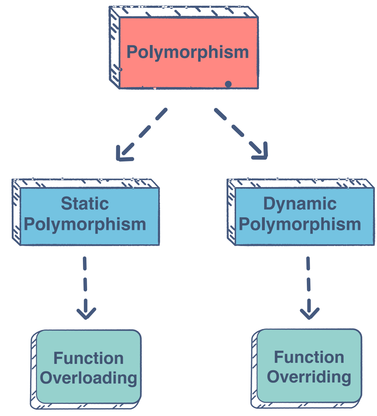
Types of Polymorphism

There are two types of polymorphism:

- **Static polymorphism** is also known as compile time polymorphism.
- **Dynamic polymorphism** is also known as runtime polymorphism.



Static Polymorphism & Dynamic Polymorphism

Static Polymorphism	Dynamic Polymorphism
Polymorphism that is resolved during compile time is known as static polymorphism. Method overloading is used in static polymorphism.	Polymorphism that is resolved during run time is known as dynamic polymorphism. Method overriding is used in dynamic polymorphism.

Example of Static Polymorphism

```java
class Calculator {

  int add(int num1, int num2) {
    return num1 + num2;
  }

  int add(int num1, int num2, int num3) {
    return num1 + num2 + num3;
  }

  public static void main(String args[]) {

    Calculator obj = new Calculator();
    System.out.println("10 + 20 = " + obj.add(10, 20));
    System.out.println("10 + 20 + 30 = " + obj.add(10, 20, 30));
  }

}
```

Run Save Reset

Here, we have two definitions of the same method `add()` in Calculator class. Which `add()` method would be called is determined by the parameter list at the compile time. That is the reason this is also known as **compile time polymorphism.**

Example of Dynamic Polymorphism

```java
// Shape Class
class Shape {

  public double getArea() {
    return 0;
  }

}

// A Rectangle is a Shape
class Rectangle extends Shape {   // extended form the Shape class

  private double length;
  private double width;

  public Rectangle(double length, double width) {
    this.length = length;
    this.width = width;
  }

  public double getArea() {
    return this.length * this.width;
  }

}

// A Circle is a Shape
class Circle extends Shape {
```

Run Save Reset

Here, we have three classes `Shape`, `Circle,` and `Rectangle.` `Shape` is a parent class while `Circle` and `Rectangle` are the child classes. The child classes are overriding the method `getArea()` of the parent class. We have child classes objects assigned to the parent class reference. So to determine which method would be called, the type of the object would be determined at runtime. That is the reason it is also known as **runtime polymorphism.**