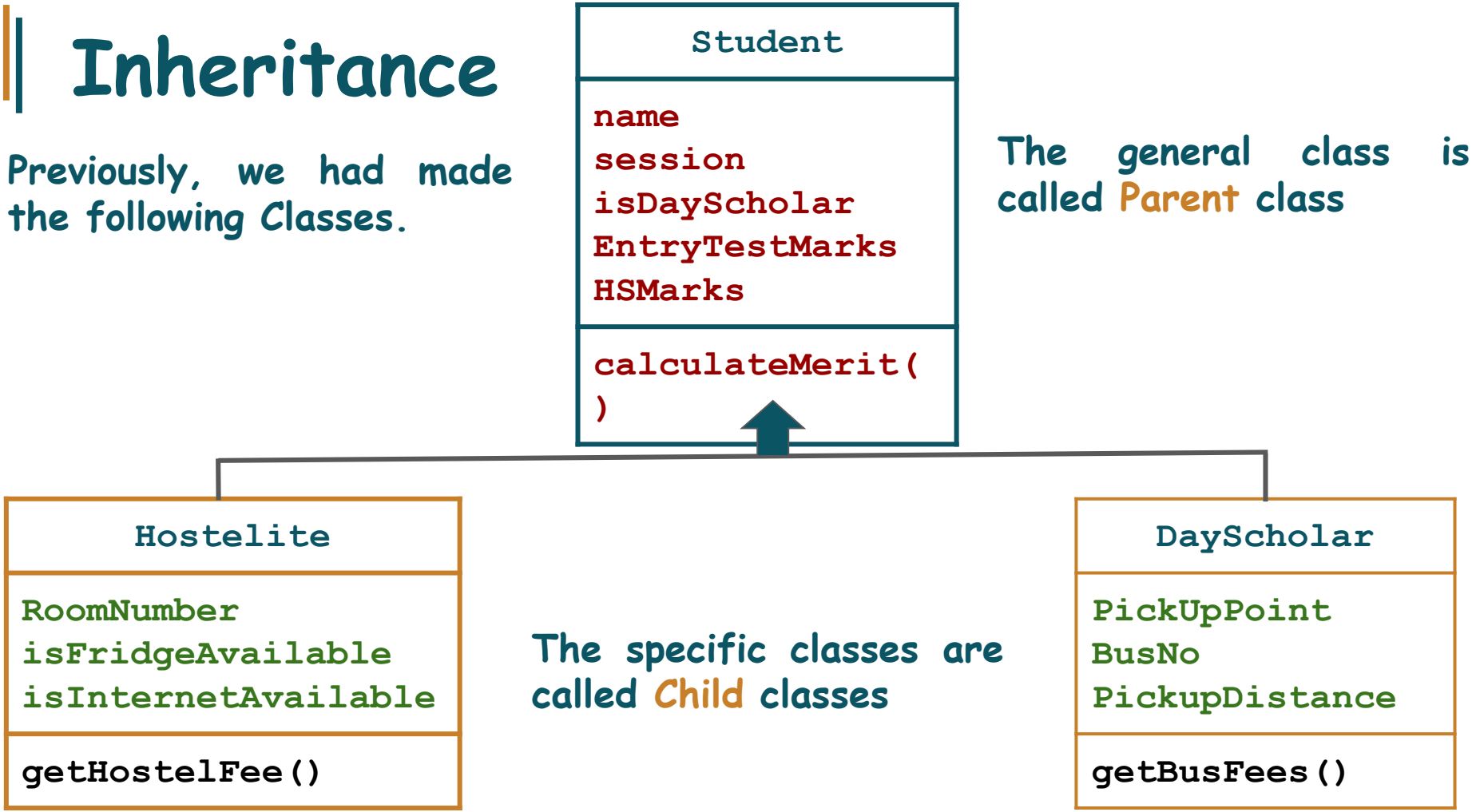
Relationship between the Constructors of Parent and Child

Inheritance

Previously, we had made the following Classes.

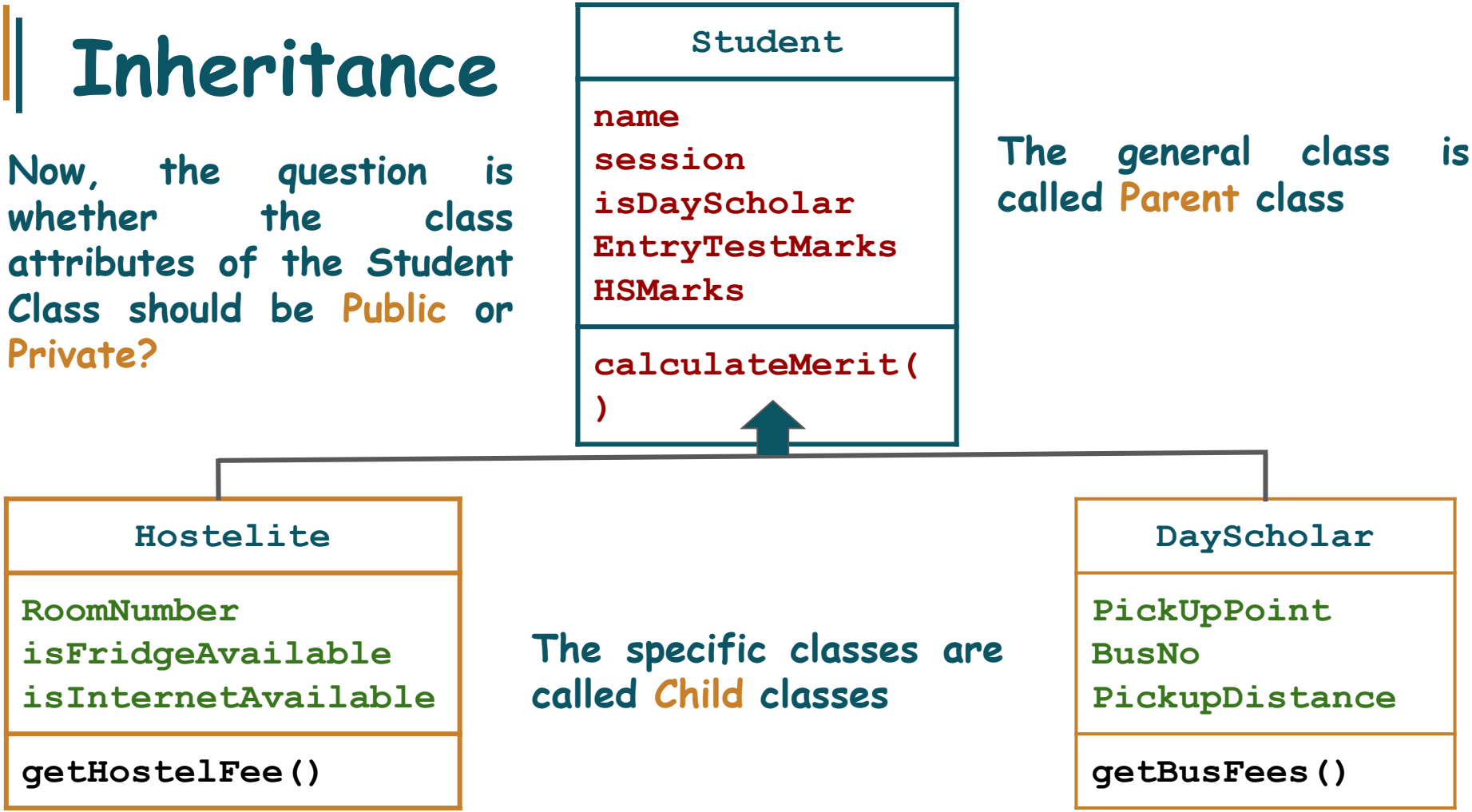


The general class is called Parent class

The specific classes are called Child classes

Inheritance

Now, the question is whether the class attributes of the Student Class should be Public or Private?



The general class is called Parent class

The specific classes are called Child classes

Inheritance

Let's see if we make them Public.

```
class Student
{
    public string name;
    public string session;
    public bool isDayScholar;
    public int EntryTestMarks;
    public int HSMarks;
}
```

Inheritance: Any Problem in This?

Let's see if we make them Public.

```
class Student
{
    public string name;
    public string session;
    public bool isDayScholar;
    public int EntryTestMarks;
    public int HSMarks;
}
```

Inheritance: Any Problem in This?

Let's see if we make them Public. They will be accessed in the Child class. But any class will access them as well by creating the object of Student Class.

```
class Student
{
    public string name;
    public string session;
    public bool isDayScholar;
    public int EntryTestMarks;
    public int HSMarks;
}
```

Inheritance:

Let's see if we make them Private.

```
class Student
{
    private string name;
    private string session;
    private bool isDayScholar;
    private int EntryTestMarks;
    private int HSMarks;
}
```

Inheritance: Any Problem in This?

Let's see if we make them Private.

```
class Student
{
    private string name;
    private string session;
    private bool isDayScholar;
    private int EntryTestMarks;
    private int HSMarks;
}
```

Inheritance: Any Problem in This?

Let's see if we make them Private. Outside classes will not be able to access them. But the Child class will not be able to access the attributes also.

```
class Student
{
    private string name;
    private string session;
    private bool isDayScholar;
    private int EntryTestMarks;
    private int HSMarks;
}
```

Solution

Object Oriented Programming also have another Access modifier.

1. Public
2. Private
3. Protected

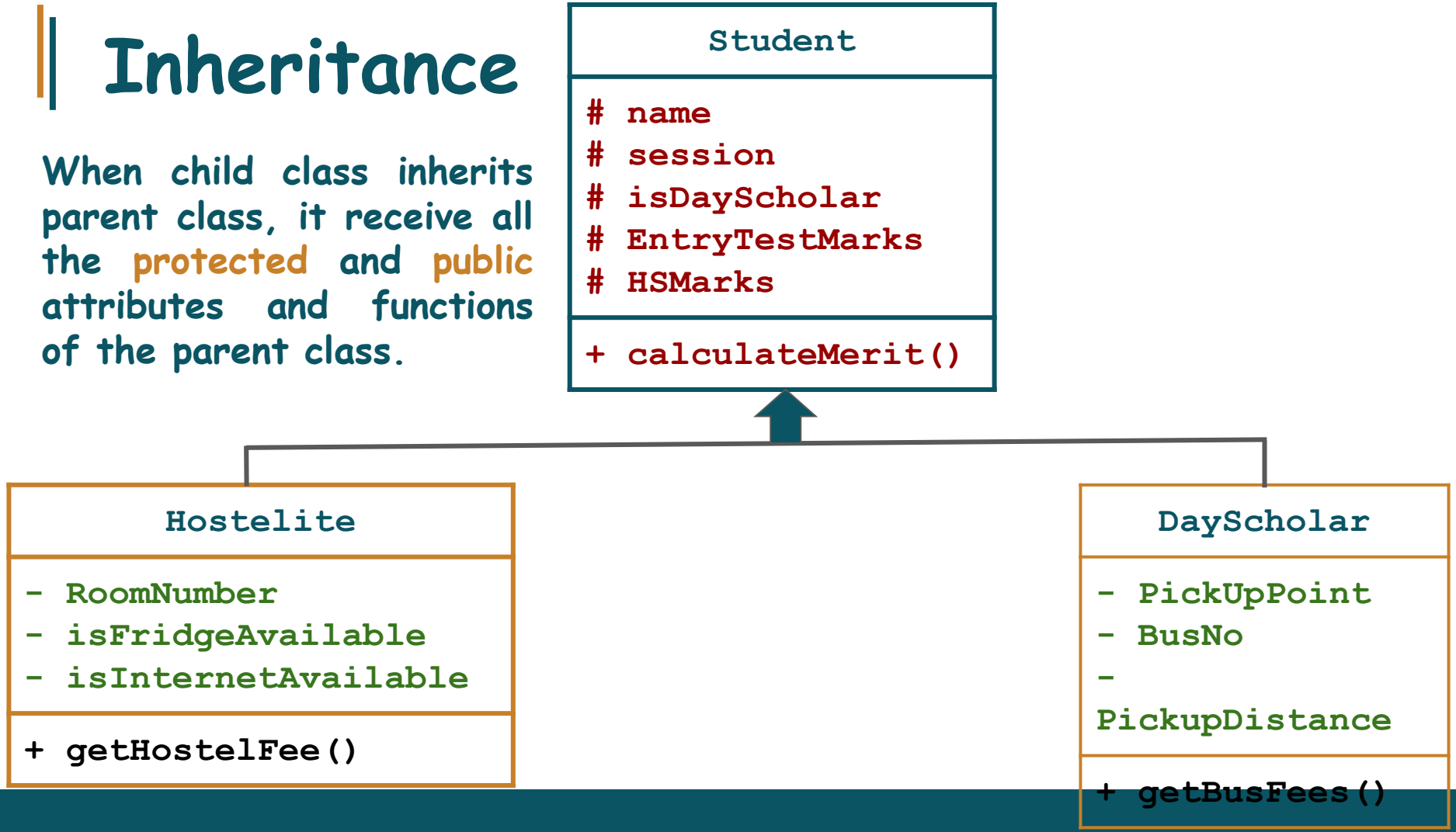
Encapsulation

Access modifiers are an integral part of object oriented programming. Access modifiers are used to implement Encapsulation of OOP. Access modifiers allow you to define who does or who doesn't have access to certain features.

Modifier	Description
public	There are no restrictions on accessing public members.
private	Access is limited to within the class definition. This is the default access modifier type if none is formally specified
protected	Access is limited to within the class definition and any class that inherits from the class

Inheritance

When child class inherits parent class, it receive all the protected and public attributes and functions of the parent class.



Working Example

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;

    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

Working Example

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;

    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

Working Example

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;

    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

Working Example

Object of Hostelite class have all the protected and public attributes and functions of Student class that it has inherited

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

Driver Program

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;

    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setName("Ahmad");
    std.setRoomNumber(12);
    Console.WriteLine(std.getName() + " is
allocated Room " + std.getRoomNumber());
    Console.ReadKey();
}
```

Driver Program

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;

    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    std.setName("Ahmad");
    std.setRoomNumber(12);
    Console.WriteLine(std.getName() + " is
allocated Room " + std.getRoomNumber());
    Console.ReadKey();
}
```

```
Ahmad is allocated Room 12
```

Child Inherits Parent's Legacy

It means Child Class gets the access of all the public and protected attributes and functions of its Parent Class.

Child Class uses all the public and protected attributes and functions of its Parent Class as its own.

Child Inherits Parent's Legacy

Now, let's see what is the relationship of Parent and Child Constructors because the constructors are automatically called.

Activity

What will be the output after the execution of the main?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

What will be the output after the execution of the main?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

It will print "Parent Constructor" on the screen

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
Parent Constructor
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

It will print "Parent Constructor" on the screen.
But Why?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
Parent Constructor
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Child Inherits Parent's Legacy

Whenever we create the object of child class it automatically calls the default constructor of parent class.

Activity

Let's add the Default Constructor inside the child class as well.

Activity

Now, What will be the output after the execution of the main?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;
    public Hostelite()
    {
        Console.WriteLine("Child Constructor");
    }
    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

Now, What will be the output after the execution of the main?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;
    public Hostelite()
    {
        Console.WriteLine("Child Constructor");
    }
    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

It will call the Parent Constructor and then the Child Constructor.

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student()
    {
        Console.WriteLine("Parent Constructor");
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
Parent Constructor
Child Constructor
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;
    public Hostelite()
    {
        Console.WriteLine("Child Constructor");
    }
    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Child Inherits Parent's Legacy

Whenever we create the object of child class it automatically first calls the default constructor of parent class and then its own constructor.

Activity

Now Let's add a Parameterized Constructor inside the Parent class.

Activity

What will be the Output?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name)
    {
        this.name = name;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

What will be the Output?

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name)
    {
        this.name = name;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Activity

This will give us Compile time error as we are not passing the name.

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name)
    {
        this.name = name;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()
    {
        double merit = 0.0;
        // Code to calculate merit
        return merit;
    }
}
```

There is no argument given that corresponds to the required formal parameter 'name' of 'Student.Student(string)'

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()
    {
        int fee = 0;
        // Code to calculate fee
        return fee;
    }
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite();
    Console.ReadKey();
}
```

Solution?

So, how can we explicitly pass the parameters in the parameterized constructor of the parent class through child class?

Activity

We use the base keyword to call the parameterized constructor of parent Class

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name)
    {
        this.name = name;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()...
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public Hostelite(string name): base (name)
    {
    }

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()...
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite("Ahmad");
    Console.WriteLine(std.getName());
    Console.ReadKey();
}
```

Activity

We use the base keyword to call the parameterized constructor of parent Class

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name)
    {
        this.name = name;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()...
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public Hostelite(string name): base (name)
    {
    }

    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()...
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite("Ahmad");
    Console.WriteLine(std.getName());
    Console.ReadKey();
}
```

More than one Parameter

Similarly, we can explicitly pass more than one parameters in the parameterized constructor of the parent class through child class using the base keyword.

Activity

We use the base keyword to call the parameterized constructor of parent Class

```
class Student
{
    protected string name;
    protected string session;
    protected bool isDayScholar;
    protected int EntryTestMarks;
    protected int HSMarks;
    public Student(string name, string session)
    {
        this.name = name;
        this.session = session;
    }
    public void setName(string name)...
    public void setSession(string session)...
    public void setIsDayScholar(bool isDayScholar)...
    public void setEntryTestMarks(int EntryTestMarks)...
    public void setHSMarks(int HSMarks)...
    public string getName()...
    public double calculateMerit()...
}
```

```
class Hostelite : Student
{
    private int RoomNumber;
    private bool isFridgeAvailable;
    private bool isInternetAvailable;

    public Hostelite(string name, string
session): base (name, session)
    {
    }
    public void setRoomNumber(int RoomNumber)...
    public void setIsFridgeAvailable(bool
isFridgeAvailable)...
    public void setIsInternetAvailable(bool
isInternetAvailable)...
    public int getRoomNumber()...
    public int getHostelFee()...
}
```

```
static void Main(string[] args)
{
    Hostelite std = new Hostelite("Ahmad",
"2021");
    Console.ReadKey();
}
```

Self Assessment: Inheritance

Implement the Following Classes

Bicycle

```
# cadence: int
# gear: int
# speed: int;
```

```
+ Bicycle(int cadence, int speed, int gear)
+ void setCadence(int cardence)
+ void setGear(int gear)
+ void applyBrake(int decrement)
+ void speedUp(int increment)
```

MountainBike

```
- seatHeight: int
```

```
+ MountainBike(int seatHeight, int cadence, int speed, int gear)
+ void setSeatHeight(int seatHeight)
```