Static Polymorphism

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

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

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.