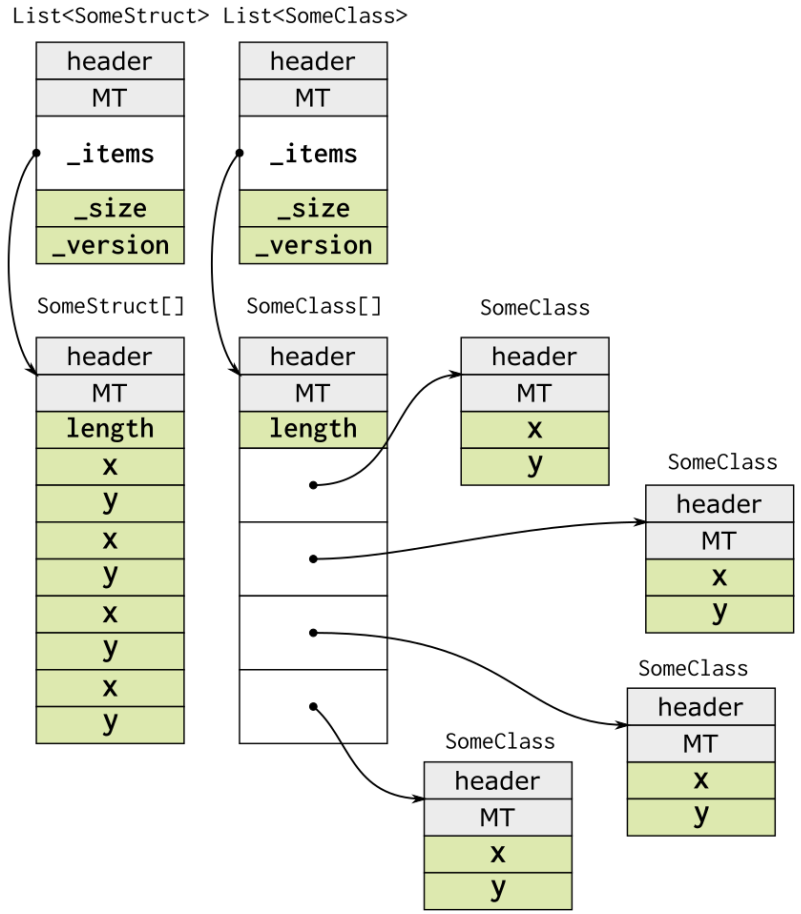
```
var a = new SomeClass[] ...;
for (int i = 0; i < 4; ++i)
 Console.WriteLine($"{a[i]} {a[i]}");
```



List of structs vs array of classes

Lists are just wrappers around arrays, so the story is the same:

```
class SomeClass
{
    public int x;
    public int y;
}

struct SomeStruct
{
    public int x;
    public int y;
}
```



Array of structs vs array of classes

- a nice example of *leaking abstraction* that we can use if caring about data locality
  - may be applied to many collections - most are baked by arrays

Array of structs vs array of classes

- a nice example of *leaking abstraction* that we can use if caring about data locality
  - may be applied to many collections - most are baked by arrays
- this is not the general advice *"always use arrays/lists of structs"*!

Array of structs vs array of classes

- a nice example of *leaking abstraction* that we can use if caring about data locality
  - may be applied to many collections - most are baked by arrays
- this is not the general advice *"always use arrays/lists of structs"*!
- in case of structs:
  - if array is sparse and element big - we take much more space!
  - efficient access to elements is not trivial - remember about pass-by-value semantics!

Array of structs vs array of classes

- a nice example of *leaking abstraction* that we can use if caring about data locality
  - may be applied to many collections - most are baked by arrays
- this is not the general advice *"always use arrays/lists of structs"*!
- in case of structs:
  - if array is sparse and element big - we take much more space!
  - efficient access to elements is not trivial - remember about pass-by-value semantics!
- we will learn how to measure the difference

DEMO