1. **Tìm hiểu từ khoá super và cách sử dụng , cho vd**Khái niệm:   
   **- 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.  
   Bất cứ khi nào bạn tạo ra một thể hiện (hay còn gọi là một instance) của lớp con, một Cách sử dụng:

* Sử dụng từ khóa super để tham chiếu trực tiếp đến biến instance của lớp cha gần nhất
* Sử dụng từ khóa super để gọi đến phương thức của lớp cha gần nhất
* Sử dụng super() để gọi trực tiếp constructor (hàm tạo) của lớp cha gần nhất

VD:   
class Parent {

public Parent() {

System.out.println("Hàm tạo lớp cha");

}

}

class Child extends Parent{

public Child() {

super();

System.out.println("Hàm tạo lớp con");

}

}

public class Main{

public static void main(String[] args) {

Child child = new Child();

}

}

1. **Kế thừa trong hàm khởi tạo , cho vd**Lớp con không được kế thừa các thuộc tính và phương thức private từ lớp cha.Trong kế thừa, các lớp con chỉ được kế thừa các thuộc tính và phương thức có public, protected và defaultphạm vi truy cập (trong trường hợp lớp con và lớp cha có cùng gói).  
   VD:  
   class Person{

public Person() {

System.out.println("Khởi tạo lớp người");

}

}

class Student extends Person{

public Student() {

System.out.println("Khởi tạo lớp học sinh");

}

}

Public class Main {

public static void main(String[] args) {

Student s = new Student();

}

}

1. **Lớp tĩnh  
   Biến static**   
   Muốn một lớp có những biến dùng chung cho tất cả các đối tượng thuộc lớp đó. Ta gọi các biến dùng chung này là biến của lớp (class variable), hay gọi tắt là biến lớp. Chúng không gắn với bất cứ một đối tượng nào mà chỉ gắn với lớp đối tượng. Chúng được dùng chung cho tất cả các đối tượng trong lớp đó.  
   VD; bên cạnh biến thực thể name, lớp Cow còn có một biến lớp numOfCows với mục đích ghi lại số lượng các đối tượng Cow đã được tạo.

Mỗi đối tượng Cow có một biến name của riêng nó, nhưng numOfCows thì chỉ có đúng một bản dùng chung cho tất cả các đối tượng Cow.

numOfCows được khởi tạo bằng 0, mỗi lần một đối tượng Cow được tạo, biến này được tăng thêm 1 (tại hàm khởi tạo dành cho đối tượng đó) để ghi nhận rằng vừa có thêm một thực thể mới của lớp Cow.  
**Phương thức static**   
Lấy lại vd trên ,giả sử muốn numOfCows là biến private để không cho phép ai đó sửa từ bên ngoài lớp Cow.

Nhưng vẫn muốn cho phép đọc giá trị của biến này từ bên ngoài, nên ta sẽ bổ sung một phương thức, chẳng hạn getCount(), để trả về giá trị của biến đó.

Như các phương thức mà ta đã quen dùng, để gọi getCount(), người ta sẽ cần đến một tham chiếu kiểu Cow và kích hoạt phương thức đó cho một đối tượng Cow.

Phương thức getCount()không nên bị phụ thuộc vào các đối tượng Cow cụ thể như vậy.

Để giải quyết vấn đề này, ta có thể cho getCount() làm một phương thức của lớp (class method), thường gọi tắt là phương thức lớp – hay phương thức static - để nó có thể tồn tại độc lập với các đối tượng và có thể được gọi thẳng từ lớp mà không cần đến một tham chiếu đối tượng nào.  
**Khối static**Một khối không là gì khác ngoài các dòng mã nằm trong dấu ngoặc nhọn. Nếu chúng ta khai báo bất kỳ khối nào là tĩnh, thì các khối đó được gọi là khối tĩnh trong Java. Chúng ta sử dụng khối tĩnh khi chúng ta cần khởi tạo các biến tĩnh. Các khối tĩnh tải hoặc thực thi tại thời điểm tải một lớp. Có thể có nhiều khối tĩnh bên trong một chương trình Java. Và nhiều khối tĩnh này luôn thực thi theo cách tuần tự mà chúng được viết trong chương trình.  
VD:  
public class KhoiStatic {

static int num;

static String str;

static {

number = 10;

str = "Day la so 10";

}

public static void main(String[] args) {

System.out.println("Khoi so: " + num);

System.out.println("Khoi ky tu: " + str);

}

}  
  
**Import Static**Nhờ vào việc sử dụng static import chúng ta có thể sử dụng các biến, method static trong một class mà không cần chỉ định rõ class name trong code. Ví dụ như hàm sqrt() là một static trong Math class, thông thường chúng ta sử dụng hàm này bằng cách Math.sqrt(), nhưng với static import chúng ta có thể sử dụng method này mà không cần kèm theo class name sqrt().

1. **Override phương thức**Phương thức override và được override phải có chung kiểu trả về, tên phương thức và danh sách tham số.

Override chỉ xảy ra giữa các lớp có quan hệ kế thừa.

Một phương thức ở lớp cha được override ở lớp con thì phương thức override ở lớp con không được phép thu hẹp tầm vực của phương thức đó ở lớp cha.

Phương thức ở lớp cha được override ở lớp con thì phương thức override ở lớp con không được phép phát sinh những ngoại lệ kiểm tra (checked exception) khác loại hoặc

không có quan hệ kế thừa với các ngoại lệ được ném ra từ phương thức đó ở lớp cha.

Sử dụng từ khóa super để gọi phương thức được override ở lớp cha tại phương thức override ở lớp con.

VD:

public class Animal {

int id;

String name;

float age;   
 void Nhap(){

Scanner nhap=new Scanner(System.in);

System.out.print("Nhap id: ");

id=nhap.nextInt();

System.out.print("Nhap name: ");

name=nhap.nextLine();

System.out.print("Nhap age: ");

age=nhap.nextFloat();

}

void Xuat() {

System.out.println("ID: " + id);

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}