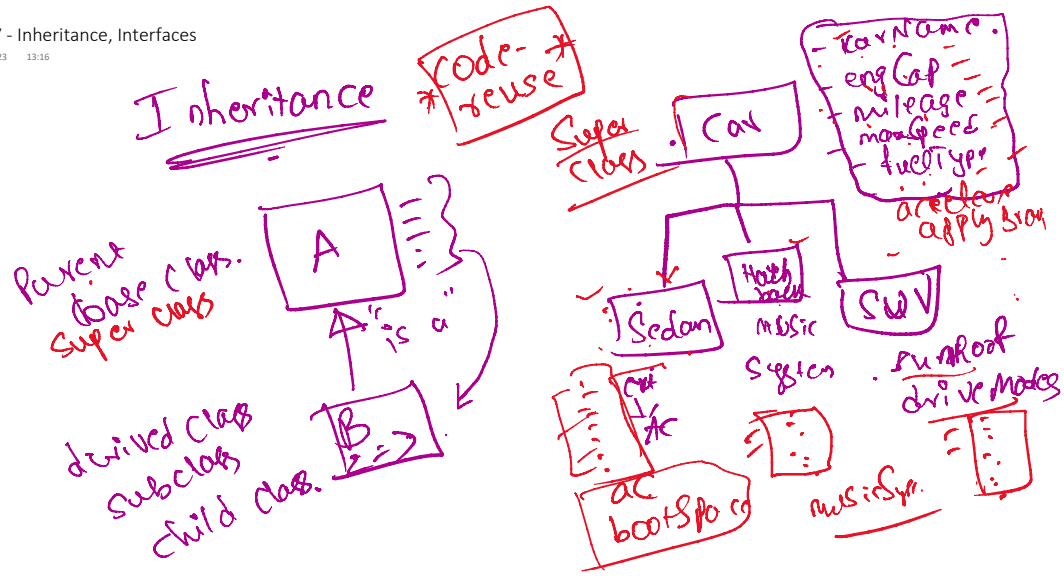
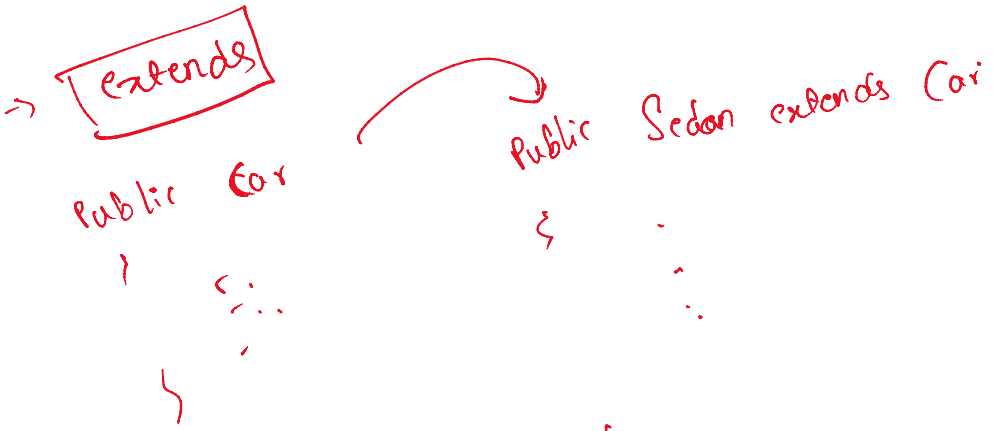
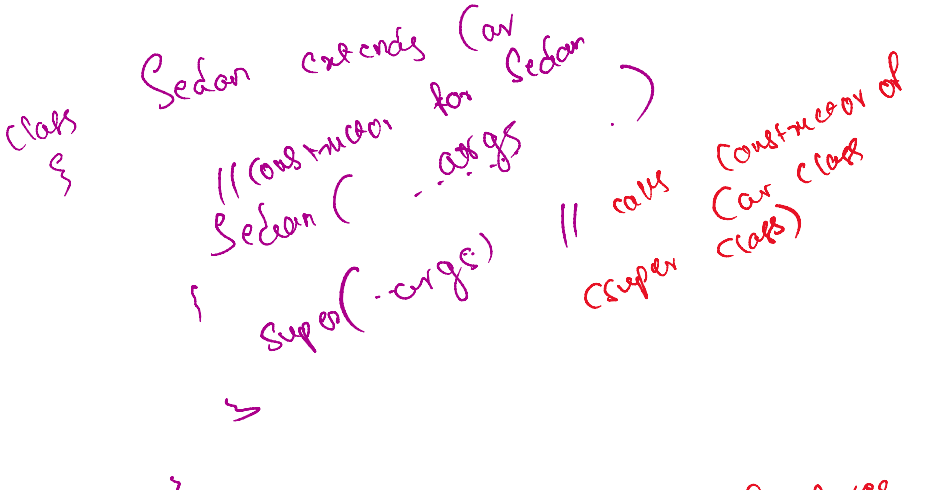
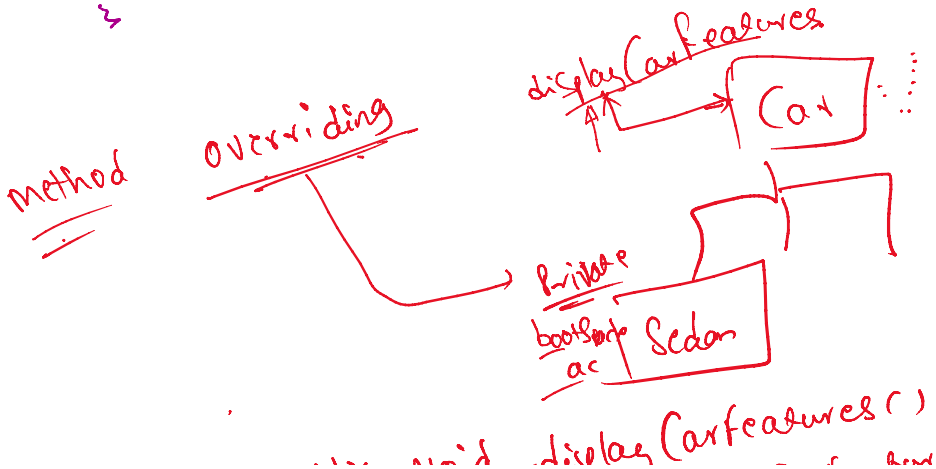
Day 7 - Inheritance, Interfaces

14 July 2023 13:16









displayCarfeatures()

```
public void displayCarfeatures()
{
    super.displayCarfeatures()
    bodyspace
    AC.
}
```

* Create Subclasses for Car

HatchBack

String musicSystem

Constructor()

override displayCarfeatures()

playMusic()

SUV

boolean sunRoof;

String driveMode : ECO, SPORTS, CITY

Constructor

override displays()

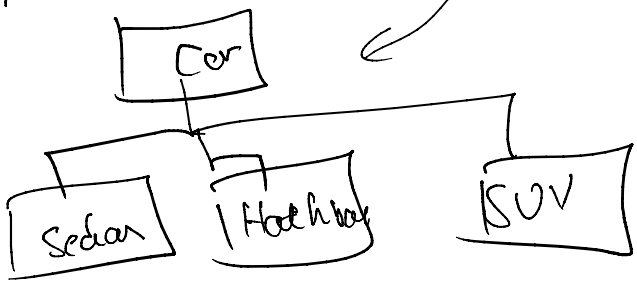
seeSunRoof() → getter

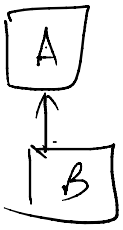
changeDriveMode(String driveMode).

override accelerate()

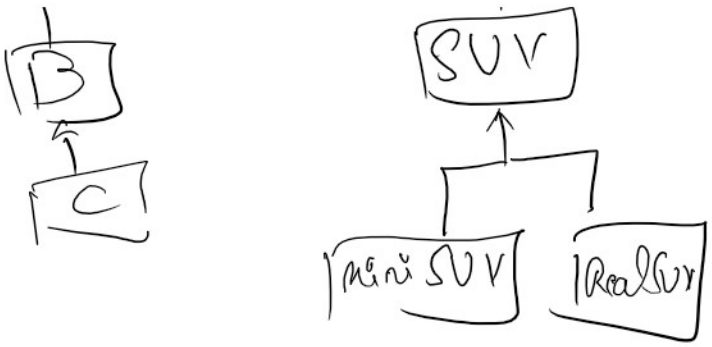
if driveMode
- ECO - accelerating slowly in ECO mode
- CITY - accelerate moderately in CITY mode
- SPORTS → accelerating rapidly in SPORTS mode.

1) Single inheritance/ hierarchical inheritance

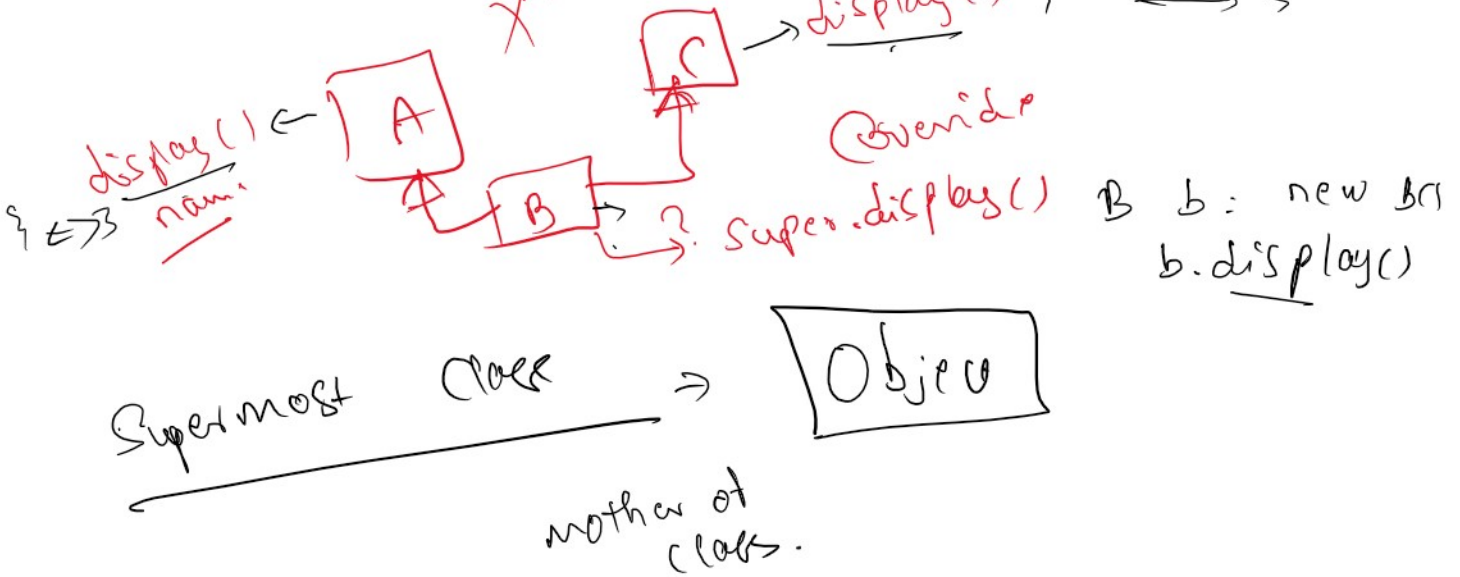




2) multilevel inheritance

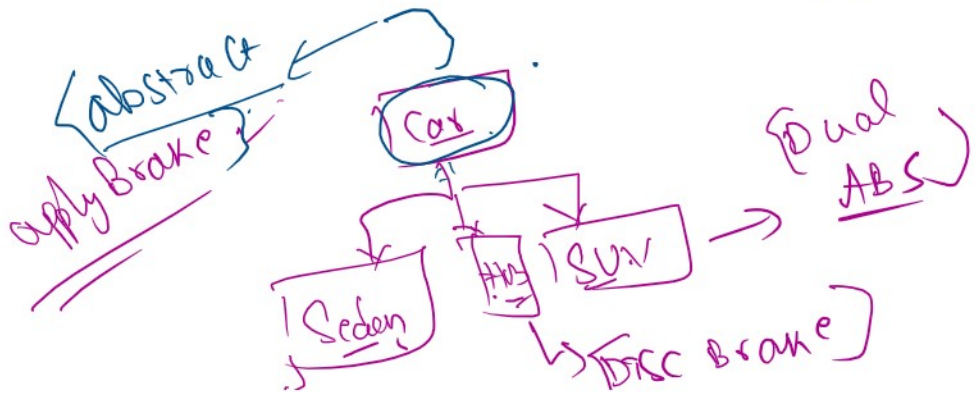


(3) multiple inheritance ✗ NOT SUPPORTED



Abstract classes

- ↳ can't be instantiated
- ↳ abstract method.
  - ↳ blank method
  - → no implementation



Sedan → Disc Brake
(Suspension Brake)

* enforcing sub-classes to implement abstract method.

```java
// MyCar.java
package chinmay.java.inheritance;

public abstract class MyCar {
    2 usages
    public MyCar(){

    }
    // abstract method
    2 usages  2 implementations
    public abstract void applyBrakes();
}
```

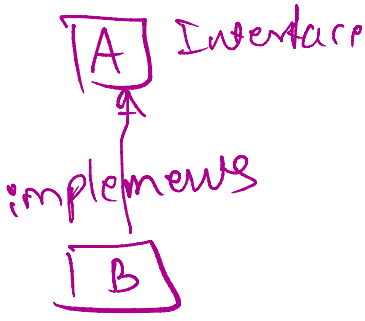
```java
// MySedan.java
package chinmay.java.inheritance;

public class MySedan extends MyCar{

    2 usages
    @Override
    //annotations -> indicate to JVM that following is an overrided
    public void applyBrakes(){
        System.out.println("Sedan car is applying " +
                "brakes with Suspension Brakes");
    }
```

```java
// AbstractDemo.java
package chinmay.java.inheritance;

public class AbstractDemo {
    public static void main(String[] args) {
        MySedan sedan = new MySedan();
        sedan.applyBrakes();
        MySUV suv = new MySUV();
        suv.applyBrakes();
    }
}
```

```java
// MySUV.java
public class MySUV extends MyCar{

    2 usages
    @Override
    public void applyBrakes() {
        System.out.println("Applying brakes in SUV " +
                "Car with Dual-ABS Brakes");
    }
}
```

Interface

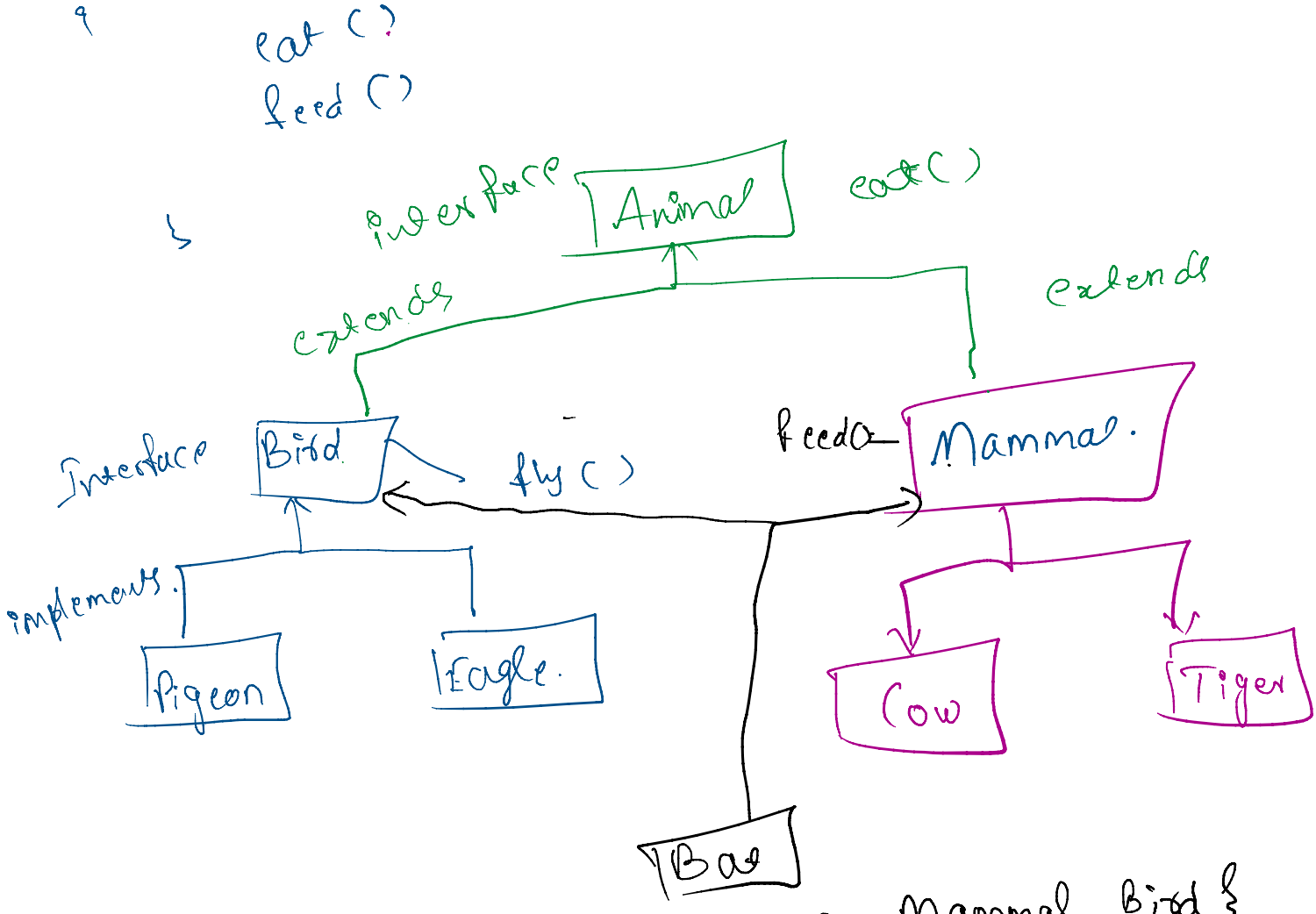
→ all methods are abstract.




```
Public interface Car
{
    void accelerate();
    void applyBrakes();
    void changeGears();
    -
    -
}
```

---

interface Mammal



class Bat implements Mammal, Bird {

}