Class and Object

Storing Data in Procedural Way

Record of a student is highly correlated but it is stored in different disjoint arrays and linked with the index number.

```
int array_size = 5;
string [] sname = new string[array_size];
float [] matricMarks = new float[array_size];
float [] fscMarks = new float[array_size];
float [] ecatMarks = new float[array_size];
float[] aggregate = new float[array_size];
```

Any Solution ?

Can we Improve the solution in any way?

```
int array_size = 5;
string [] sname = new string[array_size];
float [] matricMarks = new float[array_size];
float [] fscMarks = new float[array_size];
float [] ecatMarks = new float[array_size];
float[] aggregate = new float[array_size];
```

Any Solution ?

We have seen a solution in previous course that packages the data into single block using the help of struct keyword.

```
struct student
{
    string sname;
    float matricMarks;
    float fscMarks;
    float ecatMarks;
    float aggregate;
};
```

Class

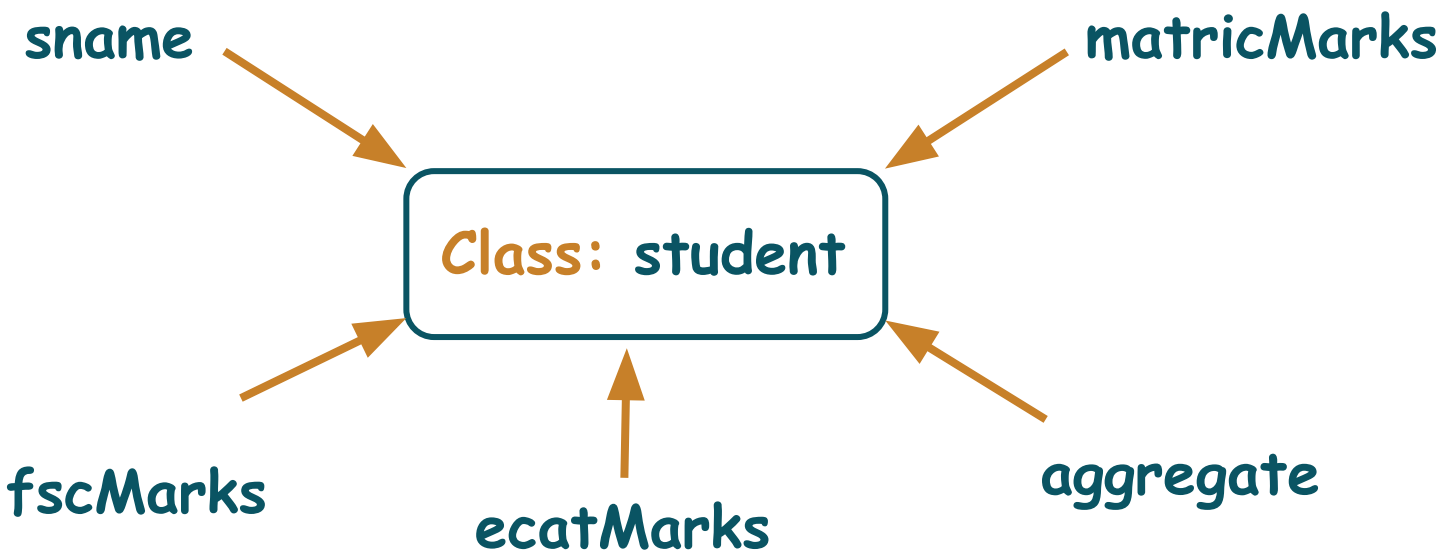
C# also has have somehow equivalent construct that is called class.

Class VS Structure

The class has more power than structure. We shall see the comparison between the two, later in the course.

Class

Class is a way to package related data in a single unit.



Class

Related data can be packaged into a class in the following way.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Class

Related data can be packaged into a class in the following way.

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Class

Related data can be packaged into a class in the following way.

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Class

Related data can be packaged into a class in the following way.

User defined DataType name

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Class

Related data can be packaged into a class in the following way.

User defined DataType name

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Data Members of the Class

Class

Related data can be packaged into a class in the following way.

User defined DataType name

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Data Members of the Class

No semicolon at the end

Class

Related data can be packaged into a class in the following way.

User defined DataType name

Keyword

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

We will use public keyword (access modifier) with every member for now

Data Members of the Class

No semicolon at the end

Variables of Class

The Class acts as a user-defined data type, therefore, we can create its variables.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

Variables of Class

A Variable of the class can be created as:

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

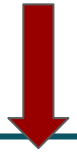
```
student s1 = new student();
```

Variables of Class

A Variable of the class can be created as:

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

name of the class



```
student s1 = new student();
```

Variables of Class

A Variable of the class can be created as:

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

name of the class

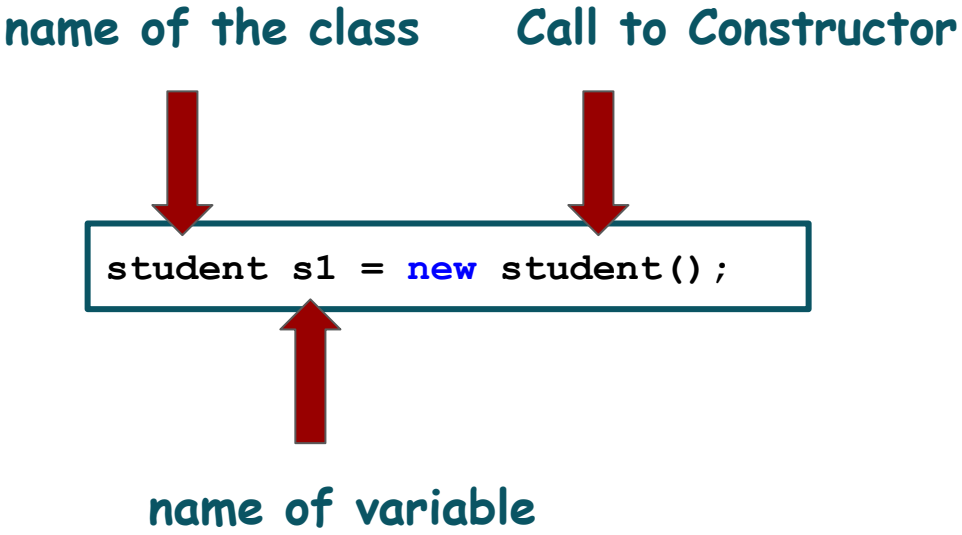
```
student s1 = new student();
```

name of variable

Variables of Class

A Variable of the class can be created as:

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```



Variable: Instance or Object

This variable of the class is also called instance or object of the class

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
```

Assigning values to object

Now, this object s1 represents all required information of a student

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
```

Assigning values to object

If you need to update any information you can use dot notation to access the members of the class.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
```

Assigning values to object

If you need to update any information you can use dot notation to access the members of the class.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
s1.matricMarks = 1000F;
s1.fscMarks = 1050F;
```

Object Attributes

Data Member of student class is also called Attribute. For example, here sname is an attribute of the class.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
s1.matricMarks = 1000F;
s1.fscMarks = 1050F;
```

Object Attributes

Data Member of student class is also called Attribute.
For example, here sname is an attribute of the class.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
s1.matricMarks = 1000F;
s1.fscMarks = 1050F;
```

Object Attributes

Similarly matricMarks, fscMarks, ecatMarks, aggregate are also the attributes of the class.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
s1.matricMarks = 1000F;
s1.fscMarks = 1050F;
```

Object Attributes

We can use the objects and their attributes like any other variables.

Object  Variable

Object Attributes

Means we can apply arithmetic operations, use them in boolean expression, and print on the console.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
s1.sname = "ABC";
s1.matricMarks = 1000F;
s1.fscMarks = 1050F;
Console.Write(s1.fscMarks/100);
```

Class and Object

When we initialize an object, it creates all variables of the class into the memory and we can reference these variables with the name of instance.

```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

```
student s1 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Class and Object

If you initialize third object, it will create another memory area that is exact replica of the class.

```
student s1 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s2 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s3 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Class and Object

s1.sname = "John"
only updates the memory of the s1 object and does not affect on the s2 and s3 objects.

```
student s1 = new student();
```

sname	"John"
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s2 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s3 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Multiple Objects

Similarly, we can generate multiple objects and each object will represent information of a different student.

```
student s1 = new student();
```

sname	"John"
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s2 = new student();
```

sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

```
student s3 = new student();
```

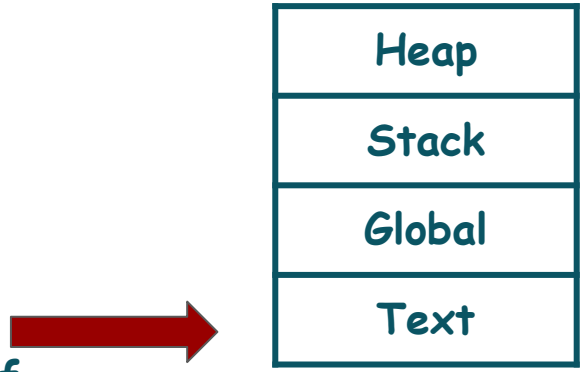
sname	""
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Review: Memory Layout of Programs

We know that memory is allocated to every program in 4 sections

Review: Memory Layout of Programs

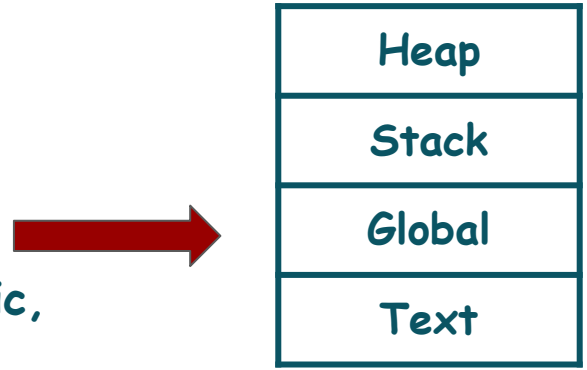
We know that memory is allocated to every program in 4 sections



Stores machine code of the compiled program.

Review: Memory Layout of Programs

We know that memory is allocated to every program in 4 sections

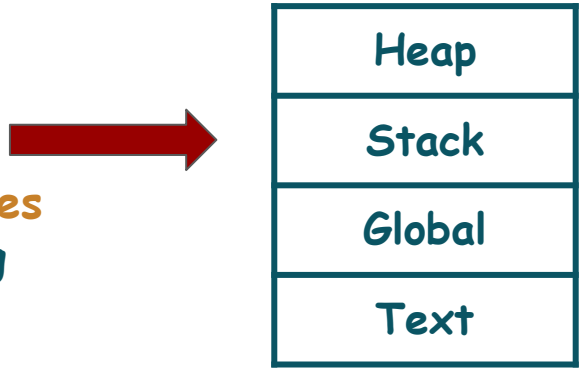


Stores all global, static, and constant variables

Review: Memory Layout of Programs

We know that memory is allocated to every program in 4 sections

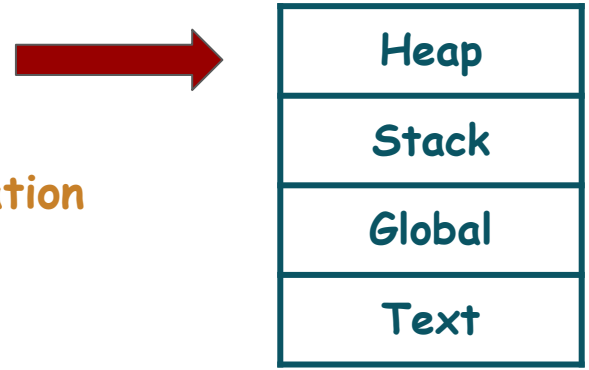
Stores all local variables and is used for passing arguments to the functions



Review: Memory Layout of Programs

We know that memory is allocated to every program in 4 sections

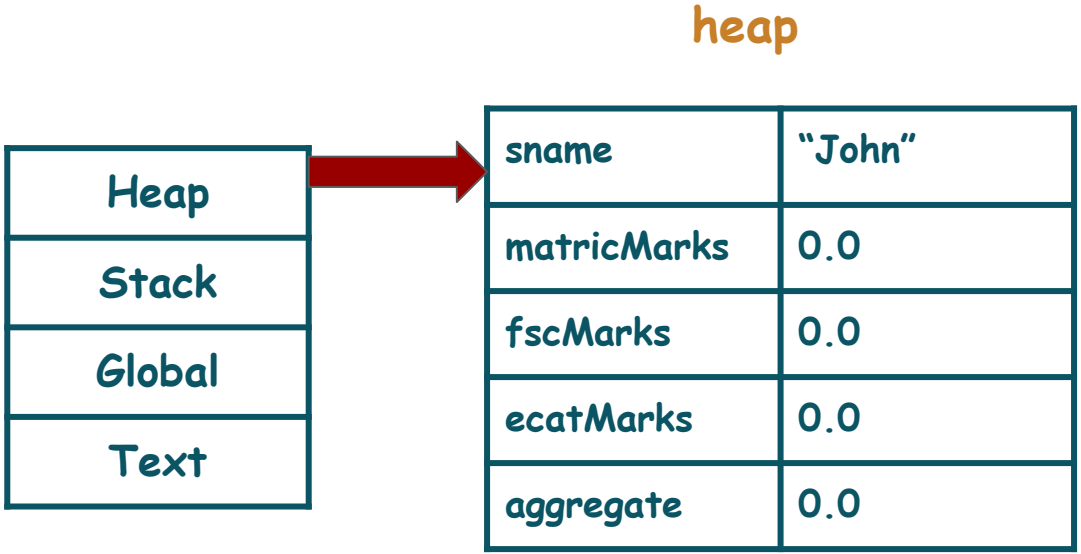
Heap section where dynamic memory allocation usually takes place



Review: Memory Layout of Programs

New keyword allocates memory on the heap.

```
student s1 = new student();
```



Review: Memory Layout of Programs

Reference to that memory address is created on the stack.

```
student s1 = new student();
```

s1	Address of heap

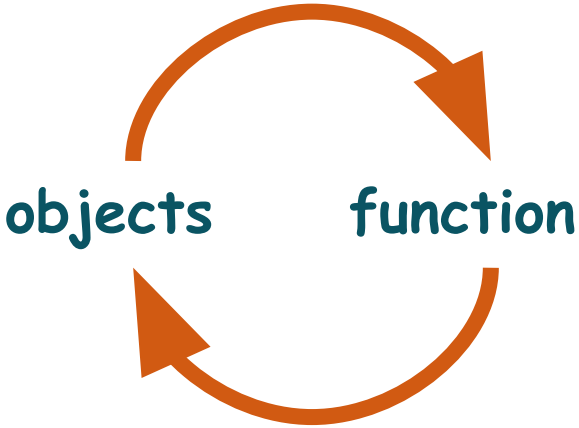
Heap
Stack
Global
Text

heap

sname	"John"
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Object in Function

We can also pass objects to a function and return objects from the functions.



Passing Object to Function by Reference

Since objects contains the Reference to the memory address allocated on the heap, therefore the objects are passed by reference to the function.

Passing Object to Function by Reference

Currently, we have made an object of student (s1) and initialized its name as "John".

```
static void Main(string[] args)
{
      student s1 = new student();
      s1.sname = "John";

}
```

sname	"John"
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Passing Object to Function by Reference

If we pass s1 to the function nameChanger then the address is only passed and the changes made in the nameChanger function are reflected in the main function as well.

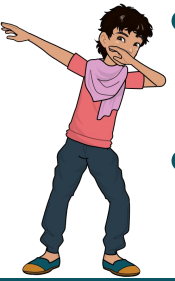
```
static void Main(string[] args)
{
      student s1 = new student();
      s1.sname = "John";
      Console.WriteLine(s1.sname);
      nameChanger(s1);
      Console.WriteLine(s1.sname);
      Console.Read();
}
```

```
static void nameChanger(student s)
{
      s.sname = "Jack";
}
```

sname	"Jack"
matricMarks	0.0
fscMarks	0.0
ecatMarks	0.0
aggregate	0.0

Conclusion

- We can represent related data as a single unit that is called class.
- Class is a custom (user defined) data type that consists of attributes (sub component of any real time object).
- Object is a variable of class type that holds all information present in the class.
- We can use an object into any arithmetic and boolean expressions.
- We can create multiple objects of class to hold different information.
- Objects reference is created on Stack and the actual memory is allocated on Heap.



Learning Objective

Write a Program that package the related data as single unit (Class) and create variable (Object) to use the data.

Self Assessment: Class and Objects

1. Write a program that shows three menu options
   - Add Student.
   - Show Students.
   - Top Students.

Add Student allows user to add a student's information that includes RollNo, Name, GPA, isHostelide, Department.
Show Student displays all the added students on the screen.
Top Student lists the information of the top 3 students.