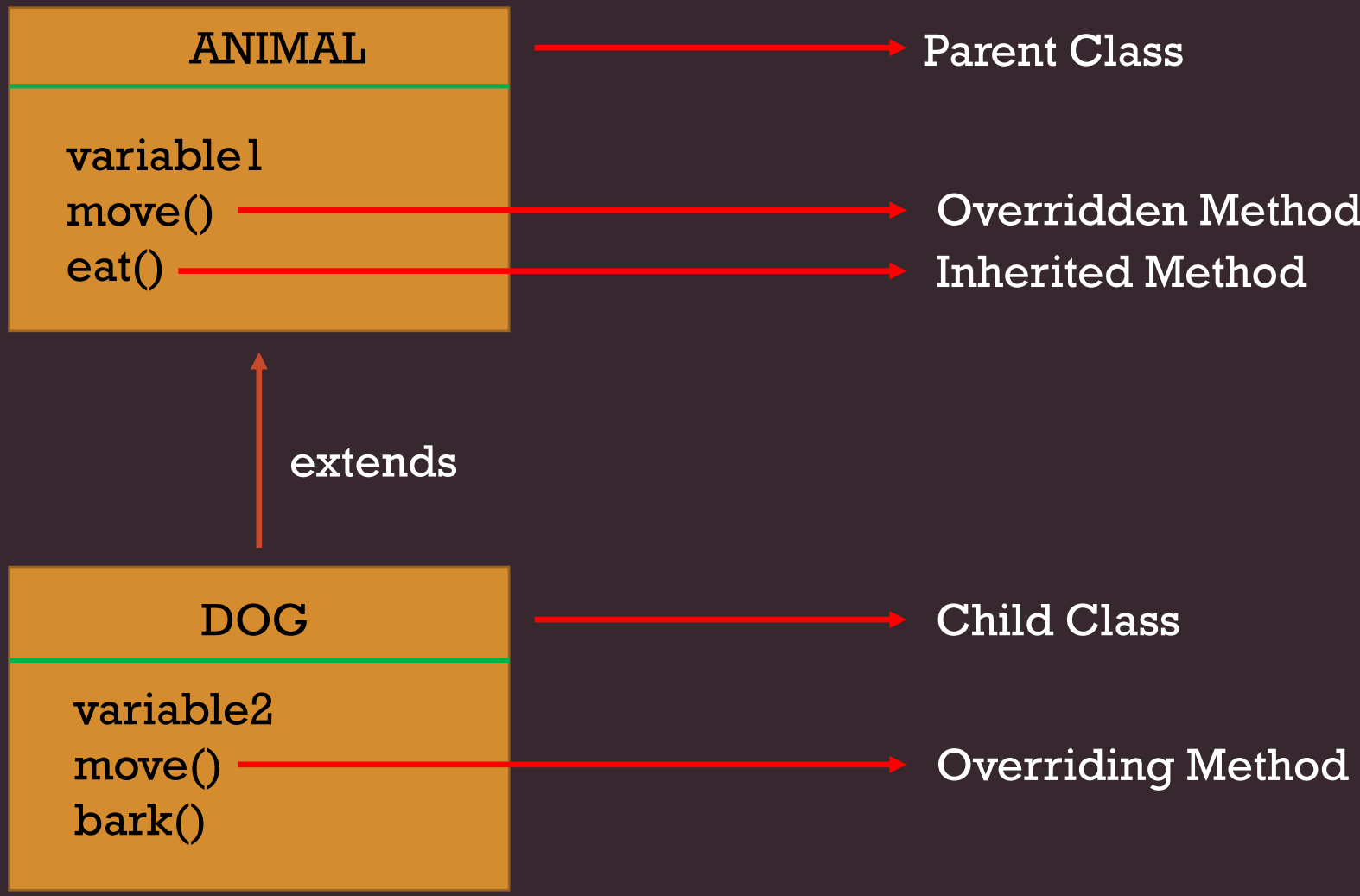
static member's inheritance

- Static members from a super class are inherited as long as access modifier allows it.
- Static variables are shared class variables, it will have single central value for all objects and sub classes.
- Static methods, can be called either using ParentClass.methodName or SubClass.methodName

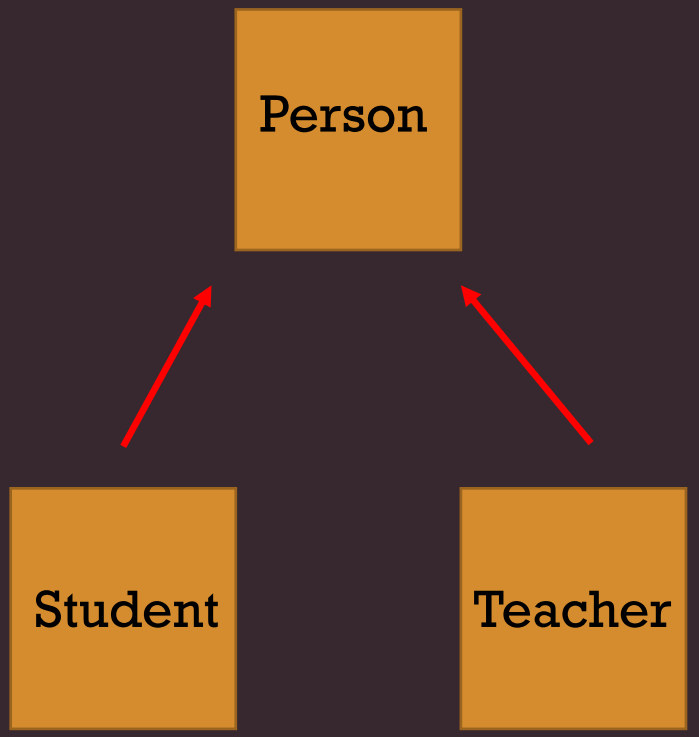
Overriding Superclass Methods

- Sometimes a subclass inherits a method from its superclass, but the method is inadequate for the subclass's purpose. Because the subclass is more specialized than the superclass, it is sometimes necessary for the subclass to replace inadequate superclass methods with more suitable ones. This is know overriding.


ANIMAL
variable1
move()
eat()
Parent Class
Overridden Method
Inherited Method
extends
DOG
variable2
move()
bark()
Child Class
Overriding Method

Method Overriding Rules

1. There must be is-a relationship(inheritance)
2. The method must have the same name as in the parent class
3. The method must have the same parameter as in the parent class
4. Access modifier: Needs to be same or more visible
   - public - > public
   - protected - > protected, public
   - default - > default, protected, public
5. Return type:
   - must be same or
   - covariant type (same class type or sub class type)




```
public Person getInfo() {}

public Student getInfo(){}

public Teacher getInfo(){}
```

Method Overloading	Method Overriding
Method overloading is performed within class	Method overriding occurs in two classes that have IS-A relationship
In case of method overloading, parameter must be different	In case of method overriding, parameter must be same
Access specifier can be changed	Access specifier must not be more restrictive than original method
private and final methods can be overloaded	private and final methods can not be overridden
Return type of method does not matter in case of method overloading, it can be same or different	Return type must be same or covariant in method overriding

super keyword in Java

The super keyword in java is a reference variable that is used to refer parent class objects. The keyword "super" is used with the concept of inheritance.

- Super with variables (super.variable)
- Super with methods (super.method)
- Super with constructors (super())

this vs super

- The keyword super is used to access/call the parent class members (variables and methods)
- The keyword this is used to call the current class members (variables and methods). This is required when we have a parameter with the same name as an instance variable
- We can use both of them anywhere in a class except static areas (static block or static method)

this() vs super()

- this() is used to call a constructor from another overloaded constructor in the same class.
- this() can be used only in a constructor, and it must be the first statement in a constructor. It is used with constructor chaining, in other words when one constructor calls another constructor.
- The only way to call a parent constructor is by calling super(). This calls the parent constructor.

```java
class Shape {
   private int x;
   private int y;

   public Shape(int x, int y) {
       this.x = x;
       this.y = y;
   }
}

class Rectangle extends Shape {
   private int width;
   private int height;

   // 1st constructor
   public Rectangle(int x, int y) {
       this(x, y, 0, 0); // calls 2nd constructor
   }

   // 2nd constructor
   public Rectangle(int x, int y, int width, int height) {
       super(x, y); // calls constructor from parent (Shape)
       this.width = width;
       this.height = height;
   }
}
```

- We have class Shape with x,y variables and class Rectangle that extends Shape with variables width and height
- In rectangle, the 1st constructor we are calling the 2nd constructor
- The 2nd constructor calls the parent constructor with parameters x and y
- The parent constructor will initialize x,y variables while the 2nd Rectangle constructor will initialize the width and height variables

Access Modifiers

- There are 4 access modifiers available in Java:
  - public
  - protected
  - default
  - private
- A top level Java class can have two access modifiers : public and default
- Variables, Constructors and methods can have all four access modifiers

default

- When no access modifier is specified for a class, method or data member – it is said to be having the default access modifier by default.
- The data members, class or methods which are not declared using any access modifiers are accessible only within the same package.

private

- The private access modifier is specified using the keyword private.
- The methods or data members declared as private are accessible only within the class in which they are declared.
- Any other class of same package will not be able to access these members.
- Top level classes can not be declared as private.

protected

- The protected access modifier is specified using the keyword protected.
- The methods or data members declared as protected are accessible within same package or sub classes in different package.

Access Modifier	Accessible to a subclass inside the same package?	Accessible to all other classes in the same package?
default	Yes	Yes
public	Yes	Yes
protected	Yes	Yes
private	No	No

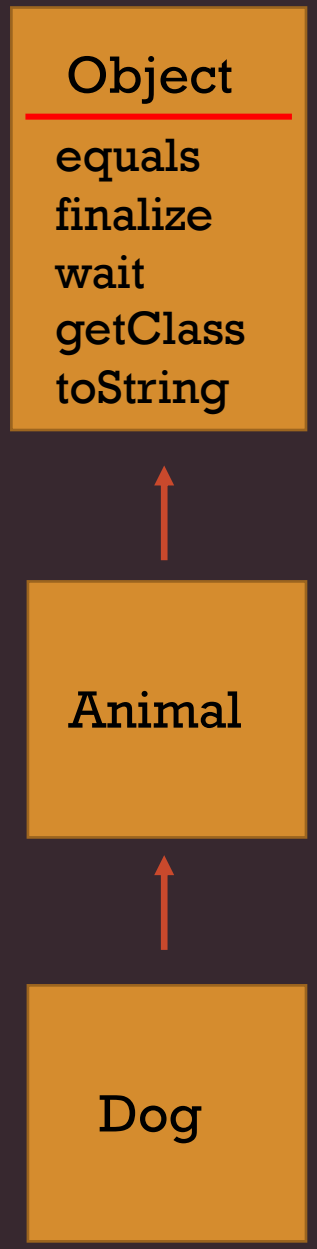
Access Modifier	Accessible to a subclass outside the same package?	Accessible to all other classes outside the same package?
default	No	No
public	Yes	Yes
protected	Yes	No
private	No	No

Object Class

- The Java API has a class named Object, which all other classes directly or indirectly inherit from.

```
public class MyClass{}

public class MyClass extends Object{}
```


Object
equals
finalize
wait
getClass
toString
Animal
Dog