Polymorphism

Polymorphism

Polymorphism → many ways to represent

Method Overriding

- When the child class have a same method declared in parent class, This is called Method Overriding

Parent Class

```java
package com.inclass.properties.polymorphism;                    language-java

public class Shapes {
        void area() {
             System.out.println("I am in Shapes");
        }
}
```

Child Class

```java
package com.inclass.properties.polymorphism;                    language-java

public class Square extends Shapes{
    void area() {
        System.out.println("Area: l x l");
    }
}
```

Main

```java
package com.inclass.properties.polymorphism;                    language-java

public class Main {
    // Poly + Morphism = Many + Ways to represent
    public static void main(String[] args) {
        Shapes shape = new Shapes();
        Square square = new Square();
        shape.area();
        square.area();
    }
}
```

Output

I am in Shapes
Area: l x l

Types of Polymorphism:

1. **Compile Time / Static Polymorphism**: this is achieved by method overloading , (same name for multiple constructors but parameters are not same)
2. **Runtime / Dynamic Polymorphism**: this is achieved by method overriding (same name for methods in inheritance)

Code

```
package com.inclass.properties.polymorphism;                language-java

public class Main {
    public static void main(String[] args) {
            Shapes square = new Square();
            triangle.area();
        }
}
```

Output

Area: l x l

Important

This is also called late binding.
Parent obj = new Child(); is known as Upcasting

reference → determines what function would be accessed
object → determines which function would be accessed

Java determines this using Dynamic Memory Dispatch, which is determined at Runtime

Override

- `@Override` is used to check for overriding

Child Class

```java
package com.inclass.properties.polymorphism;

public class Square extends Shapes{
    @Override
    void area() {
        System.out.println("Area: l x l");
    }
}
```

Final

- Using `final` keyword restricts method overriding and inheritance
- Methods with `final` works at compile time and is called Early Binding.

Static

- When `static` method gets inherited, they don't get overridden
- The method in the parent class will always run no matter form which object.