Lesson 7

Polymorphism

By the end of this lesson you will learn:

- Introduction to Polymorphism
- Method Overloading
- Method Overriding
- Constructor Overloading
- Polymorphic Behavior of Objects

Polymorphism Definition: It means ‘Different Forms of the Same Thing’. In other words ‘One Name, Different Forms’.

Example:

✓ Imagine, you and your friend are walking inside your MidTerm exam room. Just before entering the room, your friend is saying, “You are my best friend, brother. Don’t worry about the exam. I got your back. I’ll slide my script a bit right from me, all you need to do is to take a peek and write. ”Now, during the exam, no matter how much you poke your friend, your friend is neither responding nor sliding the script. You are really upset with your friend. And after the exam, your friend is like, “I’m sorry, brother. Please forgive me. I’m your best friend. Let me give you a treat. Lets have some fun in Canteen.”

What do you learn from the story?

- Before the Exam: Your Friend is a Friend.
- During The Exam: Your Friend acts Like Enemy.
- After The Exam: Your Friend is a Friend Again.

The story highlights on different forms of your friend. Sometimes he is like a friend, sometimes he is like an enemy.

- ✓ Another example is when you go to shopping you behave like a customer, when you go to school you behave like a student and in bus behave like a passenger. This is a another example of polymorphism.

In Programming the polymorphic behavior might be with methods, Constructors and Objects.

- Method Overloading.
- Method Overriding.
- Constructor Overloading.
- Polymorphic Behavior of Objects.

Method Overloading:

As the name suggests, it happens among methods. There are some conditions of method overloading. If and only If these conditions are met, we can say that there are method overloading among those methods.

These conditions are:

- Methods MUST be in same class.
- Method Name MUST be same.
- Method Parameter MUST be different.
- Method Return Type may or may not be same.

Method Overloading Example:

```
public class MyClass{

        public void add( ){. . .}

        public void add(int a) {. . .}

        public void add(int a, int b) {. . .}

        public void add(double a, double b) {. . .}

        public void add(int a, double b) {. . .}

        public void add(double a, int b) {. . .}

        //public int add(int a) {. . .}
}
public class YourClass{
        public void add( ){. . .}
        public void add(int a) {. . .}
}
```

Here all add methods are overloaded methods. All of the conditions for overloading is fulfilled.

Method Overriding: Again, it happens among methods. There are some conditions of method overriding. If and only If these conditions are met, we can say that there are method overriding among those methods.

These conditions are:

- Methods MUST be in two different classes.
- There MUST be inheritance between the two classes.
- Method Name MUST be same.
- Method Parameter MUST be same.
- Method Return Type MUST be same.

```
public class Alpha{
        public void show( ){. . .}
        public void print( ){. . .}
}
public class Bravo extends Alpha{
        public void show( ){. . .}
        public void print(int a){. . .}
}

public class Charlie extends Alpha{
        public void show(int a){. . .}
        public void print( ){. . .}
}
public class Delta{
        public void show( ){. . .}
        public void print( ){. . .}
}
```

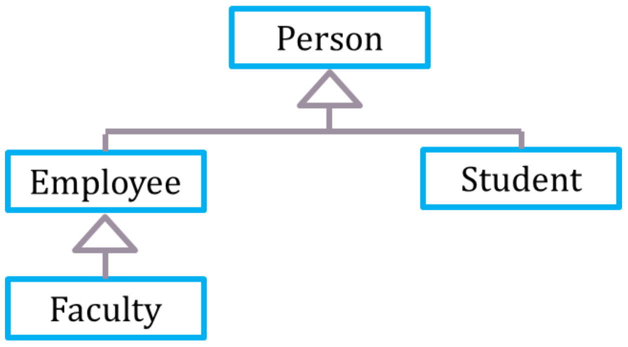
Constructor Overloading: As the name suggests, it happens among Constructors. There are some conditions of constructor overloading. If and only If these conditions are met, we can say that there is constructor overloading in that class. These conditions are:

- Constructors MUST be of same class.
- Constructor Parameter MUST be different.

Just look at all the classes we have written this far. All of them has two constructors. One is Empty and the other is Parameterized.

Polymorphic behavior of Objects: According to the Polymorphic Behavior of Objects, An Object Reference of Parent Class can hold an Object of a Child Class.

Before going for an example, let's assume the following Inheritance Tree:



From the Inheritance tree, we can write the following statements:

```
Person p = new Person( );
Employee e = new Employee( );
Faculty f = new Faculty( );
Student s = new Student( );
```

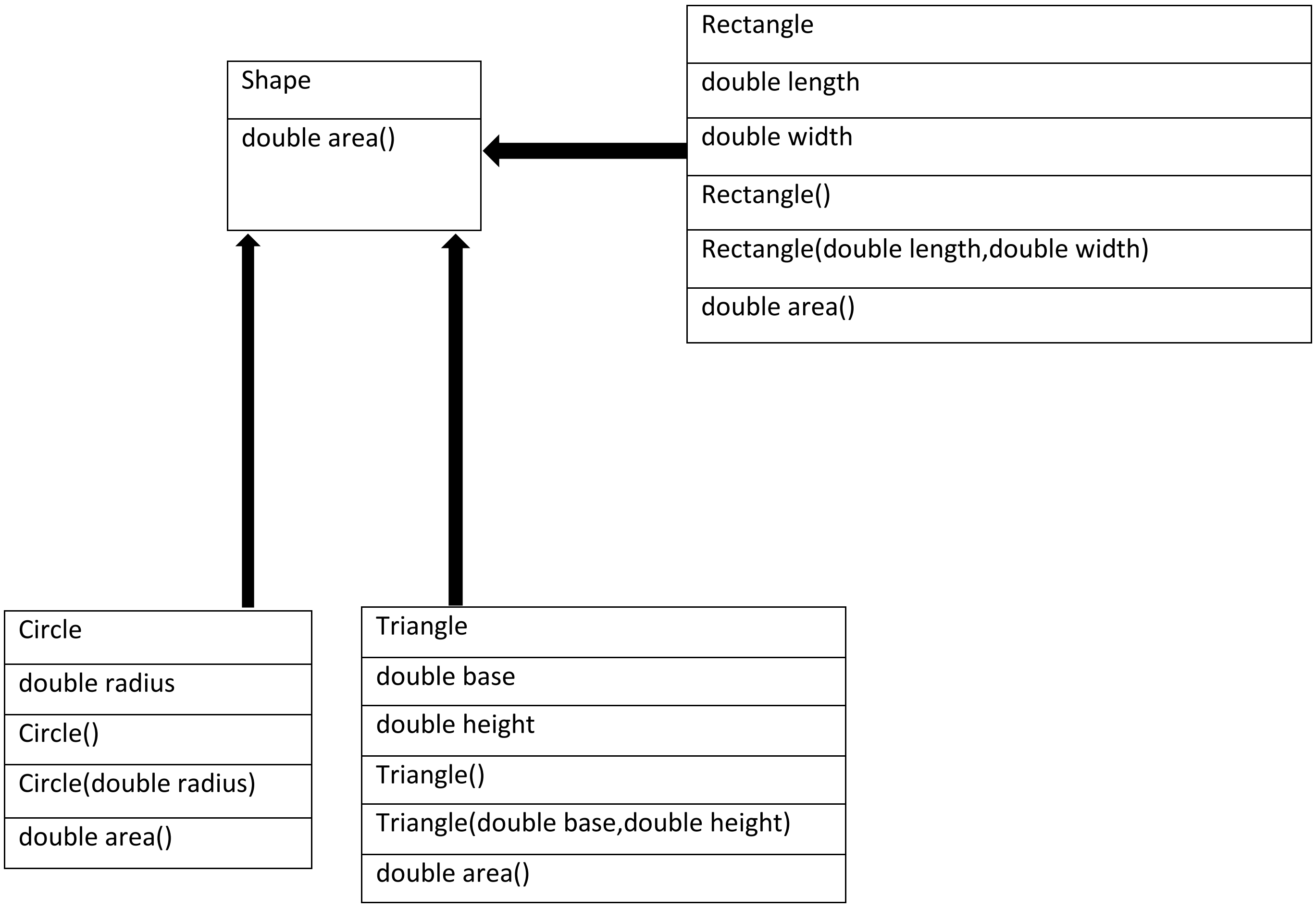
According to polymorphic behavior of objects, we can also write the following statements:

```
Person p1 = new Employee( );
Person p2 = new Faculty( );
Person p3 = new Student( );
Employee e1 = new Faculty( );
```

But we can not write the following statements:

```
Employee e2 = new Student( );
Student s2 = new Faculty( );
```

Exercise: Create the following classes:



Start
Create object array of shape class. Store object of child classes in different index. Now invoke area method of the clild classes using the object of shape class.