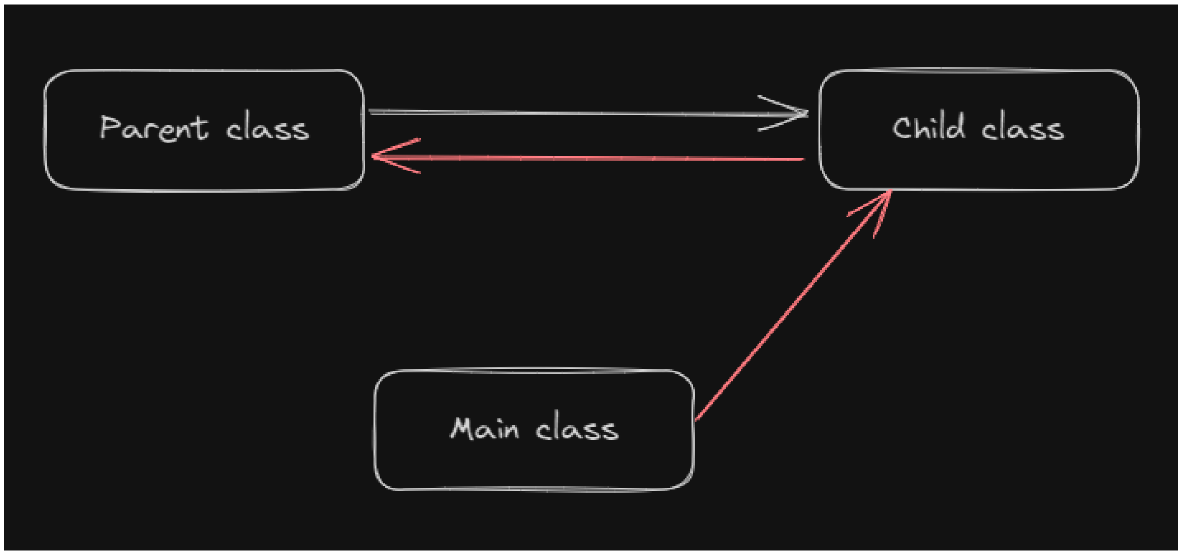
19/10/2023

▼ Java Inheritance

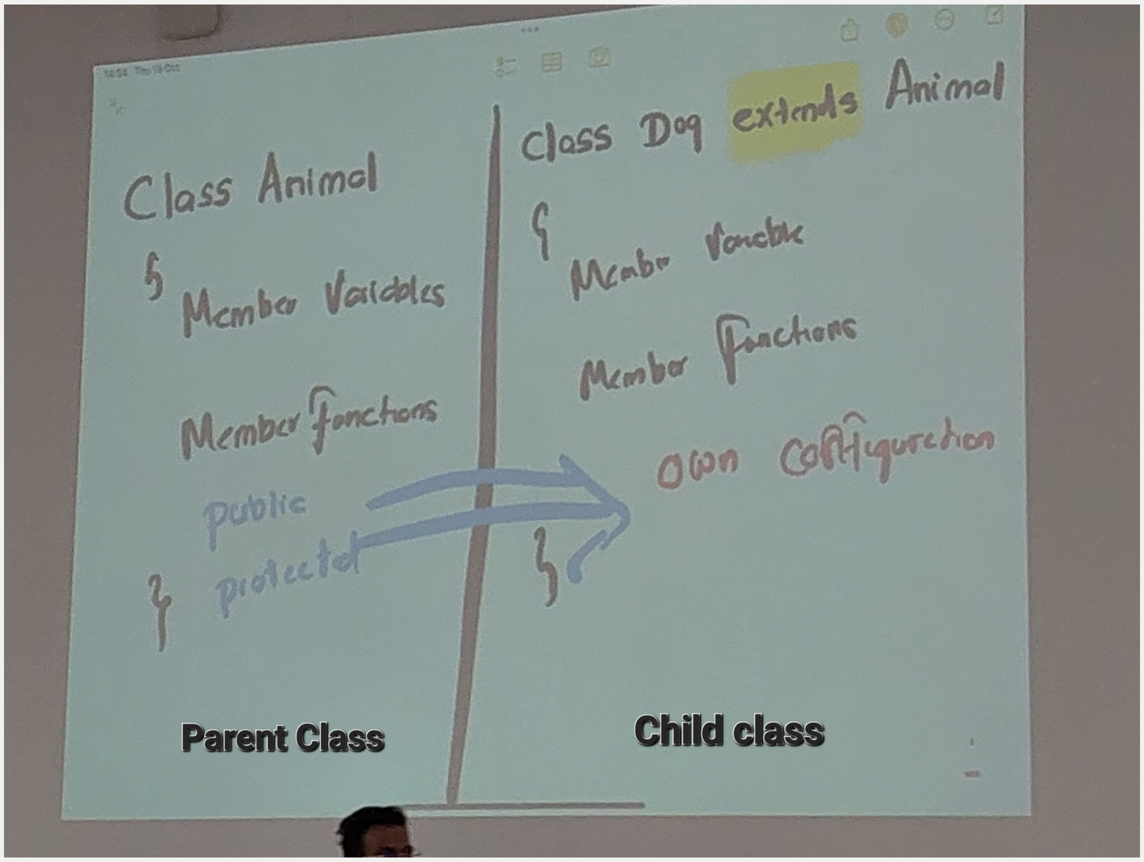
- In OOP inheritance is used to avoid code repetition in programming.
- Using this OOP concept we can minimize the repetition by using child class and parent class. Child class is also known as derived class while parent class is known as super class.

💡 In Java inheritance `extends` keyword used to perform inheritance in OOP.



Example

Assume that there is a class called animal, which is already existing in a java project.
If you want to create a similar type of class, again in the same project, no need to write down the same code, instead of that you can use inheritance OOP concept.



```
class Animal {
//methods and fields
}
//use of extends keyword
//to perform inheritance
class Dog extends Animal {
//methods and fields of Animal
//methods and fields of Dog
}
```

Example

```
class Animal {
//field and method of the parent class
String name;
public void eat() {
Lecture 5 3
System.out.println("I can eat");
}
}
```

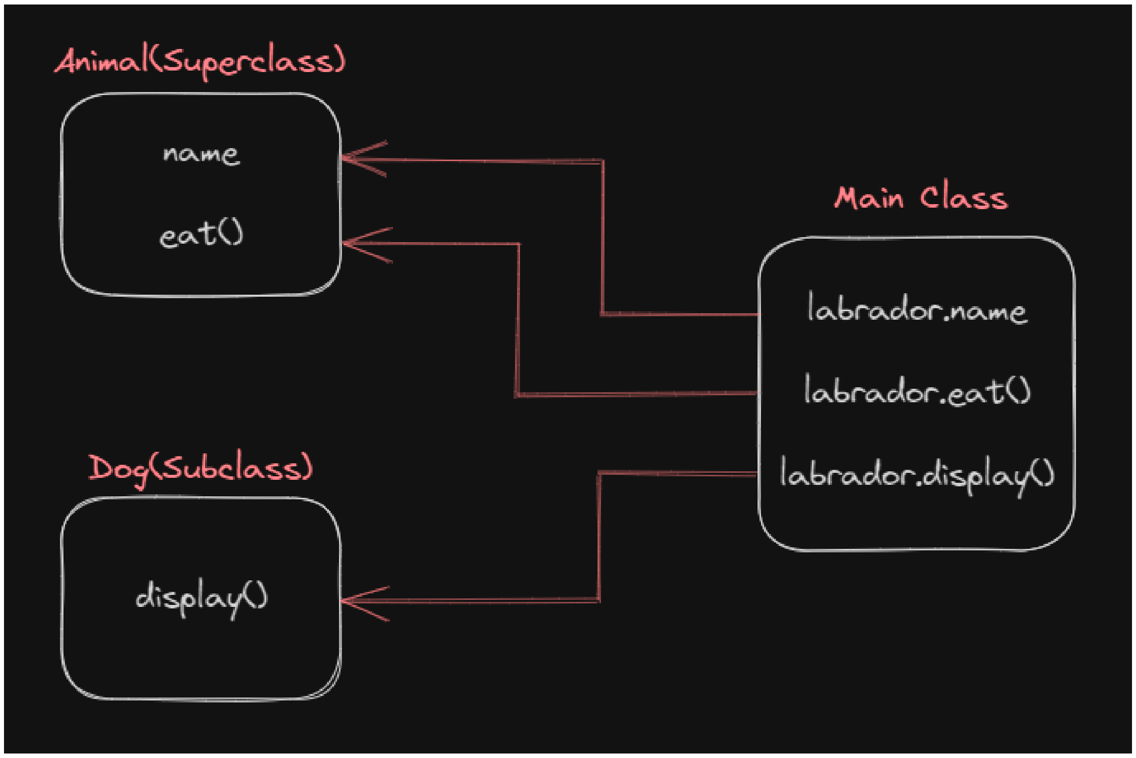
- In above code, `animal` is a existing class which has a variable and method.
- In below code, another class called dog is created by making `animal`class, as the `parent`.
- Eat method and Name variable, will automatically applied to the child class since it use inheritance.

```
//inherit form animal
class Dog extends Animal {
//new method in subclass
public void display() {
System.out.println("My name is " +name);
}
}
```

- In above code, name variable is directly accessed from the parent class, since it use inheritance.

```
class Main {
public static void main(String [] args) {
//create an object of the subclass
Dog labrador = new Dog();
//access field of superclass
labrador.name = "Rohu";
labrador.display();
//call method of superclass
//using object of subclass
labrador.eat();
}
}
```

- In above code, Labrador object is created using dog class and using that object all child class and parent class variables and methods can be accessed.



▼ is - a Relationship

- In java inheritance is - a keyword use to define, connection between child class and parent class.

Example

1. Car is vehicle
2. Orange is a fruit

- In the first example, `Car` is the `child class`, and `Vehicle` `parent class`.

Task 02

- Create a console application with 2 added classes called “Animal” and “Cat”.
- “Cat” is the derived class (child) of the “Animal class”(Base class).
- Inside the “Animal class” create a method.
- Inside the method print “I am an Animal”.
- Inside the “Cat class” create a method and display “I have Four legs”.
- Inside the main method create relevant class object and display as follows.

  “I am an Animal I have four Legs.”

```
// main class

/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
*/
package com.mycompany.lecture_5_1;
/**
*
* @author HP
*/
public class Lecture_5_1 {
  public static void main(String[] args) {
    Cat obj = new Cat();
    obj.hello();
    obj.hi();
  }
}
```

```
// animal class

/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
*/
package com.mycompany.lecture_5_1;
/**
*
* @author HP
*/
public class Animal {
  public void hello(){
    System.out.print("I am an Animal");
  }
}
```

```
// cat class

/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
```

```
*/
package com.mycompany.lecture_5_1;
/**
*
* @author HP
*/
public class Cat extends Animal {
  public void hi(){
    System.out.print(" I have Four legs");
  }
}
```

▼ Java Polymorphism

- Polymorphism is the 3rd OOP concept which available in OOP programming.
- This is a Greek word, Where the meaning of this is `Many Forms`.
- In OOP, polymorphism can be used in 2 different ways.

▼ Method Overloading

- In OOP programming, you cant write down a method, with same structure, in the same class like below.

```
class Student{
  public void StudentInfo{

  }
  public void StudentInfo{

  }
}
```

- The Issue in the above code is,

💡 Both Methods are in,

- Common return type
- Common Access Specifier
- Common method name
- Common parameter list

- In such cases, when calling above method through the object, we cant define to which method that we are calling.

- In programming, there are certain requirements, to have methods with same name and same structure in such cases, method overriding can be used.

```
class Student {
  public void StudentInfo() {
    ---------------
    ---------------
  {
}

class Alex extends Student {
  public void StudentInfo() {
    ---------------
    ---------------
  }
}
```

▼ **Method Overriding**