Object Oriented Programing

Lecture 6 Inheritance

Arsalan Rahman Mirza

Computer Science Department

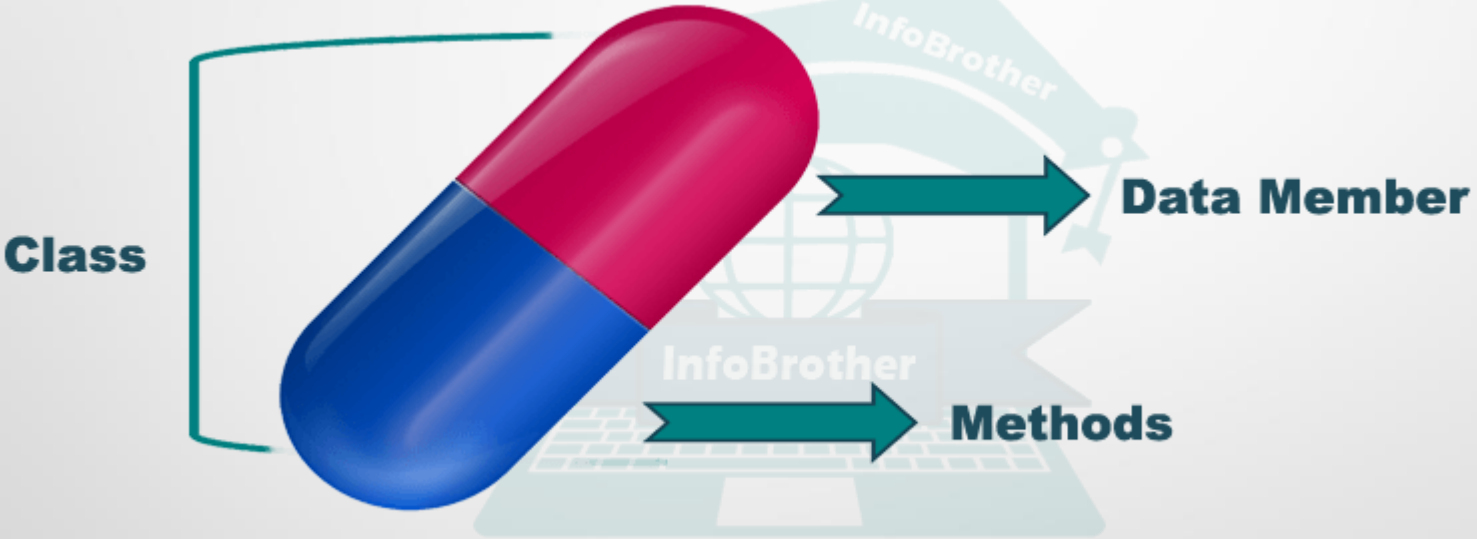
Faculty of Science

2022-2023

Topics

- Encapsulation
- Inheritance

Encapsulation



Encapsulation

Encapsulation, in the context of C#, refers to an object's ability to hide data and behavior that are not necessary to its user.

Encapsulation enables a group of properties, methods and other members to be considered a single unit or object.

The following are the benefits of encapsulation:

- Protection of data from accidental corruption
- Specification of the accessibility of each of the members of a class to the code outside the class
- Flexibility and extensibility of the code and reduction in complexity

Encapsulation

Encapsulation in C# is implemented with different levels of access to object data that can be specified using the following access modifiers:

- Public: Access to all code in the program
- Private: Access to only members of the same class
- Protected: Access to members of same class and its derived classes

inheritance

inheritance

- Inheritance is an important concept in C#. Inheritance is a concept in which you define parent classes and child classes.
- The child classes inherit methods and properties of the parent class, but at the same time, they can also modify the behavior of the methods if required. The child class can also define methods of its own if required.

Police Station

```
class Criminal
{
    string name;
    int bYear;

    int jailYears;
    string crimetype;

}
```

```
class Police
{
    string name;
    int bYear;

    int salary;
    string department;
}
```

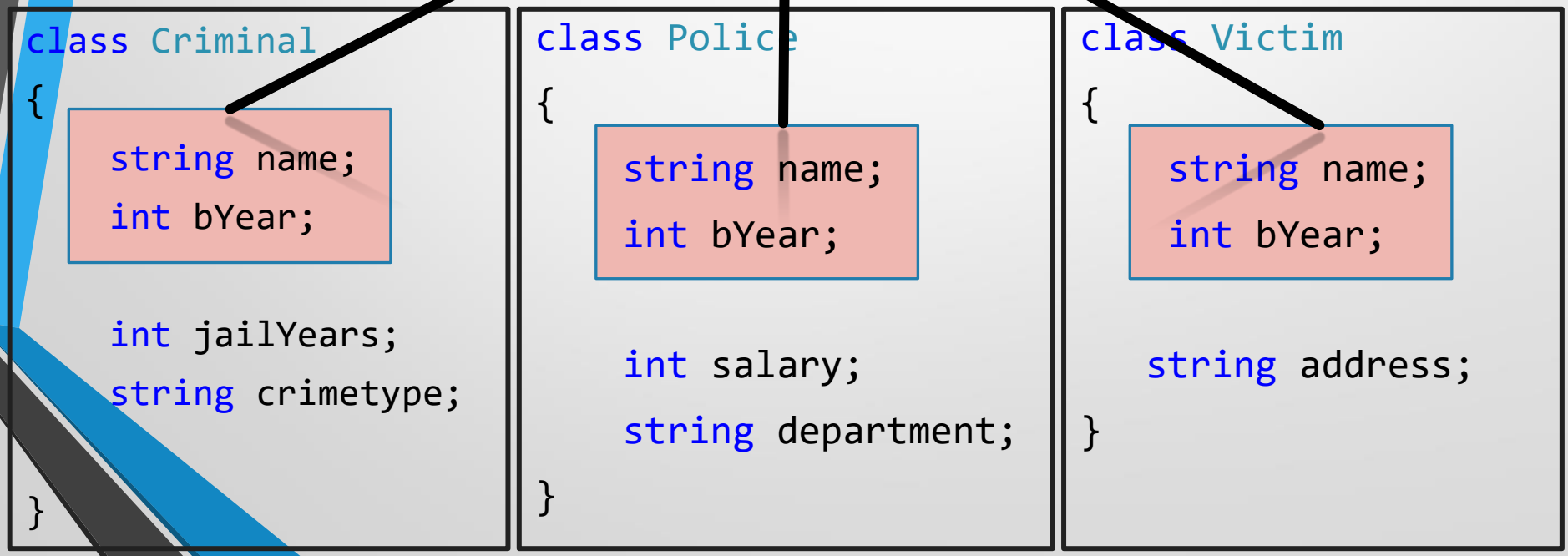
```
class Victim
{
    string name;
    int bYear;

    string address;
}
```

- What is the problem here?
- What have/has been repeated?
- Can we reduce the developing volume?

Police Station

Every person has name and Birth Year



Police Station

```
class Person
{
    string name;
    int bYear;
}
```

Polices, Criminals and Victims are Persons

Is it Done?
Not yet…

```
class Criminal
{
    string name;
    int bYear;

    int jailYears;
    string crimetype;

}
```

```
class Police
{
    string name;
    int bYear;

    int salary;
    string department;
}
```

```
class Victim
{
    string name;
    int bYear;

    string address;
}
```

Police Station

```
class Person
{
    string name;
    int bYear;
}
```

Problems

1. There is no connection

2. Person Data is Private

```
class Criminal
{
    string name;
    int bYear;

    int jailYears;
    string crimetype;

}
```

```
class Police
{
    string name;
    int bYear;

    int salary;
    string department;
}
```

```
class Victim
{
    string name;
    int bYear;

    string address;
}
```

Police Station

```
class Person
{
    string name;
    int bYear;
}
```

```
class Victim:Person
{
    string address;
}
```

```
class Criminal:Person
{
    int jailYears;
    string crimetype;
}
```

```
class Police:Person
{
    int salary;
    string department;
}
```

Police Station – Access Modifier

- There are 3 main types of Access Modifiers:
- **Private**: The type or member can be accessed only by code in the same class or struct.
- **Protected**: The type or member can be accessed only by code in the same class, or in a class that is **derived** from that class.
- **Public**: The type or member can be accessed by any other code.

Police Station

```
class Person
{
    protected string name;
    protected int bYear;
}
```

```
class Victim:Person
{
    string address;
}
```

```
class Criminal:Person
{
    int jailYears;
    string crimetype;
}
```

```
class Police:Person
{
    int salary;
    string department;
}
```

Inheritance

- In General
- **Class B is Inheritance of Class A**

```
class A
{

}

class B : A
{

}
```

Base Class for B

Derived Class of A

```
class Y : X
{
    private int d;
    protected int e;
    public int f;
}
```

```
class X
{
    private int a;
    protected int b;
    public int c;
}
```

```
class Z: X
{
    private int g;
    protected int h;
    public int i;
}
```

```
class W : Z
{
    private int j;
    protected int k;
    public int m;
}
```

X is base for **Y** and **Z**

Z is base for **W**

Y, Z are derived from **X**

W is derived from **Z**

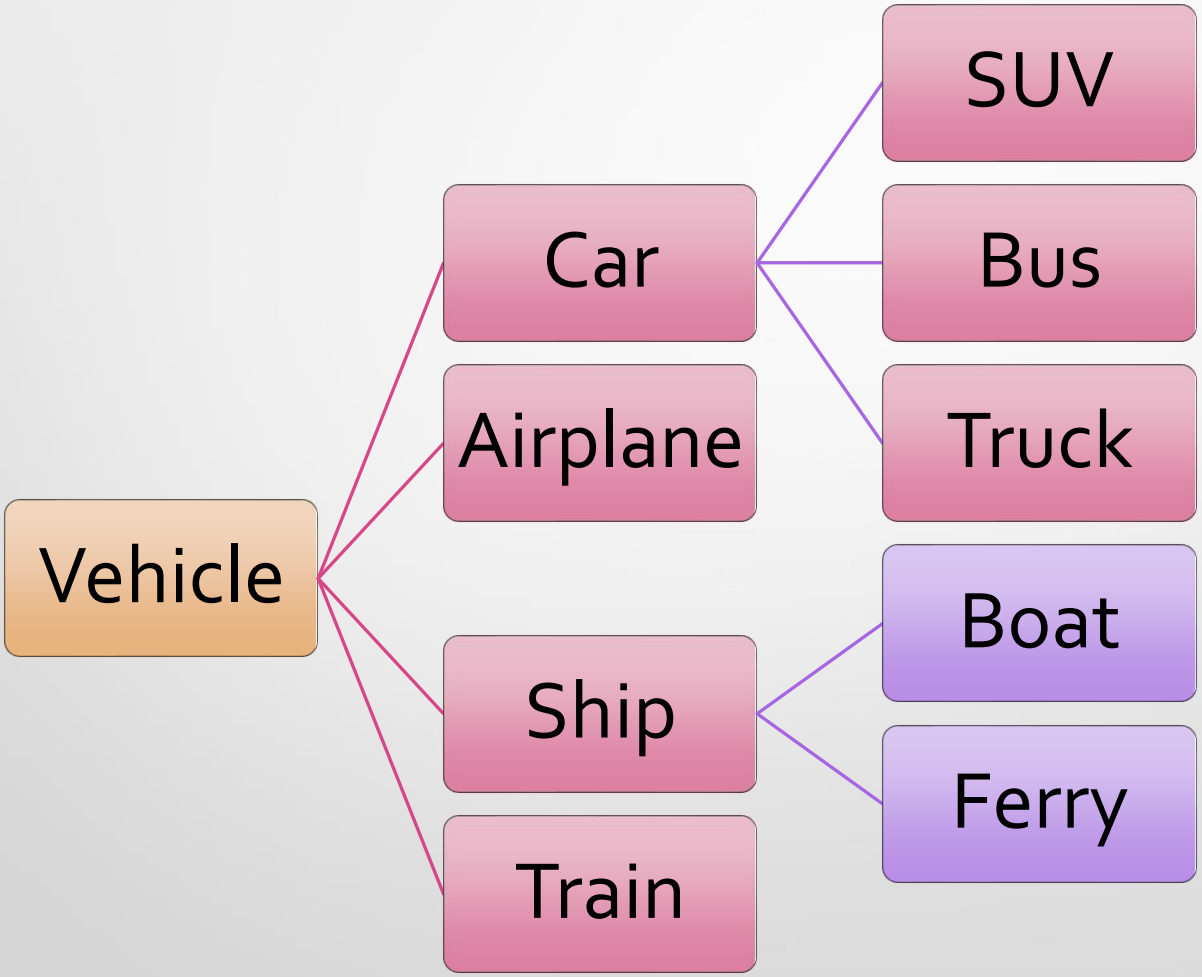
X can access **a, b, c, f, i & m**

Y can access **d, e, f, b, c, i & m**

Z can access **g, h, i, b, c, f & m**

W can access **j, k, m, h, i, f and c**

LAB activity



HW6: Create the following Classes