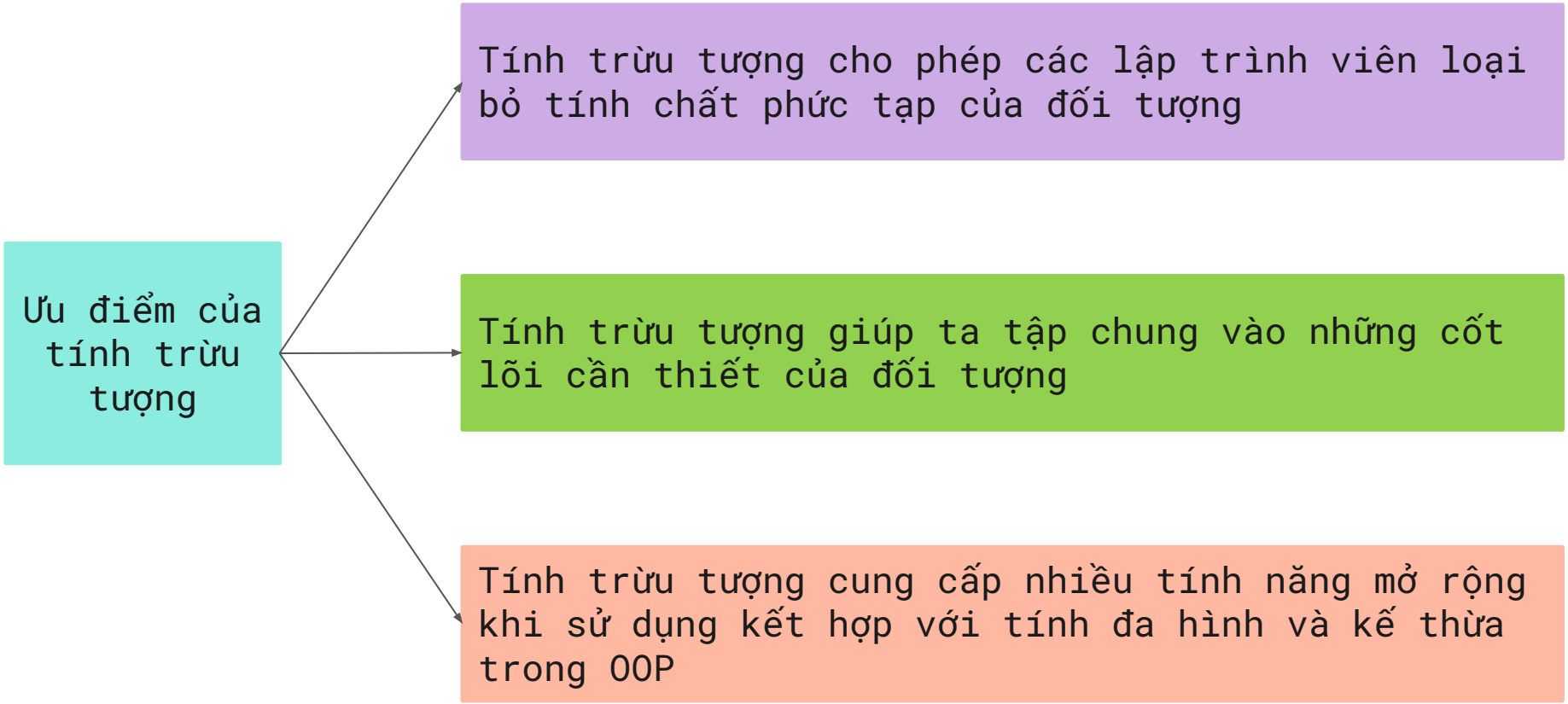
# Tính trừu tượng

# Trừu tượng trong OOP

Tính trừu tượng trong lập trình hướng đối tượng là chỉ nêu ra vấn đề mà không hiển thị cụ thể, chỉ hiển thị tính năng thiết yếu đối với đối tượng mà không nói ra quy trình hoạt động


Ưu điểm của tính trừu tượng
Tính trừu tượng cho phép các lập trình viên loại bỏ tính chất phức tạp của đối tượng
Tính trừu tượng giúp ta tập chung vào những cốt lõi cần thiết của đối tượng
Tính trừu tượng cung cấp nhiều tính năng mở rộng khi sử dụng kết hợp với tính đa hình và kế thừa trong OOP

# Trừu tượng trong java

## Lớp trừu tượng

Là lớp bị hạn chế, không thể dùng để tạo đối tượng (Để truy cập nó phải kế thừa từ lớp khác)

## Phương thức trừu tượng

Chỉ có thể sử dụng trong lớp trừu tượng và nó không có phần thân. Phần thân được cung cấp bởi lớp con

# Lớp trừu tượng

Để tạo lớp trừu tượng ta dùng từ khóa abstract trước từ khóa class
Cú pháp:

```
<Phạm vi truy cập> abstract class <Tên lớp> {}
```

Ví dụ:

```
public abstract class Person {

}
```

# Phương thức trừu tượng

Cú pháp:

```
<Phạm vi truy cập> abstract <Kiểu dữ liệu trả về>
<Tên phương thức> (<Tham số>);
```

```
public abstract class Shape {

    public abstract void draw();
}
```

Lớp abstract

Phương thức abstract

```
public class Square extends Shape {

    @Override
    public void draw() {
        System.out.println("Ve hinh vuong!!!");
    }
}
```

Phần thân được cung cấp bởi lớp con

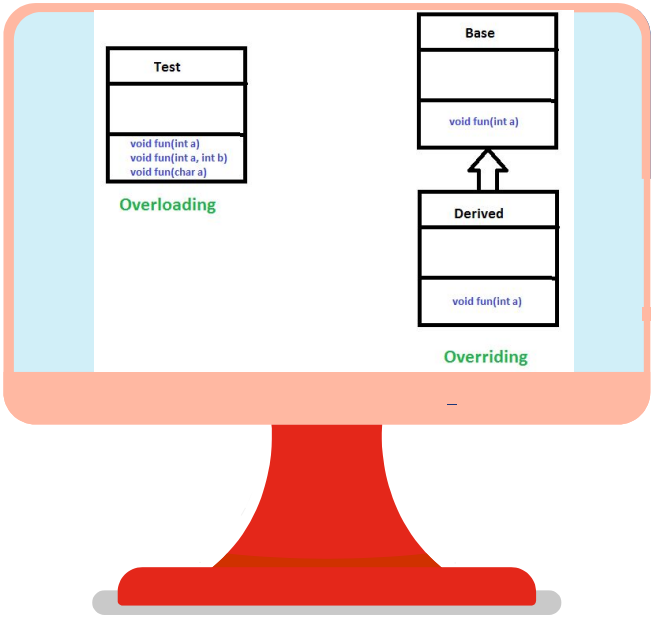
```
public class Circle extends Shape {

    @Override
    public void draw() {
        System.out.println("Ve hinh tron!!!");
    }

}
```

Phần thân
được cung cấp
bởi lớp con

# Tính đa hình

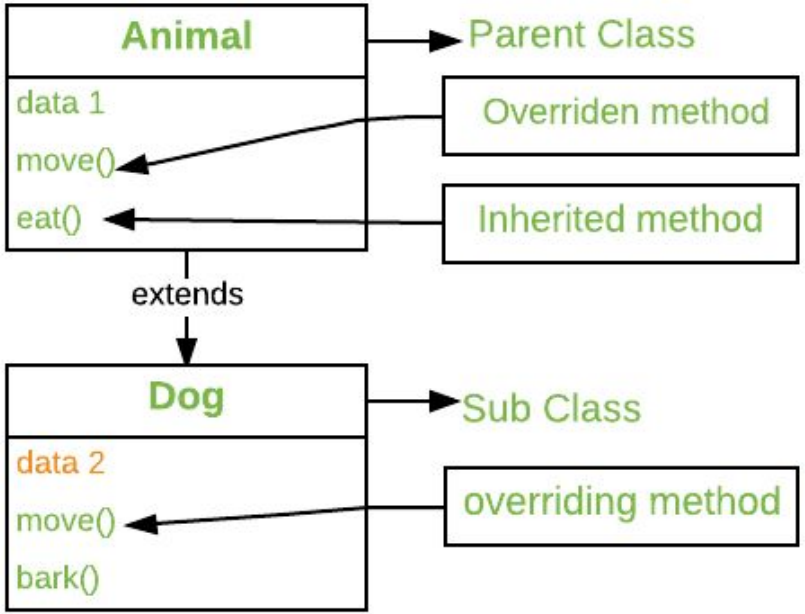
# Tính đa hình (Polymorphism)

**Tính đa hình** là khả năng một đối tượng có thể thực hiện một tác vụ theo nhiều cách khác nhau.
Trong Java, chúng ta sử dụng ghi đè phương thức để có tính đa hình.



# Ghi đè phương thức

Nếu lớp con có cùng phương thức được khai báo trong lớp cha, nó được gọi là ghi đè phương thức



```java
public class Shape {
    public void draw(){
        System.out.println("Hình gì đó!!!");
    }
}
```

Superclass

```java
public class Square extends Shape {
    @Override
    public void draw() {
        super.draw();
        System.out.println("Hình vuông!!!");
    }
}
```

Subclass

# Từ khóa super

Từ khóa super trong java là một biến tham chiếu được sử dụng để tham chiếu trực tiếp đến đối tượng của lớp cha gần nhất.

| | |
|---|---|
| super | Có thể được sử dụng để tham chiếu đến biến instance của lớp cha |
| super | Có thể được sử dụng để gọi các phương thức của lớp cha |
| super() | Có thể được sử dụng để gọi các constructor của lớp cha |

```java
public class Person {
    String name = "Ngoc";
}
```

```java
public class Student extends Person {
    String name = "Ngoc Ban Quyen";

    void display(){
        System.out.println(super.name);
    }

    public static void main(String[] args) {
        Student student = new Student();
        student.display();
    }
}
```

Tham chiếu tới biến name của lớp Person

```java
public class Person {
    void message(){
        System.out.println("Hôm nay là thứ 3!!!");
    }
}
```

```java
public class Student extends Person {
    @Override
    void message() {
        super.message();
        System.out.println("Ngày mai là thứ 4!!!");
    }

    public static void main(String[] args) {
        Student student = new Student();
        student.message();
    }
}
```

Gọi phương thức của lớp cha

```java
public class Person {
    public Person() {
        System.out.println("Phương thức của lớp Person!!!");
    }
}
```

```java
public class Student extends Person {
    public Student() {
        super();
        System.out.println("Phương thức của lớp Student!!!");
    }

    public static void main(String[] args) {
        Student student = new Student();
    }
}
```

Gọi constructor của lớp Person

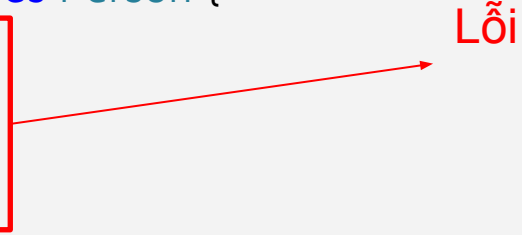
# Từ khóa final

Final trong Java được sử dụng để hạn chế thao tác của người dùng. Final có thể sử dụng trong nhiều ngữ cảnh như:

| | |
|---|---|
| Biến final | Để tạo biến có giá trị không đổi (Hằng số) |
| Phương thức final | Để ngăn chặn ghi đè phương thức |
| Lớp final | Ngăn chặn sự kế thừa |

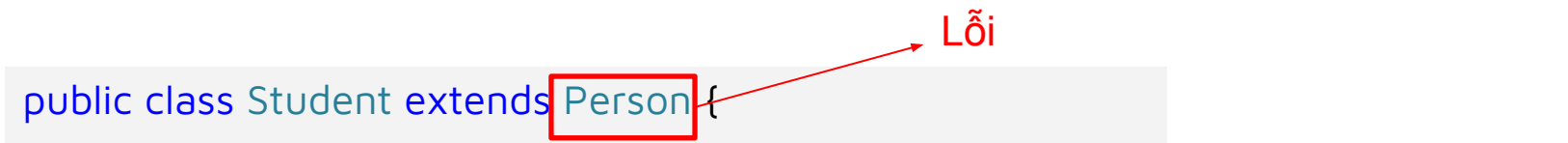
```java
public class Person {
    public final void display(){
        System.out.println("In thong tin");
    }
}
```

```java
public class Student extends Person {
    @Override
    public void display() {
        super.display();
    }
}
```




```
Error: LinkageError occurred while loading main class Student
        java.lang.VerifyError: class Student overrides final method Person.display()V
```

```java
public final class Person {
    public void display(){
        System.out.println("In thong tin");
    }
}
```




```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

        at Student.main(Student.java:2)
```

# Upcasting và Downcasting

Đây là cơ chế được sử dụng để chuyển kiểu đối với kiểu dữ liệu tham chiếu

## Upcasting

Khi biến tham chiếu của lớp cha tham chiếu tới đối tượng của lớp con.

## Downcasting

Là dạng chuyển kiểu chuyển 1 đối tượng là một thể hiện của lớp cha xuống thành đối tượng là thể hiện của lớp con trong quan hệ kế thừa.

# Upcasting và Downcasting