28-01-2023

Task 1 - Remove static keyword from main function.

means public void main (String[] args)

* Compilation of a fn require - only prototype

no compile time error, JDK have no problem.

Run time error occur, JVM can not run main fn.

~brief answer in Java PDF questions.

Round - 2

OOPS concept
- Class ✓
- object ✓
- incaptulation
- polymorphism
- Inheritence
- abstraction

classname

if we create any fun. static in a class. and we are calling it outside that class then.

mendatory classname.fn();

it we are calling inside the class where we create. it is not mendatory we can directly call it. fn();

but best is to classname.fn();

Incaptulation - बहुत सारी entity को combine करके class में रखा जाता है।

class - unit. inside of it
- Data - Data memberes
- function - member functions

Task 1 ⇒ Incap.Java

Polymorphism - poly - many बहुत सारे
morf - types प्रकार Iversion

Overloading
- function overloading
- ~~Op~~ Constructor overloading

Overriding

in java no operator overloading

* if there is only single object then static variable and instance variable behaves similarly.

28-01-2023

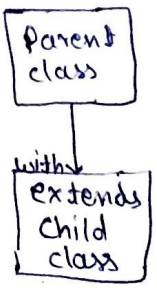
Polymorphism – min two functions are required for archiving

Inheritence – min. two class are required for archiving

Inheritence –

Parent class – Base class – Supperclass

Child class – Derived class – Subclass



keywords – extends → child class have

implements → interface concept

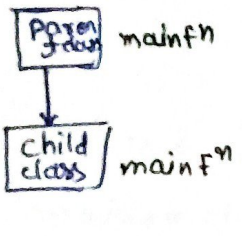
parent class – ABC
child class – XYZ

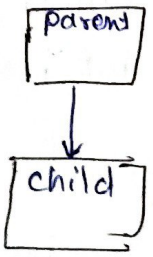
Inheritence

```
class XYZ extends ABC
```

types of inheritence ⇒ 5 types ⇒

1st type – Single inheritence. task 1. Inher.java

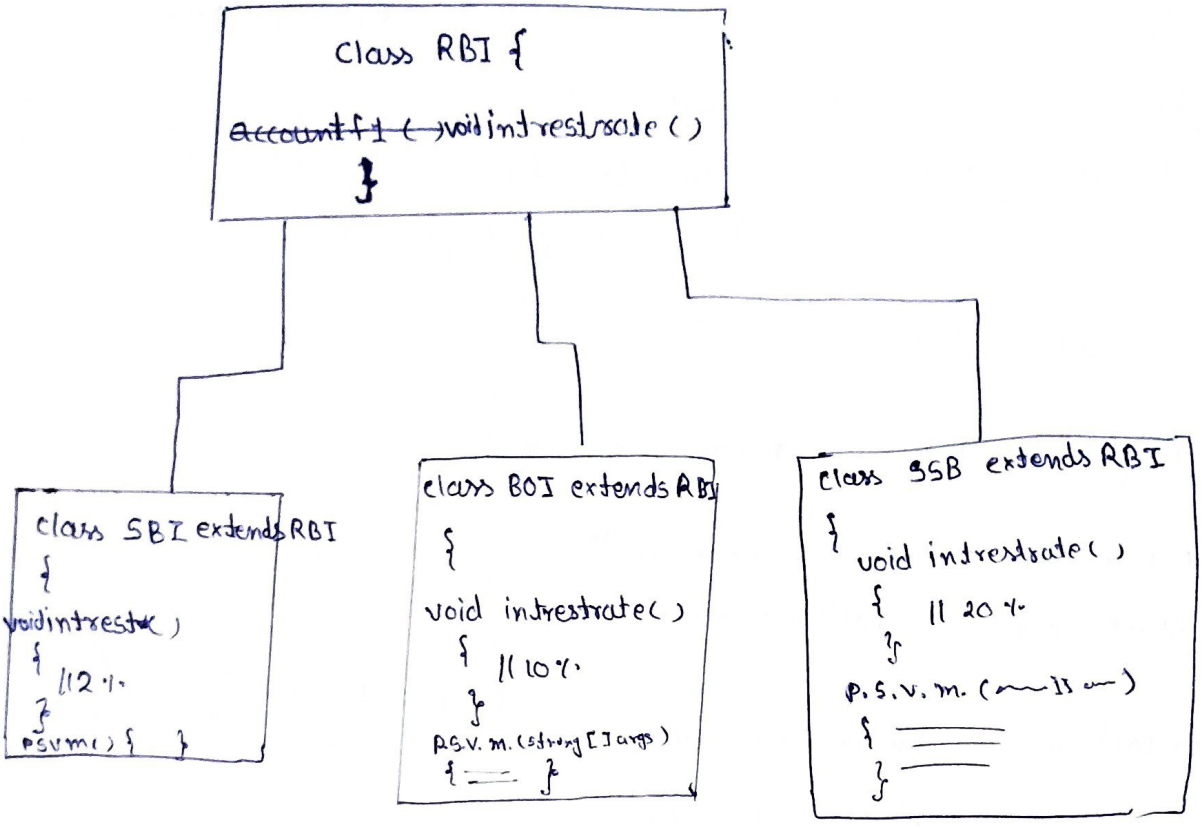




because इसके सारे fn पर child class का अधिकार है। अब child class मे main fn है तो वो सारे fn को access कर पाएगी

task 2 – parent 3 fn + main fn
child 2 fn ← we have not declare here main. now क्या parent अपने child के fn को access कर पा रही है, कैसे ?

28-01-2023



ABstraction - function without definition, have only prototype

without inheritence abstraction have no existence

inheritence boss ☺ → Abstraction
→ Polymorphism (fun. overriding)

task - Abst.java → RBI

Date - 31 - 01 - 2023

When two classes are created and they want to communicate →
1. by creating object of that class
2. by Inheritence

Javafolder B.Java

Single inheritence -: maximum class = 2

masterclass - which have main fn

slave class - without main fn

---

polymorphism - method overridding → needs two class

- Prototype same
- deffinition different

```
class Parent
   {
    void f1 ( )
      { sop ("Hi");
      }
   }
class child extends parent
   { void f1()
      { Sop ("Hello");
      }
   psvm (~~)
   { chid c1 = new child();
     c1 . f1();
   }
```

output = "Hello"

main static { }

inside main we can not not call non static fn without creating object.

→ here - object creation → child 's f() is being called

~~BBB java~~ Bcc . java

Java floder BBB.java

[illegible] object of parent class

next is
but use super . f1();

Javafolder BBB.java

★★ Super keyword

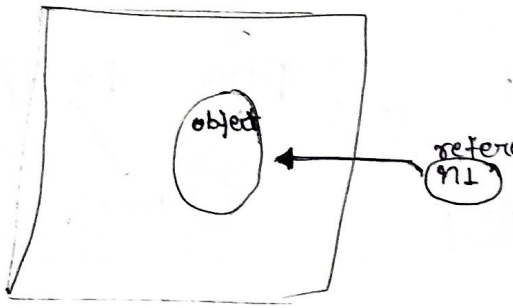
31 - 01 - 2023

In Java reference variable

Nidhi n1 = new Nidhi();

→ object created here with □ memory location

reference variable → have information of that memory location (object)



★★★ explained in Java PHASE 2-1

reference variable → represents → object → represents → class

Nidhi n1 = new Nidhi();

n1 → reference variable

new Nidhi() → object (memory location □, imagine memory of 1 GB)

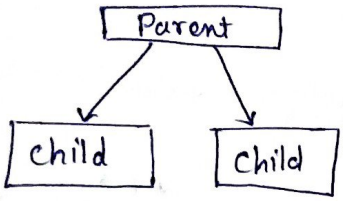
02/02/23

Inheritence →

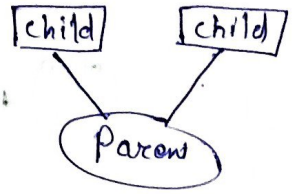
1] Single

maximum class required = 2

2] Hierarchical



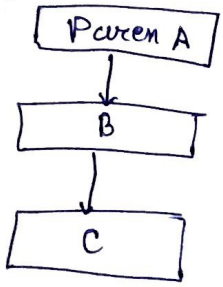
(or)



★ more then one child

```
class A { ===
        }
class B extends A { ===
                  }
class C extends A { ===
                  }
```

3] Multilevel –



★ maximum class required to archive multilevel inheritence = 3

```
class A { ===          दादा
        }
class B extend A       पापा
        { ===
        }
class C extends B      बेटा
        { ===
        }
```

★★ super keyword is used for calling its imidiate parents function. #explained latter

(2 Feb

4] Multiple Inheritence → two parent → one child

not supported in java → proper reasone in JAVA PHASE 2-1 .docx

But possible through → interface - we will study later.

But through classes → we get compile time error.

and here super keyword also does not help.

super keyword → imidiate parent

(But यहाँ तो दो parent हो जाएंगे, तो कौन से parent को call करेगा compile ambiguous (अस्पष्ट) situation (स्थिति))

```
class A {
    void f1()
    {
        ==== H
    }
} void f
```

```
class B {
    void f1()
    {
        ==== Hi
    }
}
```

```
class C extends A, B
{
    void f1()
    {
        ==== Hello
    }
}
psvm(:——)
{ object बनाएंगे }
और f1() को call करेंगे
```

हम चाहते है कि parent class का f1 चलाना है, तो super का use करेंगे पर, parent तो दो है imidiate parent (पापा) अब super काम नहीं करेगा।

when we have done overridding.

★ अगर हमने overridding नहीं की तब भी support नहीं करेगी। क्योंकि compiler को beleive नहीं है programmer पर। 😊 पता नहीं कब override करें।

5 feb 2023

1st use

Super keyword → used for calling imidiate parent's function.

explained in JAVA PHASE 2-1 Notes

```
class A
{
   void f1 ()
   {
      sop ("class A");
   }
}
```

```
class B extends A
{
   void f1 ()
   {
      super.f1();
      sop ("class B");
   }
}
```

```
class C extends B
{
   void f1 ()
   {
      super.f1();
      sop ("class C");
   }

   public static void main (String[] args)
   {
      C c1 = new C()
      c1.f1 ();
   }
}
```

C → imidiate parent → B

(Me) (papa)

B → imidiate parent → A

(papa) papa के papa

C के imidiate parent को call या fn. करना है। मतलब B के fun. को तो हम C के f1 में super.f1(); लिखेंगे।

और B के imidiate parent को call करना है तो

B के f1 में super.f1();

★ यहाँ f1() function की overriding हुयी है।

parent → parent के parent को।

child → parent

main में super.f1(); नहीं लिख सकते हैं।

क्योंकि ये non static हैं।

और main से आप direct calling करवाना चाहते है without making object. तो नहीं होगा

सिर्फ static fun. को ही direct call कर सकते हैं।

Hybride inheritence → Combination of more then two inheritence.

इसमें multiple भी हो सकता है, तो, ये भी possible नहीं है।

Polymorphism

- fun. overloading - Compile time
- fun. overriding - Run time

7 feb 2023

2nd use of super keyword. -

★★★

Constructor overriding. - for constructor overriding { parameter → same.

```
class A
{
   A()
   { s.o.p ("Parent class's constructor");
   }
}
class B extends A
{
   ------
   B()
   { s.o.p. ("child class's constructor");
   }
p. s. v. m. (String[] args)
{
   B b1 = new B();
}
```

name ⇒ class name.

for overriding

↳ relationship - inheritence require.

here by default written

★★ `super();`

in fn overloading super.fn1(); but in constructor we have no need to call it.

So `super();`

↳ working as object for calling constructor.

& this will call parent class's constructor.

★ It should be very first statment in child class's constructor.

• whether in fun overriding it is not compulsory. we can write anywhere we want.

अगर constructor overriding है तो, parent class का constructor पहले चलेगा, फिर child class का चलेगा।

★ super() लिखा कि ना हमने यहाँ पर, default constructor, 2 nd work is

1 st work - gives value to object variables.

```
class A
{
    A ()
    { S.o.p. (" 1st constructor in A class ");
    }
    A ( int x )
    { S.o.p. (" 2nd constructor in A class );
    }
}
class B extends A
{
    B ()
    { S.o.p ( " 1st constructor in B class");
    }
    B ( int x )
    { S.o.p ( " 2nd constructor in B class");
    }
p.s.v.m. ( ... [] ... )
{
    B b1 = new B ();
    B b2 = new B (2);
```

bydefault — child class

B() constructor की पहली line

super();

& B(int x) constructor की पहली line super();

output

1st constructor in A class
1st constructor in B class
2nd constructor in A class
2nd constructor in B class

A() overload A(int x)

A() override B()

B() overload B(int x)

B(int x) override A(int x)

A() ⇌ A(int x)

B() ⇌ B(int x)

Constructor overloading

A() ⇌ B()

B(int x) ⇌ A(int x)

Constructor overriding

8 feb 2023

final - if any class declared final, it can not be extended by any other class.

```
final class A { }
```

→ A cannot be inherite
मतलब parent नहीं बन सकती

Compilation → JDK → component
Java c → will compile the program
high level language → low level

जो पहले से rules हैं, उसे हम break करते है, code करते time तो Compile time error.

final - if any function declared final, it can not be overriden by child class.

```
final void f1 ( )
    {  sop ("definition is fix");
    }
```

Can't be override but can be called by child class's object.

final class के अंदर के functions को final लिखकर कोई benifit नहीं है, already उन्हे override करना possible नहीं है।

```
final class A
    {
        final f1 ( )
        {
        }
    }
```

e.g. collage है lock है और तुम बोल रहे हो class room के अंदर मत जाना

final - if any variable declared final.
we can only assign the value to that variable once. सिर्फ एक बार

default constructor do not give value to final variables.

if we have declared a local variable final then through any fun we can give value once to that variable.

but if we have declared instance variable as final then
we can assign value once by using constructor
we can not assign value by using function, why ?
∵ reasone at last page. -

fun. instance variable को value runtime पर देता. in normal case

default constructor value to object variable. bydefault	the constructor that we create gives value to object variable if we have not given, defalt value will be given

final instance variable को अगर value देना है, तो तभी दें दो जब declare किया है।

Abstraction – class having one or more then one abstract fun is known as abstract class.

Abstract function – function which do not have defination only have prototype.

```
void f1 ();
```

एक भी fun. class का abstract है, तो class भी abstract हो जाएगी हमें class के आगे abstract लिखना पड़ेगा।

अगर class abstract है, तो उसका object नहीं बन पाएगा।

अब object ही नहीं बनेंगे तो benifit क्या functions का, तो इसके functions को use करने के लिए इसकी child class बनाएंगे।

अब child class में हम abstract functions को override करेंगे। अगर नहीं किया तो compile time error आएगी।

एक भी abstract fun. बनाया – तो class abstract हो जाएगी।
पर जरूरी नहीं है कि abstract class आपने declare कर दी तो उसके अंदर abstract function हो ही।

Trough abstract class – abstraction → (0 to 100 %)
archive

```
abstract class A
{
void f1 ()
  { sop ("Hi");
  }
void f2 ()
  { sop ("Hello"); }
}
```

→ अब इसका object नहीं बनेगा।
सारा काम child class में होगा।

Interface - 2nd way of archiving interface.

through Interface - 100% abstraction can be archived.

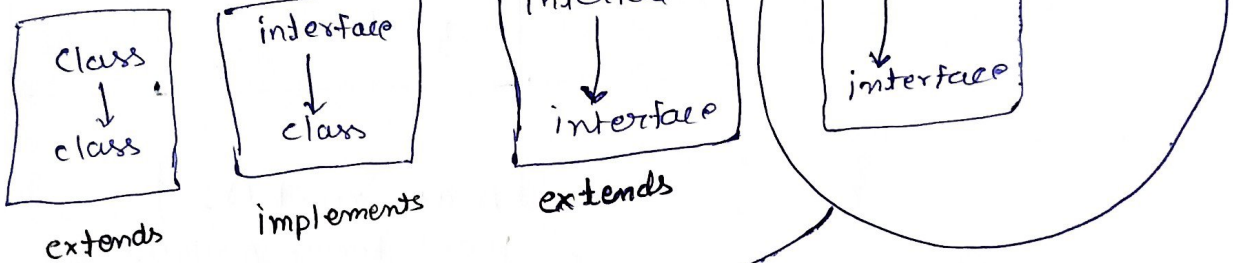
```
interface A
{
    void f1 ();
    void f2 ();
    void f3 ();
    Void f4 ();
}
```

we have not write here abstract keyword with fun. But all the functions are abstract because these are in interface.

Interface ⇒ Special class

before 8.0 version

Inheritence -



similar → extends
diff.-diff. → implements

interface A → interface B → class C impliments A, B

```
void f1 ();      void f2 ();      void f1 ()
                                  {
                                  }
                                  void f2 ()
                                  {
                                  }
```

class complete है,
और उसे interface
उसे inherite नहीं कर सकता
क्यों कि interface में अधूरे functions
होते हैं।

```
void f1 ();      void f1 ();      void f1 ()
                                  {
                                  }
```

** no issue with multiple inheritence

एक ही नाम definition
लिखो।

super.f1() भी
लिखा तो no problem, definition एक ही
है दोनों के f1 की।

interface - we can make reference variable of interface.

```
interface A
  {
    void f1();
    void f2() { sop("Hi"); }
  }

class B implements A
  {
    void f1()
    { sop("Hi"); }

  p.s.v.m(String[] args)
  {
    A a1 = new B();
    a1.f1();
    a1.f2();
  }
```

Hi
Hi

```
A a1 = new B();
```

possible because A related to B

यहाँ सिर्फ reference variable बना है, object नहीं।

```
class A
{
   void f1 ()
   {
     S.o.p. ("f1 in parent class A");
   }
}

class B extends A
{
   void f1 ()
   { sop. ("f1 in child class A");
   }

p.s.v.m. (~~ [] )
{
   B b1 = new B ();
   b1. f1 ();
}
```

f1 in child class A

```
B b1 = new A ();
b1. f1 ();
}
```

not posible (X)

```
A a1 = new B ();
a1. f1 ();
}
```

f1 in child class

in interface -

after 8.0 version →

- concreact function ✗ { normal fun. with def. }
- abstract function ✓ { without deff }
- static function ✓ with definition { can call without object with class name. }
- default

normal class –
- normal function / concreat function
- static function
- constructor

object ✓ reference variable ✓

abstract class –
- abstract function
- concreat function / normal fun
- static function
- Constructor

reference ✓ object ✗

Interface – before 8.0 version

only contains abstract function, without abstract keyword

after 8.0 version

reference ✓ object ✗

interface –
- abstract function
- static function
- default function

---

normal class

```
class A {

  A()
  { sop("constructor A"); }

 void f1()
  { sop(" normal fun. f1 in A");}

static void f2()
  { sop(" static f2 in A "); }
}
```

```
class B  extends A
  {
    B()
   { sop("constructor B"); }

   void f3()
   { sop("normal fun f3 in B");}

  static void f4()
  { sop("normal fun. f4 in B"); }

  P.S.V.m(String[] args)
  {
    B b1 = new B();
    b1.func.name)
        → सारे fun. call हो जायेंगे
          → normal ✓
          → static ✓
  } }
```

abstract class -

```
abstract class A {

  A()
  { sop ("constructor in A"); }

  void f1()
  { sop ("normal fun in A"); }

  static f2()
  { sop ("static fun. in A"); }

  abstract ~~f3~~ void f3();
}

class B extends A
{
   void f3()
   { sop ("overriden fun."); }

   B()
   { sop ("constructor in B"); }

   void f4()
   { sop ("normal fun. in B"); }

static void f5()
   { sop ("static fun. in B"); }

   p.s.v.m. (---)
   { // object & reference
```

child class object & reference variable

```
B b1 = new B();
```

For constructor calling we need only

object ⇒ `new B();`

because of constructor overriding in constructor B bydefault 1st line will be `super();` (implicitly).

So पहले parent class का constructor call होगा।

output will be ⇒ Constructor in A
Constructor in B

by using refrence variable of B b1

we can call all the fun. that have inherited by B and the the that are of B itself.

b1 can call → all the functions
f1, f2, f3, f4, f5
normal, static, overide ✓

when `A a1 = new B();`

- `A a1` → refrence variable of A
- `new B()` → object of B → तो constructor call हो जाएगा। दोनो।

a1

- A के fun. को call कर पाएगा, जो B के area में आ गए हैं।
- B ने जो A के fun. overide किए हैं, उन्हे भी call करेगा।
- B के जो खुद के fun हैं, उन्हे नहीं करेगा।

f1 ✓, f2 ✓, f3 ✓, f4 ✗, f5 ✗

abstract class	interface
object नहीं बनता ।	object नहीं बनता ।
पर constructor बन सकता है ।	Constructor नहीं बनता है ।
functions - (4) abstract, normal Concreat, constructor	functions - (3) abstract, static, default.
Data - (3) ✓ local variables ✓ instance variables ✓ static variables	Data - (1) static variables ↳ and this variables are bydefault final.
reference variable बन सकता है abstract class के reference variable और उसे जिस क्लास ने inherite किया है उसके object से । abstract class के बारे function call हो जाएगा - पीछे लिखा है 😊	reference variable बना सकते हैं ।
	instance variable नहीं होते हैं । 8.0 के पहले - local possible नहीं थे । 8.0 के बाद - static के अंदर local हो सकते हैं ।

```
interface A
{
  void f1();
  void f2();
  static void f3()
  {
    S.o.p("static in A interface");
  }
  // constructor बनता नहीं।
  // default ~~value~~ fun. बना सकते हैं।
}

class B implements A
{
  void f1()
  { sop("f1 overriden"); }

  void f2()
  { sop("f2 overriden"); }

  B()
  { sop("Constructor of B class"); }

  static void f4()
  { sop("static ~~of~~ B class"); }

p.s.v.m.(String[] args)
{
  // object & reference
```

new B();

→ इससे B class का constructor direct call हो जाएगा। और interface का तो constructor बनता ही नहीं। reason आगे →

B b1 = new B();

reference variable of B

इससे $f_1()$; ✓

$f_2()$; ✓

static $f_3()$; ✗

static $f_4()$; ✓

static void $f_3()$; क्यों नहीं call होगा

~~क्योंकि~~ static void $f_3()$; interface का function है और इसके static function ~~किसी~~ भी class के द्वारा inherite नहीं ~~किए~~ जा सकते हैं।

तो ये fun. inherite नहीं हुआ मतलब वो child class के area में आया ही नहीं तो वो call होगा ही नहीं, उसके reference से।

A a1 = new B();

→ Paren class के मतलब interface के सारे fun. को access कर सकता है।

पर यहां B के area में तो static fun3() आया ही नहीं है। B से realate ही नहीं कर रहा और object तो हमारा B का ही है तो, a_1 से भी नहीं call होगा।

अगर करना है तो A का ही object भी बनाओ, जो हम बना नहीं सकते।

static void f4() भी नहीं होगा

क्योंकि a_1 केवल parent methods or जो child में overridden है उन्हे call कर सकता है।

~~static void f4()~~

```
interface A
{  int a;
```

not possible

तो आपको इन्हे declare करते time ही value देना पड़ेगा ।

क्यों interface के अंदर सिर्फ static variables होते है और वो भी bydefault final होते हैं ।

static variable को final इसलिए बनाते है, क्यों default constructor उसे value देने नही, नही आता। क्यों interface के अंदर constructor होता ही नही ।

interface के अंदर instance variable होते नहीं है ,

Constructor का काम होता है instance variable को value देना । पर वो तो है ही नहीं ।

और static को value देना, तो Static variable को तो final बना दिया है ।

तो Constructor बनता इसलिए ही नही है ।

Constructor नहीं बनता है, interface का ।

direct
strong
class to class ~~loose~~ coupling

class to interface loose coupling
indirect