program

* Multidimensional array

abstraction

Concrete Class →

abstract class

Concrete method

abstract method

Interface

abstraction

data abstraction using private keyword Access specifier

method abstraction (implementation abstraction)

method ની private કોઈ પર રાખવા ઉપયોગમાં નહિં.

implement
process ಮಾಡುವ ವಿಧಾನ

Hardware abstraction

ಮಿಷನ್ ಭಾಷೆ ಮತ್ತು ಮಿಷನ್
ಭಾಷೆ ಬರೆಯುವ data copy ಮಾಡುವ
ಮಿಷನ್ abstraction ಕೊಡುವ

abstract keyword.

* ಯಾವ class ನ body ಇಲ್ಲವೋ
concrete class → ಮತ್ತು class
ಅಲ್ಲಿ ಯಾವ ಮಿಷನ್ method concrete
ಇಲ್ಲವೋ.

concrete method
ಎಂದರೆ body ಇರುವಂಥ.

```
void fun()
{
   ==
   ==
}
```

abstract method.

void fun()

{

}

ફંક્શન લિખિંગ ઓન નથી

Concrete class અહીં ઉક દોન

નથી મગ આપળી નિબ

abstract void fun();

કોઈ ર ઘણાગવાં નથી કે નેર

કારણ concrete class ના object બનાર હોનો તો બોલાવાથી fun ના call કરશકો એટલે abstract fun ના કોના ઊ

class Demo
{
    void fun()
    { ← body વગરની bracket મળશે.
    }
    abstract void gun();
}

compile time error ધ્યાન

Conc. class ના અંદ abstract method

ચાલાન નથી.

Demo d = new Demo();
d. fun();
d. gun(); ~~concrete~~ create class

કાય કરાવવું

જ્યા class ના object બનાવ
કરી શકાતા તેવી class
ના માત્ર abstract method
ધરાવે નહિ.
(કારણ body ન નાહિયે)
મળી ની ક્રિયાવતી

abstract class Demo
{
abstract void fun();

}

abstract class ના object
બનાવ કરતા રોક નાહિ.

---

abstract class Demo
{
some abstract method
or
some non abstract
or
All abstract methods
or
all non abstract methods
(concrete)
}

abstract method ની body
child class મદદથી કરાવવાની
inheritance કરાવવાના.

class test extends demo.
{
void fun() overriding.
{

}
} Test t = new test();
t.fun();
મળી child class ના object દ્વારા
કહેલ મેથડ call કરાવવાં.

Q. પ્રેક class માંના 4 abstract method
child મદદથી 2 ની body દીધી
2 ની બાકી રિતી
તર કરાવી?

ક્યાંની ના concrete class હોવા દ્વારા
નહીં n ⟹ n ના implement
પાડિયો.

n method present હોય
અને abstract
(less than) n માં કોઈ child માત્ર
implement કરે અને n ની થાય પણ
class માં abstract
કરવું જોઈએ.
(કારણકે method નું define નહોય
તો class concrete
નહિ)

એપણ આવતા method
concrete હોય તો એપણ
inheritance આપણે abstract
કરાય શકાય

inheritance નો apply?
આ છે.
① to implement new
functionality આપણી
functionality નાખીશું
direct implement કરાવશો.

} Interface
}

interface Demo.
{

abstract method.

abstract void fun();

~~}~~ abstract void fun();
~~ab~~

}

Java 8 च्या पुढे static आणि default method interface च्या आत implement ~~करायच्या असत~~ करू शकतो.

static आणि default keyword वापरून.

interface Demo
{ void fun();

}

हे चालतं. 1201

आपल्या आत प्रत्येक abstract method

implicitely ચાલે છે.

interface Demo
{
public abstract void fun()
{ by default.
}

~~by default~~

interface Demo
{
void fun();
public void fun(),
abstract void run();
public abstract void fun();
}

implicitely આપવાની જરૂર નથી.

interface ના object તૈયાર કરવા
શકે નહિ

Q abstract class માંની abstract
→ method private આપી
final કરવા યોગ નહિ

abstract class સંતાનપણ
કારણ overriding ના હોવાથી
નહિ.

---

interface માં આપણા variable
લઈ શકતા
પણ value દેવી

interface Demo.
{
final int a = 10; ← કારણ ત્યાં સીધી value દે આપણી force કરવામાં final. કારણ

}

આપણી ઓરિજિનલ value દે.
કારણ ત્યાં constructor
નહિ છે.

~~ઈ variable class~~
interface ના object તૈયાર હોય નહિ
માત્ર મારવા આપણા non-static ના

memory માં ઓક્યુપાઈ ન નહીંને
તેથી જ static આપણી
final ગણવો આપણે public
પણ ગણવો.

implicitely public static
final ગણાતાં variable
interface માં હશે.

---

interface માં આપ constructor
ચાલશે નહીં.

child - parent આવી hierarchy
નહીં રહે [illegible]

---

by default interface
માં ~~abstract~~.

method → public abstract
variable → public static
final.

વાળા body વગરના મેથડ ની concrete class બનાવવી abstract class માં આવે
class test implements Demo
(interface)
{

}

Super keyword થી સર્વ ... constructor ...

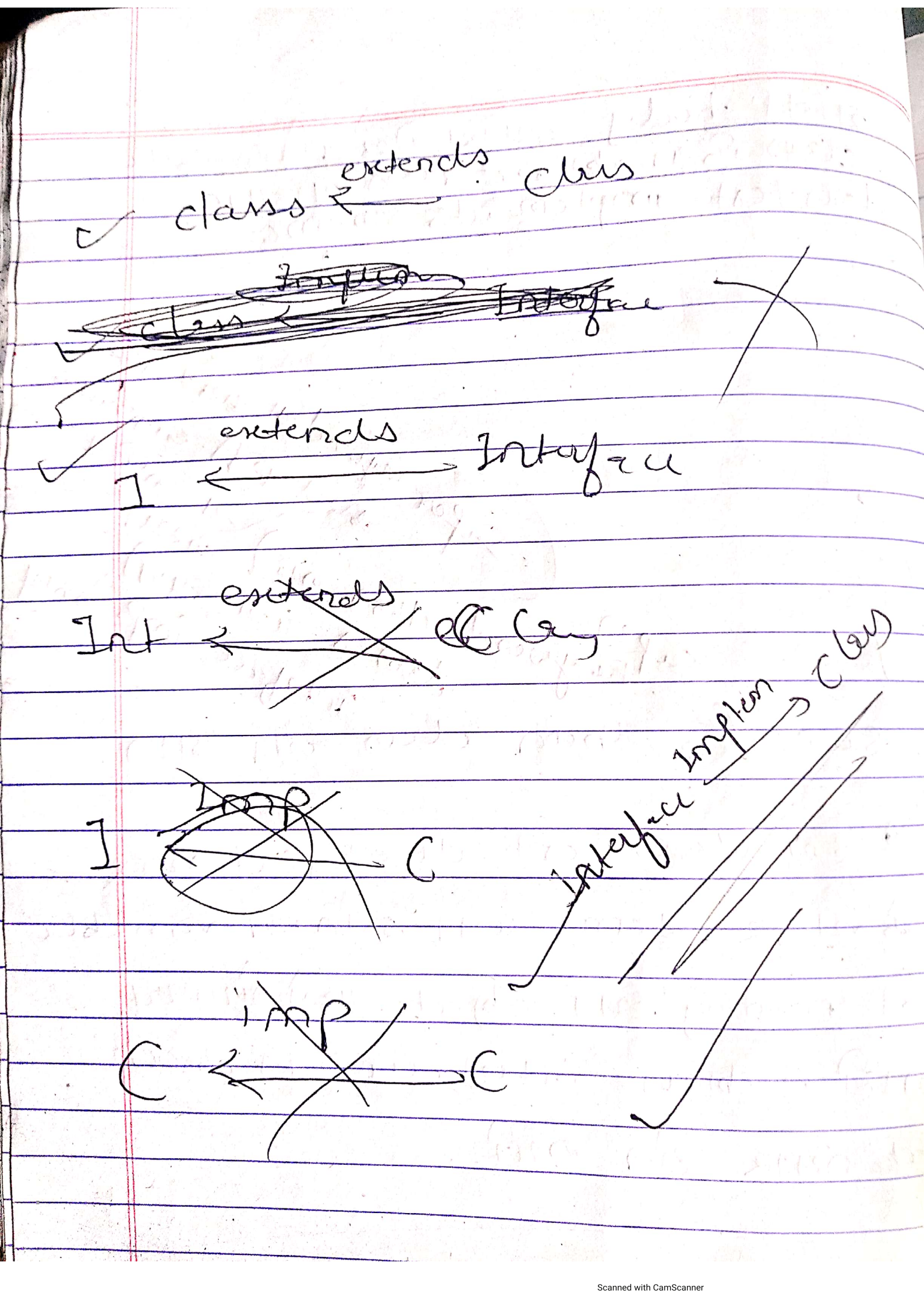
Super keyword ... classes ... interface ...

Super → પહેલા class માં આવે.

રહેલ class test નો object બનાવ કોઈપણ demo માં આવેલા variable નો memory માં object મળશે નહીં. કારણ interface નો object ન બનાવ શકે નહી.

class test નો object બનાવ કરાયા ... → ... concrete ... method ... implement ...

... inheritance ... inheritance ... constructor ... null applicable નથી.


class
extends
class
implements
Interface
extends
Interface
extends
implements
class
Interface
implements
class

Interface જ્યારે class implements

જો કોઈ interface માં એક
abstract method ન હોય તો
પણ કોઈ method નથી તે
empty interface કહેવાય.

Marker interface
↓
JVM ને special behaviour

1. cloneable
2. serializable
3. Remote

Runable ← class આહે.

બાકી શબ્દ able interface
આહે. exception → Runable

abstract class
{
abstract method
or
concrete method
}

~~ક્યારે~~ કો। ગણો ઓવરરાઇડ

કારણ।

→ overriding → virtual થાય છે.

object type નું function call

હોય, reference parent વાળા ઓછા

તરીકેનાં ~~virtual~~ ગણો.

abstraction designing phase

વખતે ગણાતી હોય.

employee

{

}

manager

{

salesman

{

pure virtual funcn ~~આરએમ~~ કહેવાય
એટલે abstract method

જ્યારે call object type પર કરાવાય
ન કે parent ના body કરાવો
ઉપરની બધાવવી in an virtual ક્લાસ
પણ declare કરવાનું - ~~ક~~ એટલે એને
abstract ક્લાસ હોવાય.

---

common funcn હોય: Parent
માં implement કરવાના
∴ abstract class માં જ બધા
concrete method હોય.
જેમ કે working hours ના Same
છે અને જાણકારી ને details
employee માં એ લખેલ હોય
Calc. Sal જે દરેકમાં અલગ છે
એટલે તેને abstract કરી
બાળકો implement કરે.

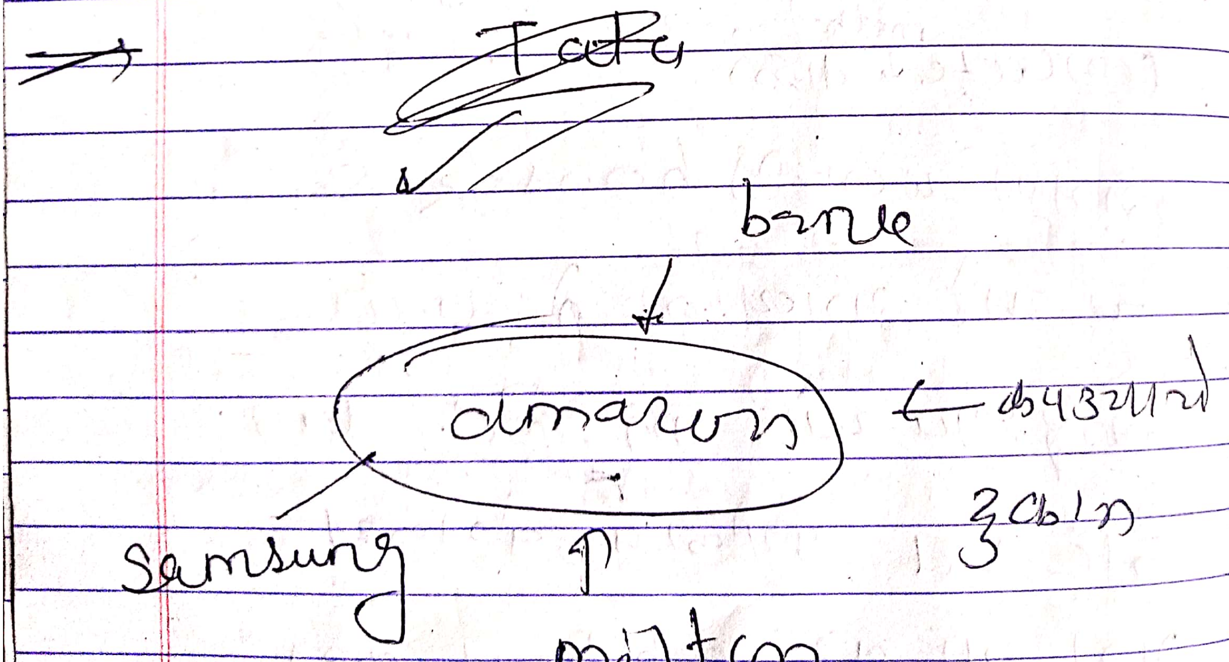
abstract ની આગળ જગ્યામાં concrete
પણ ચાલશે પણ કે કરવાની?

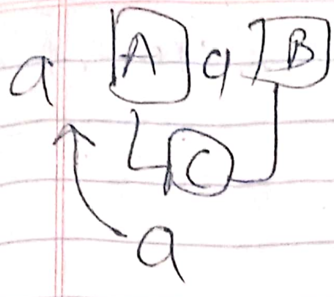
→ ના ફક્ત child class માં method
ની access આપવા માટે જ તો
class abstract ના હોવાથી.

* abstract ની આગળ જગ્યામાં abstract
method ચાલશે.

⊖ પણ interface માટે પણ જગ્યામાં
abstract ના આપવા
મળે છે? ના

→

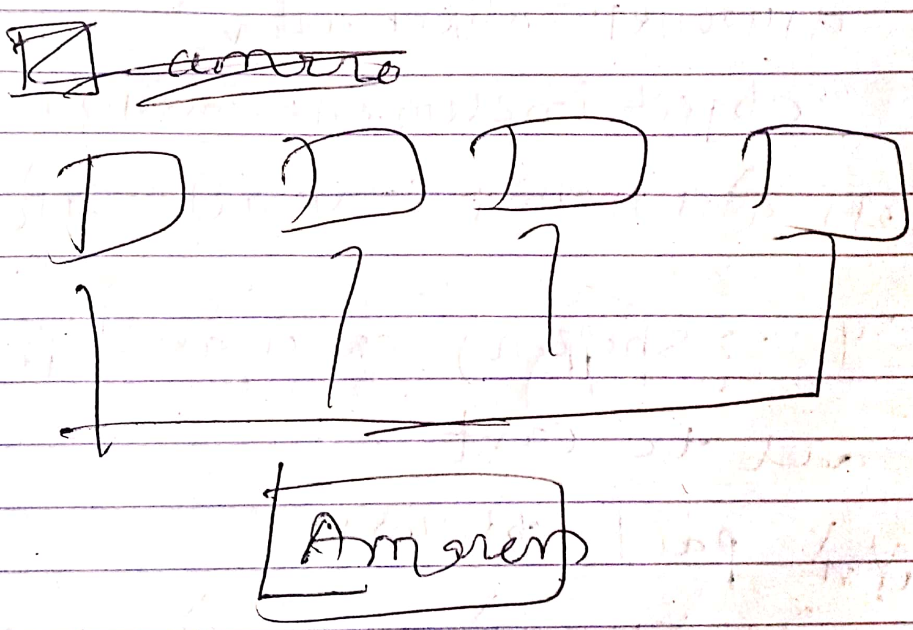




multiple inheritance
ambiguity problem

C++ → virtual base class
ને overcome કરવામાં.

પણ Java નથી કારણ
Java ન ~~~~ explicit virtual
નથી ∴ Java ન multiple inheritance
ચાલતું નથી.



તે ચાલશે નહીં.

Java માં multiple inheritance ચાલતું
નથી ~~~~ કારણ virtual ની જેમ નથી
implicitly

તે overcome કરવા માટે interface આવ્યાં

→ મહારાજાએ કે abstract class નું

* જો કોઈ abstract class પૂર્ણ

child class

multiple inheritance હોય નહીં

તેથી interface આવ્યાં

કારણ કે multiple inheritance

નાં behaviour જાણવા માટે.

interface → specification

contract

વાપરનાર → object user

object implement કરનાર।

આ બંનેની વચ્ચેનો contract છે.

e.g → shopping cart એ shopping

add to cart

પૈસા paid કરાવ્યા।

પૈસા cut કરી કરાવ્યા કે amazon

ને નહીં માહિતી

પણ કઈ રીતે કરાવ્યા કે માહિતીથી

implement કરનાર bank આપે