Extending the Functionality

اَللّٰهُمَّ اُرْزُقْنِی عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

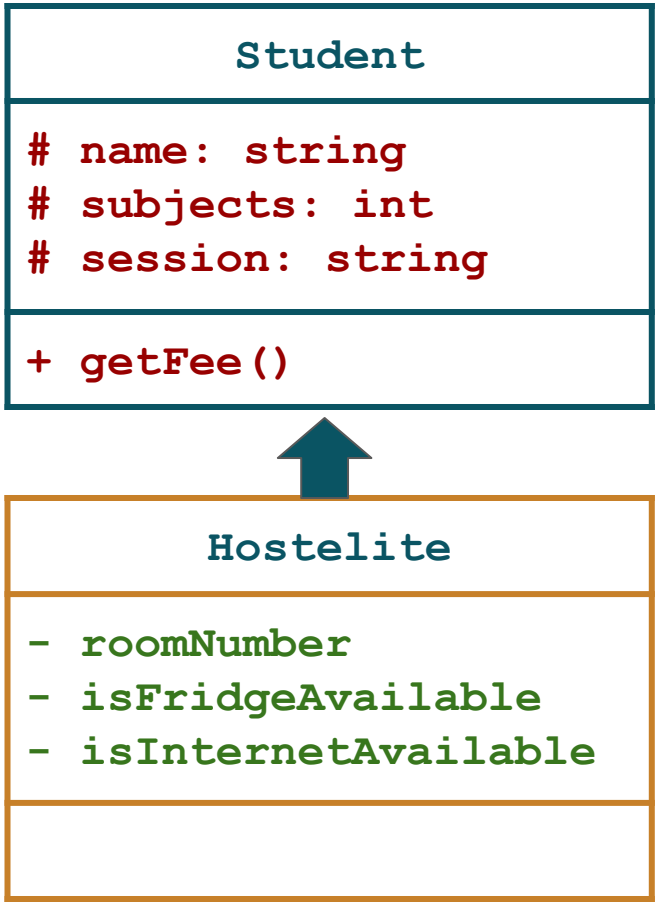
اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

Inheritance

When a class inherits another class, all the functionality (attributes and functions) become part of this class.

Problem Scenario

For example, in this case Hostelite class inherits Student. Now the object of Hostelite class have all Functionality that exists in the student class.



Problem Scenario

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Hostelite: Student
{
  private int roomNumber;
  private bool isFridgeAvailable;
  private bool isInternetAvalable;
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Problem Scenario: Modify the Functionality

Let's say, the child class (Hostelite) needs to modify/extend the capability of the parent class (Student).

Problem Scenario: Modify the Functionality

Let's say, the child class (Hostelite) needs to modify/extend the capability of the parent class (Student).

Problem Scenario: Modify the Functionality

Suppose, the fee for Hostelite is bit different than the general Student.
It has extra fee of hostel room, internet and Fridge that is required to be added in the Registered Subject fees.

Problem Scenario: Modify the Functionality

So the problem is the child class (Hostelite) needs to modify/extend the capability of the parent class (Student)

What to Do ?

Function Overriding

Object Oriented Programming offers us a way to modify/extend the functionality of the parent class through function overriding.

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Wheneve
r extend
the
functiona
lity in
the child
class
with the
same
name,
use new
keyword

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

It means we are changing what the Parent class does for the Child class.

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Anything Wrong with this?

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Repetition of Code.

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Here it is just one line why it could be the problem ?

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Function Overriding: One Line why bother ?

Whenever same business logic repeats itself across the code, it will get difficult to track all the places and update the change every where.

Modify the Functionality

Object Oriented Programming has better solution, when you need to extend the functionality of Base Class (Parent Class) you can call the parent function from child function.

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee = base.getFee();

        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;

    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }

    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

base is reserved word to call the parent method

```
class Hostelite: Student
{
    private int roomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvalable;
    public new int getFee()
    {
        //Fee 4000 per subject
        int fee = base.getFee();

        if (isFridgeAvailable)
        {
            fee = fee + 1000;
        }
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setSubjects(4);
    int fee = std.getFee();
    Console.WriteLine("Fee " + fee);
    Console.ReadKey();
}
```

```
class Student
{
    protected string name;
    protected string session;
    protected int subjects;
    public void setName(string name)
    {
        this.name = name;
    }
    public void setSession(string session)
    {
        this.session = session;
    }
    public void setSubjects(int subjects)
    {
        this.subjects = subjects;
    }
    public int getFee()
    {
        //Fee 4000 per subject
        int fee;
        fee = subjects * 4000;
        return fee;
    }
}
```

Function Overriding: Advantage

Some one may argue, we even we need overriding? We can declare another function with some different name into the child class.

Function Overriding: Advantage

Some one may argue, we even we need overriding? We can declare another function with some different name into the child class.

Any Answer?

Function Overriding: Advantage

It helps to create consistency in the classes.
Now developer is aware who ever the student whether hostelite, general student or day scholar, all will have same method (getFee()) that he/she can use to calculate the fee.

Conclusion

- While extending a class, the subclass inherits all of the public and protected attributes and behaviours from the parent class.
- Function overriding is a feature that allows us to have a same function in child class which is already present in the parent class.
- Base keyword is used, if we want to extend the functionality of a function in child class.

Learning Objective

Child Overrides Behaviour of its Parent Class

Self Assessment: Write Output

```
class Animal
{
    public void sound()
        {
            Console.WriteLine("This is parent class");
        }
}

class Dog: Animal
{
    public new void sound()
        {
            Console.WriteLine("Dog bark");
        }
}

class Cat: Animal
{
    public new void sound()
        {
            base.sound();
            Console.WriteLine("Cat meow");
        }
}
```

```
Dog d = new Dog();
Cat c = new Cat();
d.sound();
c.sound();
```