Polymorphism

Problem Scenario

Sometimes due to different conditions, we want our functions act differently.
This is called polymorphism.
Poly = many
Morphism = shapes (states)

Polymorphism: Types

We have discussed two types of Polymorphism.

1. Dynamic Polymorphism
2. Static Polymorphism

Dynamic Polymorphism

Problem Scenario: Extend the Code

we want to develop a system such it allows to create three types of shapes.

Rectangle: to represent a rectangle, width and height are required. Formula to find the area is wxh.

Square: to represent a square, a single side (s) is sufficient. Formula to find the area of square is $s \times s = s^2$

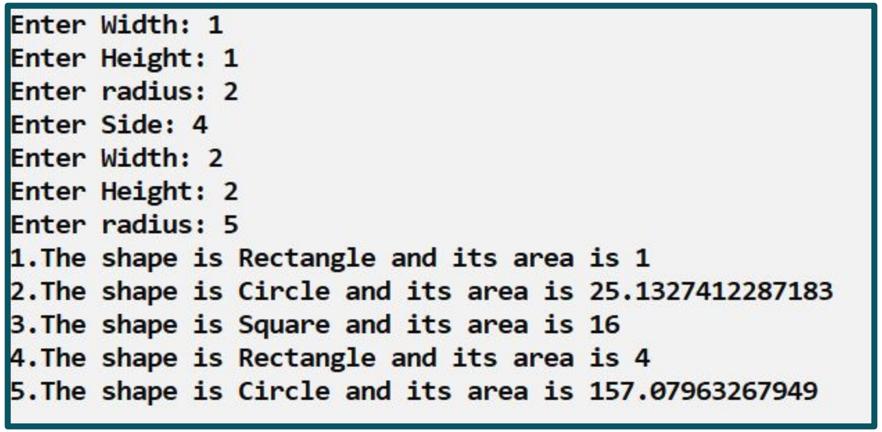
Circle: to represent a radius (r) is enough. Formula to find the area of Circle is $2\pi r^2$

Problem Scenario

1. System is required to save information of these three shapes.
2. In main method any type and any number of shape objects could be created.
3. All these objects should be created through the respective UI Class and should be added in to a SINGLE COMMON LIST. We do NOT want to create separate DLs for each type of shape.
4. When a program runs it shows all objects, object type (type of shape) and the area of the shape.

Problem Scenario: DriverProgram

```
class Program
{
    static void Main(string[] args)
    {
        Rectangle r = RectangleUI.createShape();
        ShapeDL.addIntoList(r);
        Circle c = CircleUI.createShape();
        ShapeDL.addIntoList(c);
        Square s = SquareUI.createShape();
        ShapeDL.addIntoList(s);
        Rectangle r1 = RectangleUI.createShape();
        ShapeDL.addIntoList(r1);
        Circle c1 = CircleUI.createShape();
        ShapeDL.addIntoList(c1);

        ShapeUI.showAreas(ShapeDL.getList());
        Console.ReadKey();
    }
}
```



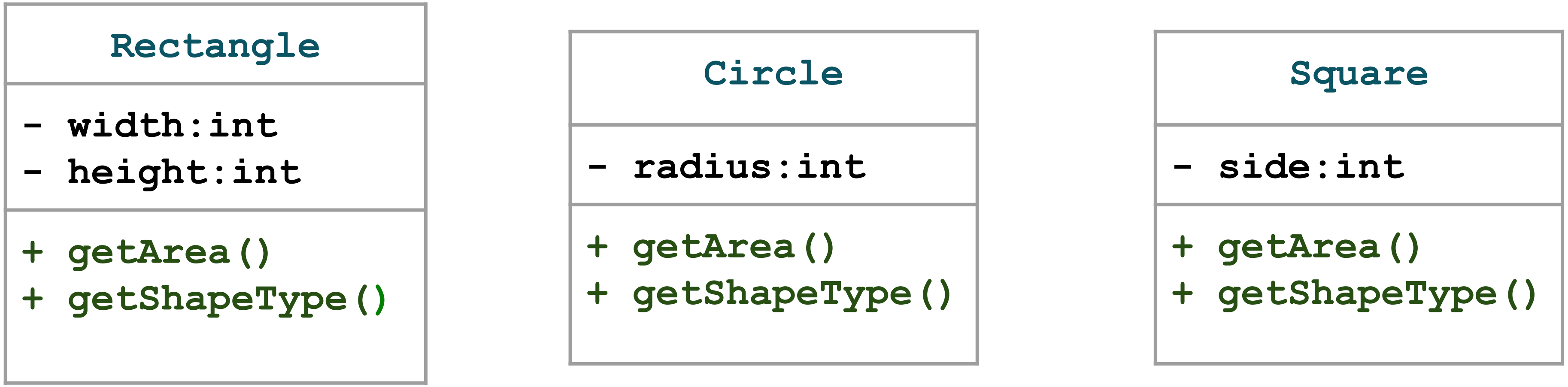
```
Enter Width: 1
Enter Height: 1
Enter radius: 2
Enter Side: 4
Enter Width: 2
Enter Height: 2
Enter radius: 5
1.The shape is Rectangle and its area is 1
2.The shape is Circle and its area is 25.1327412287183
3.The shape is Square and its area is 16
4.The shape is Rectangle and its area is 4
5.The shape is Circle and its area is 157.07963267949
```

Problem Scenario: Hint 01

One thing is obvious, that we need three separate classes each representing the corresponding shape and all these classes will have following two methods.

- getArea()
- getShapeType()

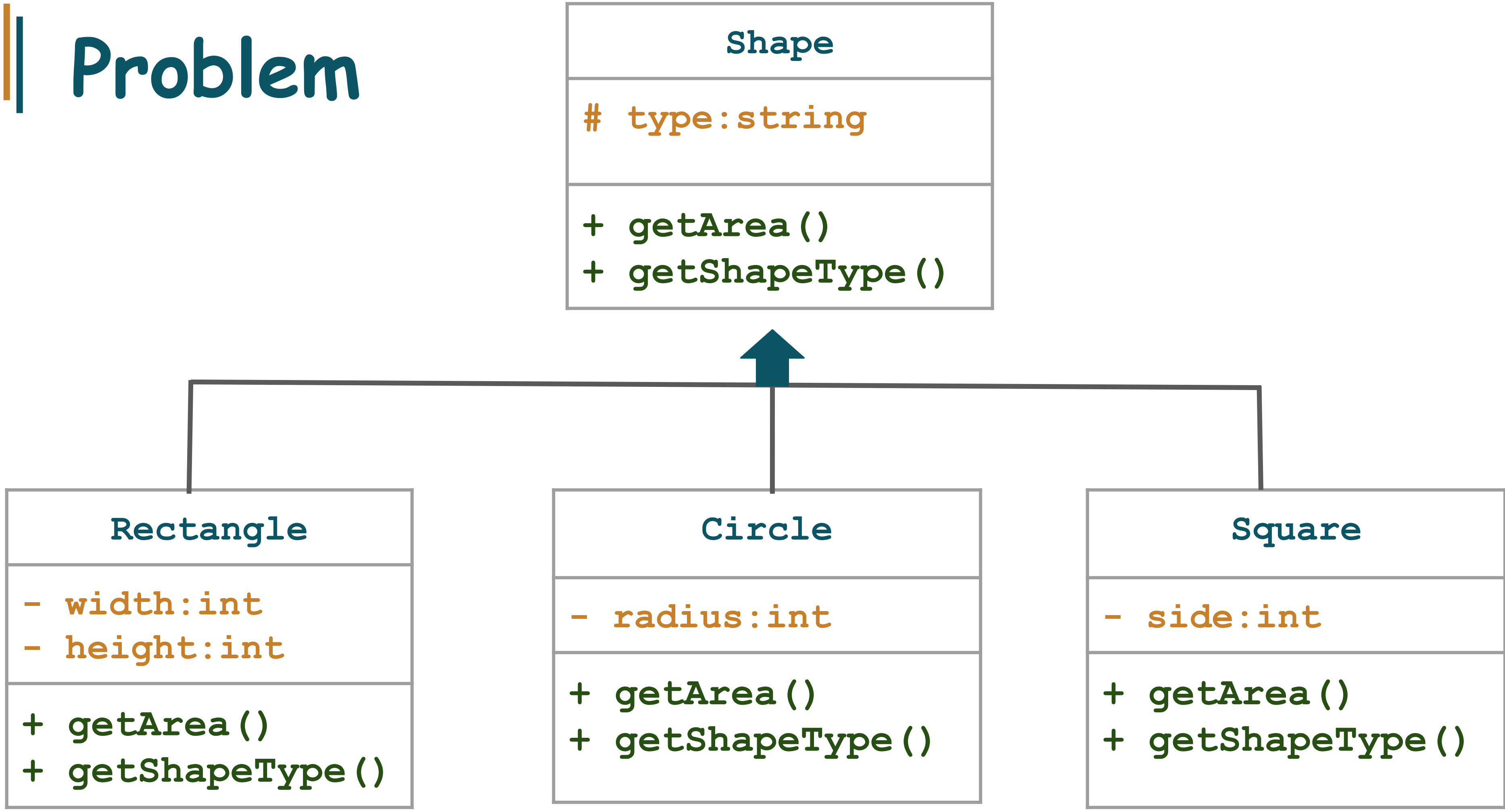
Problem Scenario: Hint 01



Function Overriding

But for single list that contains different type of shapes we need to create a general (common) class and all classes should extend the parent class

Problem


Shape
# type:string
+ getArea()
+ getShapeType()
Rectangle
- width:int
- height:int
+ getArea()
+ getShapeType()
Circle
- radius:int
+ getArea()
+ getShapeType()
Square
- side:int
+ getArea()
+ getShapeType()

List of Shape Object

Now, we can easily create a single list of shape class that can contain all type of shapes but there is one problem with it.

Can you highlight the problem ?

The Problem

If we call getArea() and getShapeType() it should be called according to the respective child shape not the functions of the parent class.

Dynamic Behaviour

Can Parent Class could have dynamic behaviour at run time ?

Dynamic Behaviour

Can Parent Class could have dynamic behaviour at run time ?

Yes, Object Oriented Programming allows Parent Class to use the functionality of its Child Class.
This functionality will be assigned at runtime.

Problem Scenario: Extend the Code

To implement it in the code, we can use our previous knowledge of inheritance and Function Overriding.

Problem Scenario: Extend the Code

We can create following functions in parent class with virtual keyword and override these in corresponding child classes

1. getArea() and
2. getShapeType().

Problem Scenario: Solution

We can create a parent class Shape with two methods.

```
class Shape
{
   public virtual double getArea()
   {
         return 0;
   }

   public virtual string getShapeType()
   {
         return "undefined.";
   }
}
```

Problem Scenario: CircleBL

```
class Circle:Shape
{
         int radius;
         public Circle (int radius)
          {
              this.radius=radius;
         }
         public override double getArea()
          {
              double area;
              area=2*Math.Pow(radius,2)*Math.PI;
              return area ;
         }
         public override string getShapeType()
          {
              return "Circle";
         }
}
```

Problem Scenario: RectangleBL & SquareBL

```
class Rectangle:Shape
{
  public int width;
  public int height;
  public Rectangle(int width,int height)
  {
     this.width=width;
     this.height=height;
  }
  public override double  getArea()
  {
      return width * height;
  }
  public override string getShapeType()
  {
      return "Rectangle";
  }
}
```

```
class Square:Shape
{
   int side;
   public Square (int side)
   {
    this.side=side;
   }
   public override double getArea()
   {
     double area;
     area=Math.Pow(side,2);
     return area ;
   }
   public override string getShapeType()
   {
      return "Square";
   }
}
```

Problem Scenario: ShapeDL

```
class ShapeDL
{
    private static List<Shape> shapesList = new List<Shape>();

    public static void addIntoList(Shape r)
    {
        shapesList.Add(r);
    }

    public static List<Shape> getList()
    {
        return shapesList;
    }
}
```

Problem Scenario: UI

```
class CircleUI
{
  public static Circle createShape()
  {
      Circle c;
      Console.Write("Enter radius: ");
      int radius = int.Parse(Console.ReadLine());
      c = new Circle(radius);
      return c;
  }
}
```

```
class SquareUI
{
    public static Square createShape()
    {
        Square s;
        Console.Write("Enter Side: ");
        int side = int.Parse(Console.ReadLine());
        s = new Square(side);
        return s;
    }
}
```

```
class RectangleUI
{
    public static Rectangle createShape()
    {
        Rectangle r;
        Console.Write("Enter Width: ");
        int width = int.Parse(Console.ReadLine());
        Console.Write("Enter Height: ");
        int height = int.Parse(Console.ReadLine());
        r = new Rectangle(width, height);
        return r;
    }
}
```

Problem Scenario: ShapeUI

```
class ShapeUI
{
    public static void showAreas(List<Shape> shapesList)
    {
        string message;
        for (int idx = 0; idx < shapesList.Count; idx++)
        {
            message = "The shape is {0} and its area is {1}";
            message = (idx + 1) + "." + message;
            Shape s = shapesList[idx];
            Console.WriteLine(message, s.getShapeType(), s.getArea());
        }
    }
}
```

Problem Scenario: ShapeUI

```
class ShapeUI
{
    public static void showAreas(List<Shape> shapesList)
    {
        string message;
        for (int idx = 0; idx < shapesList.Count; idx++)
        {
            message = "The shape is {0} and its area is {1}";
            message = (idx + 1) + "." + message;
            Shape s = shapesList[idx];
            Console.WriteLine(message, s.getShapeType(), s.getArea());
        }
    }
}
```

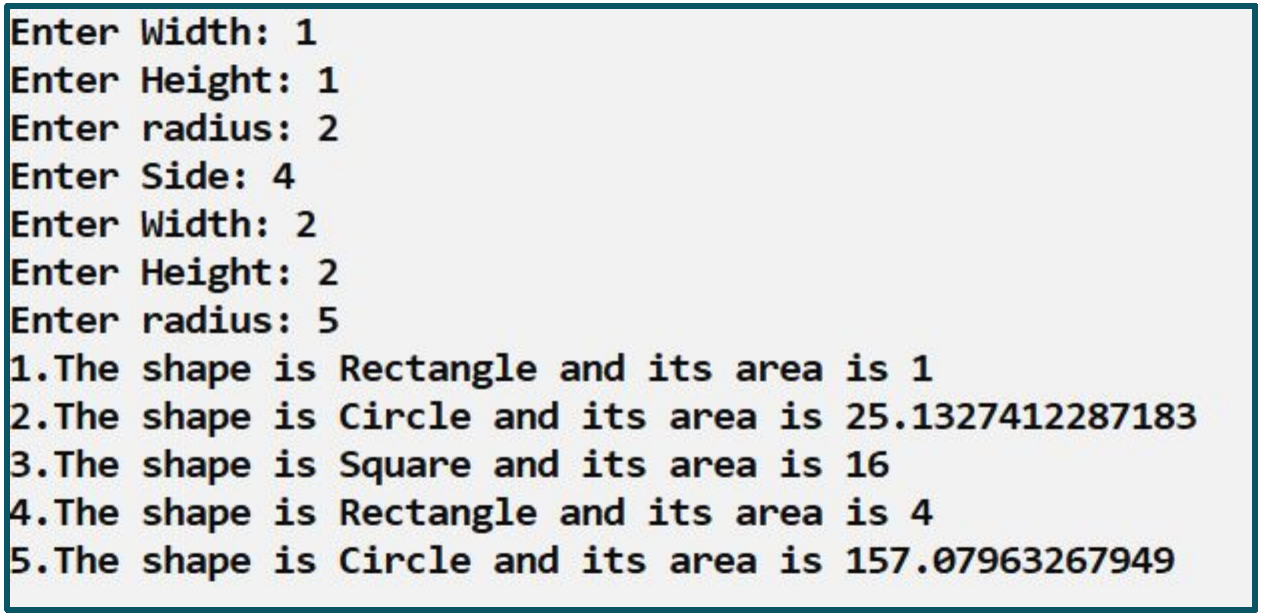
Problem Scenario: DriverProgram

```
class Program
{
    static void Main(string[] args)
    {
        Rectangle r = RectangleUI.createShape();
        ShapeDL.addIntoList(r);
        Circle c = CircleUI.createShape();
        ShapeDL.addIntoList(c);
        Square s = SquareUI.createShape();
        ShapeDL.addIntoList(s);
        Rectangle r1 = RectangleUI.createShape();
        ShapeDL.addIntoList(r1);
        Circle c1 = CircleUI.createShape();
        ShapeDL.addIntoList(c1);

        ShapeUI.showAreas(ShapeDL.getList());
        Console.ReadKey();
    }
}
```

Problem Scenario: DriverProgram

```
class Program
{
    static void Main(string[] args)
    {
        Rectangle r = RectangleUI.createShape();
        ShapeDL.addIntoList(r);
        Circle c = CircleUI.createShape();
        ShapeDL.addIntoList(c);
        Square s = SquareUI.createShape();
        ShapeDL.addIntoList(s);
        Rectangle r1 = RectangleUI.createShape();
        ShapeDL.addIntoList(r1);
        Circle c1 = CircleUI.createShape();
        ShapeDL.addIntoList(c1);

        ShapeUI.showAreas(ShapeDL.getList());
        Console.ReadKey();
    }
}
```

Problem Scenario: Output



```
Enter Width: 1
Enter Height: 1
Enter radius: 2
Enter Side: 4
Enter Width: 2
Enter Height: 2
Enter radius: 5
1.The shape is Rectangle and its area is 1
2.The shape is Circle and its area is 25.1327412287183
3.The shape is Square and its area is 16
4.The shape is Rectangle and its area is 4
5.The shape is Circle and its area is 157.07963267949
```

Static Polymorphism

Problem Scenario

For example, during the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

Problem Scenario

For example, during the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

How to implement this requirement?

Problem Scenario: Solution

One possible solution is that we declare single function with two parameters

```
getFee(string semester, boolean foreigner)
```

Problem Scenario: Solution

One possible solution is that we declare single function with two parameters

getFee(string semester, boolean foreigner)

Any Problem in this?

Problem Scenario: Solution

One possible solution is that we declare single function with two parameters

```
getFee(string semester, boolean foreigner)
```

A more cohesive function needs to have one single responsibility. Also, lots of fee rules will make its logic more complicated.

Problem Scenario: Solution

One possible solution is that we declare single function with two parameters

getFee(string semester, boolean foreigner)

A more cohesive function needs to have one single responsibility. Also, lots of fee rules will make its logic more complicated.

Then any Better Solution?

Problem Scenario: Solution

Another possible solution is we declare three functions with separate names

```
getFee()
getFeeForSummer()
getFeeForSummerForeginer()
```

Problem Scenario: Solution

Another possible solution is we declare three functions with separate names

```
getFee()
getFeeForSummer()
getFeeForSummerForeginer()
```

Any Problem in this?

Problem Scenario: Solution

Another possible solution is we declare three functions with separate names

```
getFee()
getFeeForSummer()
getFeeForSummerForeginer()
```

We have same functionality of calculating the fees but with different names.

Problem Scenario: Solution

Another possible solution is we declare three functions with separate names

```
getFee()
getFeeForSummer()
getFeeForSummerForeginer()
```

Then what is the Best Solution?

Function Overloading

Programming languages allow us to declare the functions with same name but with different parameters. This is called Function Overloading.

Function Overloading: Example

```
public int add(int a, int b)
{
    return a + b;
}
```

Function Overloading: Example

```
public int add(int a, int b)
{
    return a + b;
}
public int add(string a, string b)
{
    return int.Parse(a) + int.Parse(b);
}
```

Function Overloading: Example

```
public int add(int a, int b)
{
    return a + b;
}
public int add(string a, string b)
{
    return int.Parse(a) + int.Parse(b);
}
public int add(int a, int b,int c)
{
        return a + b + c;
}
```

Function Overloading: Example

```
public int add(int a, int b)
{
    return a + b;
}
public int add(string a, string b)
{
    return int.Parse(a) + int.Parse(b);
}
public int add(int a, int b,int c)
{
        return a + b + c;
}
public float add(float a, float b, float c, float d)
{
        return a + b + c + d;
}
```

Function Overloading

Overloaded methods have same name but either of following or both should be true

1. Different Data types of parameters
2. Different number of Arguments

Function Overloading

Overloading is type of static form of polymorphism.

Function Overloading: Why Static?

The compiler decides at the time of compilation which function has to be called.
Therefore, it is called the static polymorphism.

Function Overloading: Why Static?

For example in the previous example of add function overloading, if we call function

```
    add(2,5)
    or
    add("3","5")
```

the compiler decides at the compile time which function has to be called.

Function Overloading: Problem Scenario

Now Propose the Solution for following scenario with help of function overloading:

During the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

Function Overloading: Problem Scenario

During the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

```
float getFee()
{
   float fee = 0;
   //calculate fee
    return fee;
}
```

Function Overloading: Problem Scenario

During the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

```
float getFee()
{
    float fee = 0;
    //calculate fee
     return fee;
}
```

```
float getFee(string semester)
{
    float fee = 0;
    fee = getFee();
    //some code
    return fee;
}
```

Function Overloading: Problem Scenario

During the summer semester, a school charges double fees, and if the student is foreigner it charges 20% more than the regular summer fees.

```
float getFee()
{
   float fee = 0;
   //calculate fee
    return fee;
}
```

```
float getFee(string semester)
{
    float fee = 0;
    fee = getFee();
    //some code
    return fee;
}
```

```
float getFee(string
semester, bool
isForeigner)
{
  float fee = 0;
  fee = getFee();
  //some code
  return fee;
}
```

Food for Thought

Have we done Function Overloading Before in this Semester?

Constructor Overloading

Yes, we have.
When we declared multiple constructors with different number of parameters.

Constructor Overloading: Example

```
class Customer
{
  private string name;
  private int age;
  private string city;
  private string contact;

public Customer(string name)
{
       this.name=name;
}

public Customer(string name, int age)
{
       this.name=name;
       this.age=age;
}
```

```
public Customer(string name,int
age,string contact)
{
   this.name=name;
   this.age=age;
   this.contact=contact;
}
}
```

Constructor Overloading: Example

```
class Customer
{
  private string name;
  private int age;
  private string city;
  private string contact;

public Customer(string name)
{
       this.name=name;
}

public Customer(string name, int age)
{
       this.name=name;
       this.age=age;
}
```

```
public Customer(string name,int
age,string contact)
{
   this.name=name;
   this.age=age;
   this.contact=contact;
}
}
```

```
Customer c1 = new Customer("abc");
Customer c2 = new Customer("xyz",12);
Customer c3 = new Customer("aaa",12,"0300");
```

Constructor Chaining: Example

```
class Customer
{
  private string name;
  private int age;
  private string city;
  private string contact;

public Customer(string name)
{
       this.name=name;
}

public Customer(string name, int age) : this(name)
{
       this.age=age;
}
```

Constructor Chaining: Example

```
class Customer
{
  private string name;
  private int age;
  private string city;
  private string contact;

public Customer(string name)
{
       this.name=name;
}

public Customer(string name, int age) : this(name)
{
       this.age=age;
}
```

Conclusion

- If a function behave differently for different scenarios- it is called polymorphism.
- Function overloading is form of a static polymorphism.
- Static polymorphism means compiler decides at compile time which function will be called.
- Function Overloading means same function name with different parameter list.
- The parameter list should be different in data type or it should be different in number of arguments or may be both condition are true.

Conclusion

- Object Oriented Programming offers us a way to extend the functionality of the parent class through function overriding.
- Function overriding is called Dynamic Polymorphism, because it decides at run time which function will be called.
- In function overriding, the name and parameter list of the function should be same.
- When we assign child class object to parent object and we want the functionality of parent object function is replaced with the child functionality, we add virtual keyword in the parent class and override keyword in child class.

Learning Objective

Implement Static Polymorphism through Overloaded functions and Implement Dynamic Polymorphism through Parent Child Relation and Overriding.