



PEPCODING

PURSUIT OF EXCELLENCE AND PEACE

MODULE
OOPS



+91 11 4019 4461



PepCoding, 3rd Floor, 15 Vaishali,
Pitampura Opposite Metro Pillar 347,
Above Karur Vysya Bank, New Delhi,
Delhi 110034



www.pepcoding.com



/pepcoding

OOPS

Encapsulation: Wrapping of data under single unit.

classes objects
(data, functions)

(Mechanism that binds together code and the data it manipulates)

⇒ JAVA

person p1 = new person();

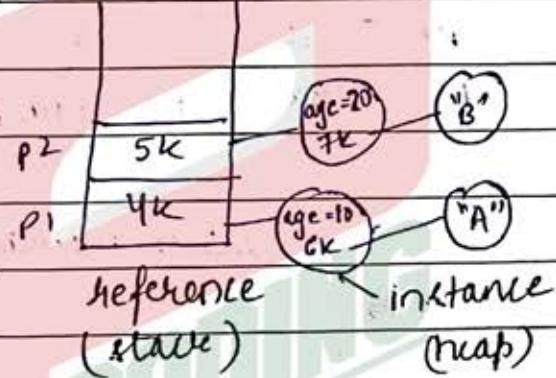
p1.age = 10

p1.name = "A"

person p2 = new person();

p2.age = 20

p2.name = "B"



Ques: Predict the changes when swapped like -

a) swap (person one, person two)

{ person temp = one

one = two

two = temp }



As per the stack diagram,
all the changes are limited to
stack only.
Hence, no change ⇒ NOT SWAPPED!

b) swap (person one, person two)

{ string tn = one.name

one.name = two.name

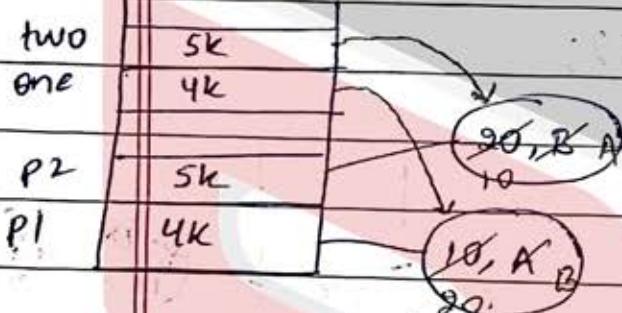
two.name = tn

int tage = one.age

one.age = two.age

two.age = tage

}



(Swapping on values)

SWAPPED! because changes
are made directly on
addresses of objects.

Since memory is accessed → hence permanent
changes.

c) swap (person one, person two)

{ one = new person();

int tage = one.age

one.age = two.age

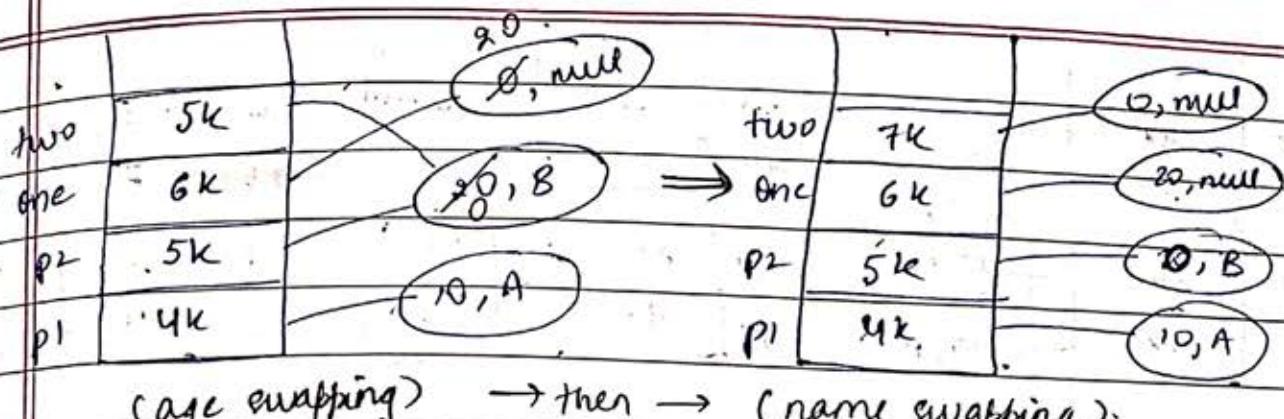
two.age = tage

two = new person();

string tnname = one.name

one.name = two.name

two.name = tnname



Output : $p1 = 10, "A"$

$p2 = 0, "B"$

Only age of $p2$ changes to 0 as ~~ptr~~ "one=new person()" made 'one' to point to some new address 6K, hence original $p2$'s age is swapped with new 'one' object.

Hence, no visible changes after "two = new person()".

d) swap (person one, person two)

```
{
    int tage = one.age
    one.age = two.age
    two.age = tage
}
```

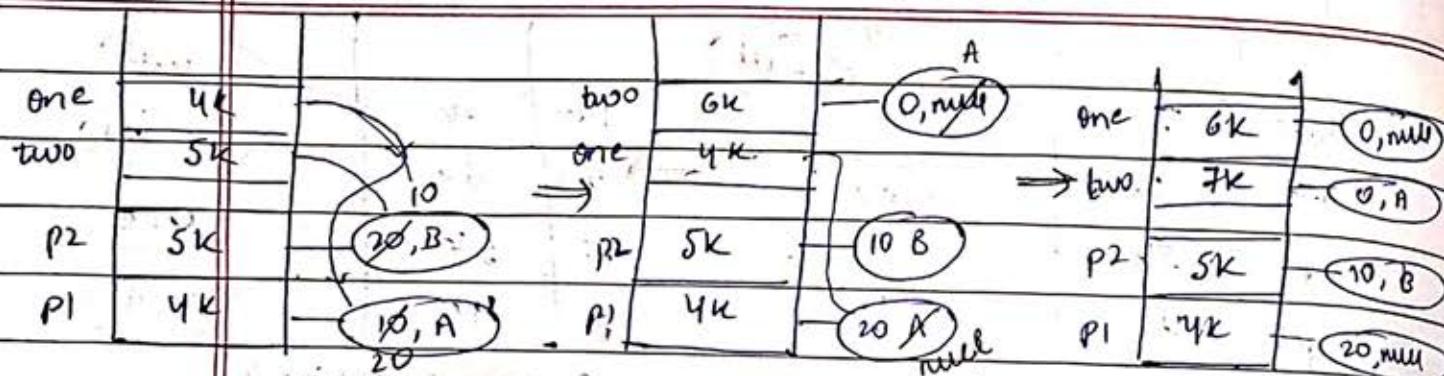
$two = \text{new person}()$

~~string~~ tname = one.name

one.name = two.name

two.name = tname

one = new person() }



age swapped

name swapped

After dropping down, the memory of `one` and `two` will release and GARBAGE COLLECTOR will swipe out heap storage of '`one`' and '`two`'.

At last we will be left with -

$$p1 = 20, \text{null}$$

$$p2 = 10, B$$

(as per
stack
diagram)

e) swap (person one, person two)

$$\{ \quad \text{int} \text{ age} = \text{one}. \text{age}$$

$$\text{one}. \text{age} = \text{two}. \text{age}$$

$$\text{two}. \text{age} = \text{age}$$

$$\text{one} = \text{new person}()$$

$$\text{two} = \text{new person}()$$

$$\text{String tname} = \text{one}. \text{name}$$

$$\text{one}. \text{name} = \text{two}. \text{name}$$

$$\text{two}. \text{name} = \text{tname}$$

Only age will get swapped

$\rightarrow p1 = 20 \ A$

$p2 = 10 \ B$

C++ vs. Java

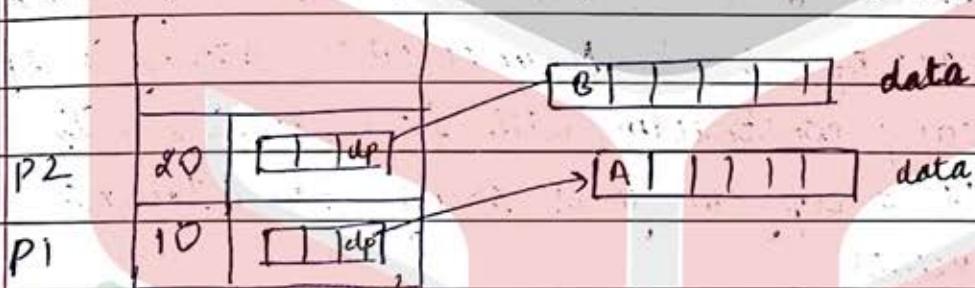
person p1; person p2;

p1.age = 10;

p1.name = "A";

p2.age = 20;

p2.name = "B";



string = size, capacity, data pointer

In C++, objects are made directly on stack.
(NO concept of reference - instance)

Ques. Predict the changes when swapped like! -

1. swap (person one, person two)

 |
 | person temp = one

 | one = two

 | two = temp

y

-emp	10	A
tur	20	B
one	10	B
b2	20	"B"
p1	10	"A"

changes made on stack
but are not permanent
as when function is
completed, its effect
disappears.

Hence, NOT SWAPPED!

But this kind of swapping is dangerous. Why?

- If a string / vector / object is passed without reference in C++, then it is $O(n)$ operation in place of $O(1)$ because when string is copied, its copy constructor will fire which will create a copy of your string at different addresses.

e.g.: person p1;

$p2 = p1;$

p2	10	5K	A
p1	10	4K	A

So, HEAVY STRINGS CAN MAKE THIS SLOW

- Consider the following case:-

person p1;

$p1.age = 10$

$p1.name = "A"$

if (true)

```
{ person p2 = p1;
  delete;
```

y

p2	10	4k
p1	10	4k

A

4k

If copy constructor wouldn't be working the way it works, i.e. 2 objects pointing to same address, then it could cause the problem of DANGLING POINTER.

which means when we drop down and scope of 'if' ends, destructor will be fired, deleting all introductions of 'if' \Rightarrow p2 will be deleted, leaving behind a dangling pointer for p1.

p1	10/4k	?
----	-------	---

\Rightarrow RULE OF 3 \rightarrow constructor operator =

\downarrow \rightarrow copy
destructor constructor

while writing the destructor in C++, you must write copy constructor and operator =, otherwise it will cause above mentioned problem.

2. swap (person &p1, person &p2)
 $\{$ //swap and p1, p2 are hidden pointers
 $\}$ ($\& \Rightarrow$ Hidden pointer)

temp	10	A
two	1004	
one	1000	
p2	20	B
p1	10	A

(Ans)

Since, true time
address.) reference
was received,
hence SWAPPED!

- Difference b/w copy constructor and assignment operator?

Person p1;

Person p2;

Person p3 = p2; ↗ copy constructor

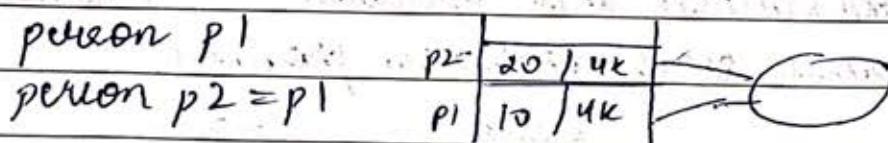
p3 = p1; ↗ assignment operator

- If already exists - assignment operator
(after clearing previous memory)

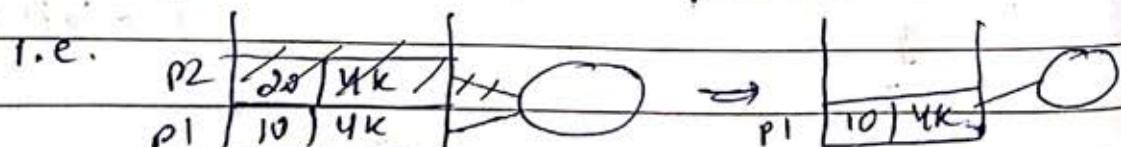
- If new allocation - simply copy - copy constructor

- ⇒ What if no copy constructor & destructor is written?

In this case, default copy constructor and default destructor will be used.

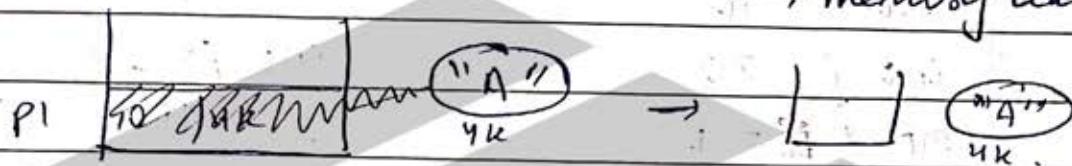


But nature of default destructor ⇒ it only removes pointer and itself but not material in heap.



But problem of MEMORY LEAK occurs when,
if there arises a case where p1 vanishes
(for example, its initialised is 'if', and scope
of 'if' ends) then, p1 too will disappear
leaving the material of heap intact

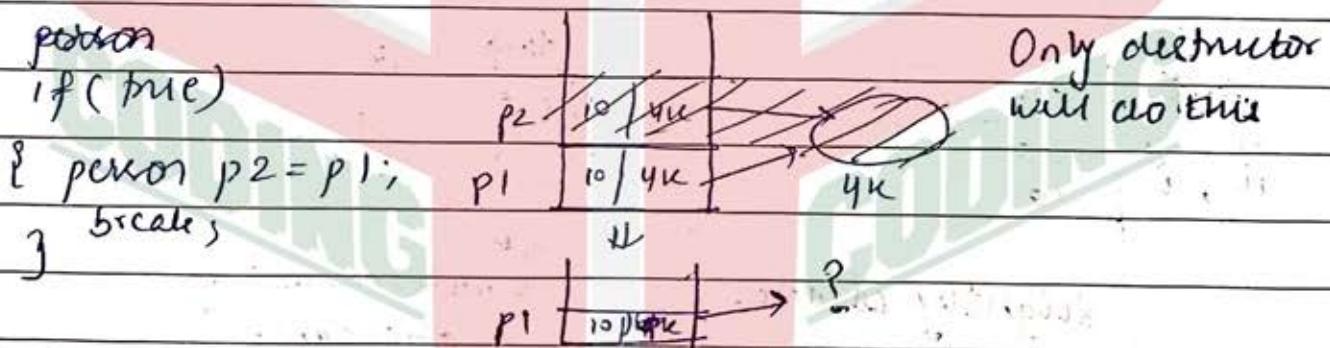
→ memory leak.



So we need to write our own ~~destructor~~

↳ but remember to write copy constructor and assignment operator along with it otherwise problem of DANGLING POINTER will arise.

person p1 ≠



* SO WE NEED DESTRUCTOR and OTHER TWO WHEREVER WE HAVE ~~ONE~~ POINTER TO HEAP OR ALLOCATION OF MEMORY ON HEAP

eg: whenever 'new' is used..

QUESTION what if we use all the 3 properly?
then performance will be bad as all copy operations will be of cost O(n) in place of O(1),

So, USE CALL BY REFERENCE in place of CALL BY VALUE for better performance.

Difference between "*" and "&"

REFERENCE

```
int i = 10
int j = 20
int & ii = i
int & jj = j
int temp = ii
ii = jj
jj = temp
```

POINTERS

```
int i = 10
int j = 20
int * ii = & i
int * jj = & j
int temp = * ii
* ii = * jj
* jj = temp;
```

temp	32
jj, j	1K
ii, i	20
	10

temp	1K
jj, j	20
ii, i	10

slightly cleaner

writes exactly
the same

"DUMMIES" /

"DUMMY POINTER"

example: fun (int &ii, int &jj)

(reference
version)

```
int t = ii;
ii = jj;
jj = t;
```

t	10	10
jj	2000	2000
ii	1000	1000

It is
actually
* and /

main () {

int t = 10

int j = 20

fun (i, j) }

i	20	20
j	1000	1000

→ Pointer version

```
fun( int *a, int *b ) { }
```

```
int temp = *a;
```

```
*a = *b;
```

```
*b = temp;
```

```
main( ) { }
```

```
i = 10
```

```
j = 20
```

```
fun( &i, &j );
```

{	temp	10.	}
	b*	2000	

{	j	20 10	}
	i	10 20.	

2000

1000

2000

1000

Both works exactly the same way, its just that '&' version is slightly easy to understand.

&a → prints address of 'a'

Constant Pointers

Reference

vs

Pointers

- 1) Cannot re-point them
- 2) Cannot be null
- 3) Needs to be initialised at declaration
- 4) Usage - convenient

{ primitive → normal }
Other → .

- 1) Can re-point them
- 2) Can be null
- 3) Can be initialised later
- 4) Usage - tricky

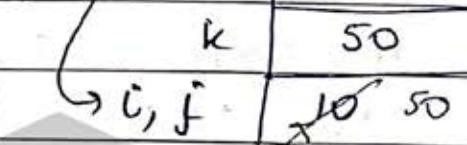
{ primitive → * }
Other → '→' arrow

ex → int j = 10 ;

int k = 50 ;

int & l = j ;

l = k ;



Important: ~~so~~ 'i' just becomes another name for 'j'

NOTE :

$l = 10$	$*m$	3000	
$j = 20$	$*k$	1000	3000
$\text{int } *k = \&j$	j	20	2000
$\text{int } \&l = i$	l, i	10	1000

l becomes another name of i

1) $*k \rightarrow$ prints value at address of k = 10

2) ~~print~~ $\&k \rightarrow$ prints address of k = 3000

3) $\text{cout} \ll *l \Rightarrow \text{WRONG } X$

4) $\text{cout} \ll \&l \Rightarrow 1000.$

5) $\text{cout} \ll l++ \Rightarrow 11$

6) $l = j \Rightarrow l \text{ become } 20$

7) $\text{cout} \ll m \Rightarrow 10$

Segmentation fault → pointing to a memory

which do not hold anything.

$\&k \Rightarrow$ address of k

$*k \Rightarrow$ value at address of k

NOTE: $(\&i)++$ is not allowed because

its equivalent to incrementing an address

of $1000 \rightarrow 1001$

$$\& i = &i + 1$$

↳ error: expression must be a modifiable value.

→ Concept of Repointing ?

Reference:

A reference variable cannot be reassigned.

eg. $\text{int } n = 5; \text{ int } y = 6;$
 $\text{int } &x = n;$

Pointers:

This property comes in handy when a developer is implemented data structure like linked lists, trees etc..

eg. $\text{int } n = 5, \text{ int } y = 6;$
 $\text{int } *p; p = &x; p = &y;$

→ Swapping operation ways Is Swap possible?

1) swap (person* one, person* two)

{ person t = one *

one * = two *

-two * = t }

Swap possible; working → just like reference version

2) swap (person one, person two)

{ // swapping age

// swapping name separately

t	10	9
two	20 A B	
one	10 A B	
	2	

p2	20 B	9
p1	10 A	

NO SWAPPING POSSIBLE,
since all changes only on stack

SUMMARY

Edition

⇒ ~~Java~~

Java programming

Java

INHERITANCE

parent class

```
public class parent {
    int d1 = 10;
    int d2 = 100;
```

void fun1()

{ print ("parent fun1") }

void fun()

{ print ("parent fun") }

child class

public class child

{ int d2 = 20; }

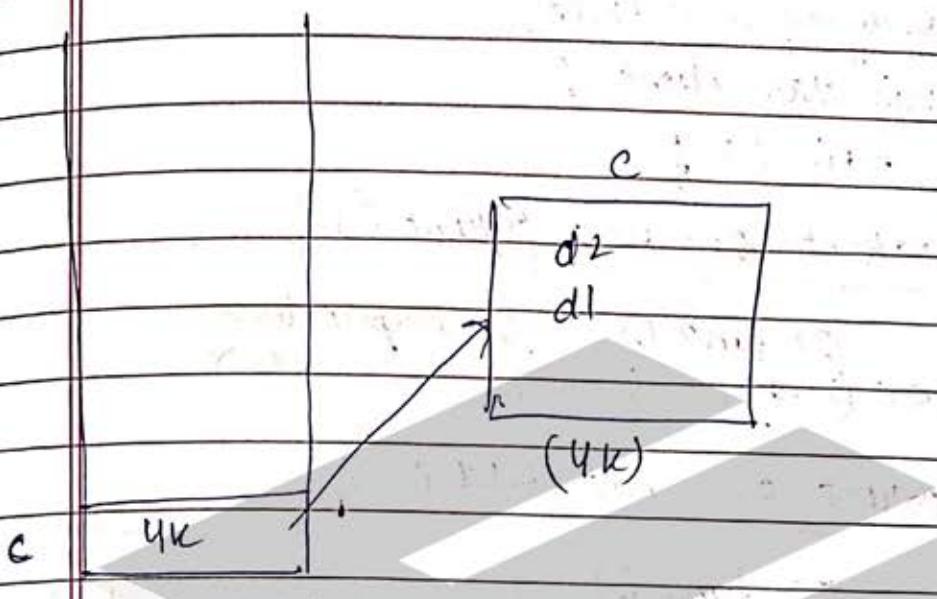
int d = 200;

void fun2() { print ("child fun2") }

void fun() { print ("child fun") }

RULES ... (IMPORTANT)

1. Editing → compile time → Compiler → LHS or reference
2. Executing → Runtime → JVM → RHS or instance
3. Data members are always resolved on LHS

Parent $\Rightarrow 20k$ $d1 = 10$ $d = 100$ $f1$ f  $d2 = 20$ $d = 200$ $f2$ f child $\Rightarrow 10k$ \rightarrow public class parent {

child c = new child(); // 4K

print(c.d1)

parent c.fun1();

} compile time
error

y

 \rightarrow Now applying inheritance,

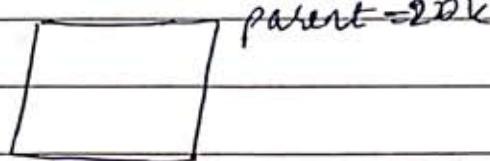
public class child extends parent {

int d2

int d

y

thus makes,

child believe that
it is derived
from 20k (parent)Child $\Rightarrow 10k$

After applying inheritance,
public class client {
 main () {

1) parent o1 = new parent();

o1.fun2(); } } compile time
point (o1.d2); error

2) parent o2 = new child();

→ o2.fun2(); } } compile time
point (o2.d2); error

→ o2.fun(); → child's fun()
(// runtime decision based on instance
and existence → child)

or

(// fun() found at child, then why go up?)

→ o2.fun1(); → parent's fun1()

→ point (o2.d) → parent's as data
members are

point (o2.d1);

always
available on RHS

↓
parent's as d1 present only is
parent

IMPORTANT NOTE : "new child ()" is entirely different event
which includes combining all functions and members and
giving a feel of being derived from child BUT "parent" ~~is~~ ^(reference) (LHS)
will decide which functions / members can the object use.

3) child : 03 = new parent();

03.fun();

03.fun1();

03.fun2();

print(03.d)

print(03.d1)

print(03.d2)

compile

time

error

Because child can never be reference to its base.
Had it been allowed,

03

d10
d1100

not
able to
find d2, p2()

BASICALLY, Child class's reference can access more elements/members which are not present in parent/base class instance:

4) child : 04 = new child();

04.fun() → child // because rule says instance

04.fun1() → parent // obviously

04.fun2() → child

print(04.d) → child // rule says instance

print(04.d1) → parent // obviously

print(04.d2) → child

Is there any way to access data members and functions of child in the combination
 parent o2 = new child();

typecast o2 in child

↳ but this will alter the functionality/
 or rule slightly.

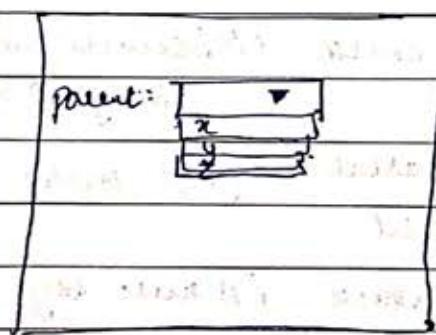
NOW o2. d will of child rather
 than parent.

- * There is no way to use values of common functions of parent while using combination -
 child D = new parent();

RUNTIME POLYMORPHISM.

If there are a number of instances available as option to a particular reference of an object, then runtime whichever instance is chosen, those functionalities can be used.

eg: drop down option application.



Parent p = null;

If x chosen

p = new parentx();

If y chosen

p = new parenty();

If z chosen

p = new parentz();

~~dynamic dispatch~~

Runtime Polymorphism

~~dynamic polymorphism~~~~dynamic binding~~

late binding

method

overriding

or
polymorphism

instance is

not known before

and is known only

at the runtime

based on the instance
that has been chosen.same name,
different functionality

#

COMPILE TIME POLYMORPHISM

which instance to be chosen — decided by COMPILER.

METHOD OVERLOADING — at the time of compiling

eg: `fun (int a, int t)``return a*t;`eg: `fun (double d, double t)`
`return d*t;`why developers have provided this feature of
keeping the name same for different input / data types?

- To keep the API's compact

- to provide ease of use and coding to users
and club all sorts of "sort" together user
instead of sortbool, sortint etc.

NOTE: Method overloading not to be done just on the basis of Return type as its not mandatory to receive values at the time of calling.

Invalid method overloading -

int fun (int , int)

String fun (int , int) X

their argument type must be different

↓ ↓
number
order type

OPERATOR OVERLOADING; Same operator, different meanings .

not allowed in Java,
but still ,

String s = "Hello"

s = s + "Hello" ;

'+' is used for concatenation . How ?

Because java uses '.append' wherever '+' is used.

IMPORTANT : { non-virtual, data members - difference
virtual - instance
classmate

Date _____

Page _____

→ C++ :

Polymorphism works only when base functions are VIRTUAL. (By default Java - all are virtual)

If these functions are not marked virtual, then:

EARLY BINDING will take place i.e.

reference will return base reference's function and will not wait for run-time (instance).

1. Parent * p1 = new Parent();

p1. d → parent

p1. d1 → parent

p1. d2 → X

p1. f1 → parent

p1. f2 → X

2. Parent * p2 = new Child();

p2. d1 → parent

.. p2. d2 = X, if virtual → child

02. f1 = ← if non-virtual → parent

02. f2 = X

02. f1 = parent

02. d = parent

3. Child * p3 = new Child();

03. d1 - parent

03. d - child

03. d2 - child

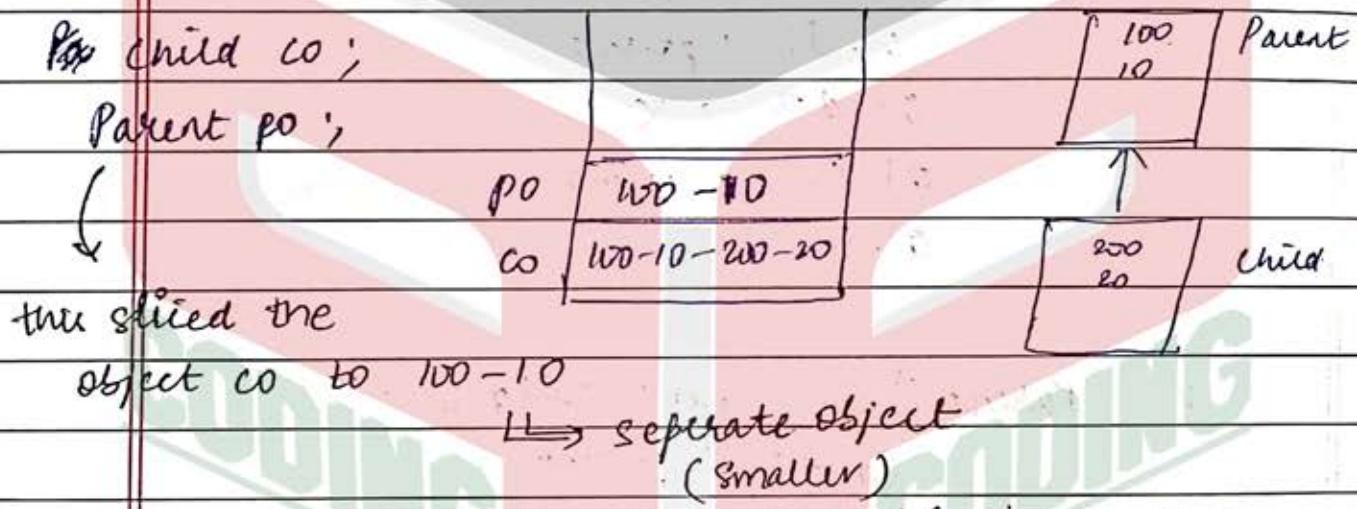
03. f1 - child

03. f2 - parent

03. f1 - child

C++ rules summary

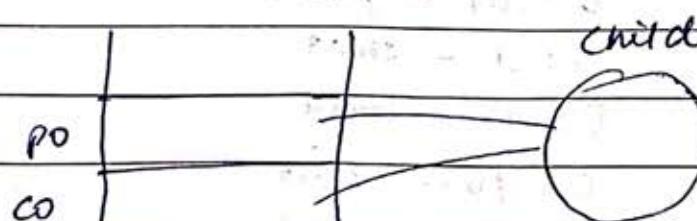
- 1) During editing \rightarrow reference
- 2) During executing
 - Non virtual ; data members \rightarrow reference
 - Virtual functions \rightarrow instance
- 3) Object slicing
 - \hookrightarrow when pointers not used



Now when 'd' is pointed or 'fun' is called, parent's functions / members will be called.

WHEREAS

Earlier when handling through pointers both po and co pointed towards the same instance and no separate objects were created.



Here, p0 cannot access d2 and f2 but would give func f of child.

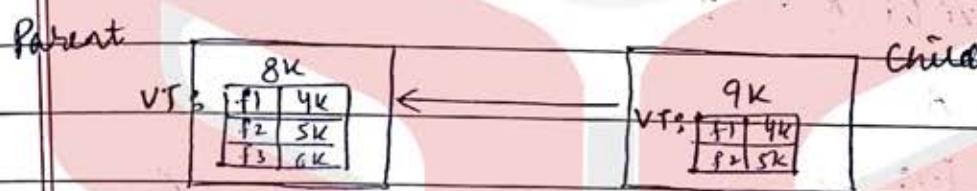
Virtual Pointers and Virtual table:

If f1, f2, f3 be virtual functions of class
 (4K) (5K) (6K)
 parent and

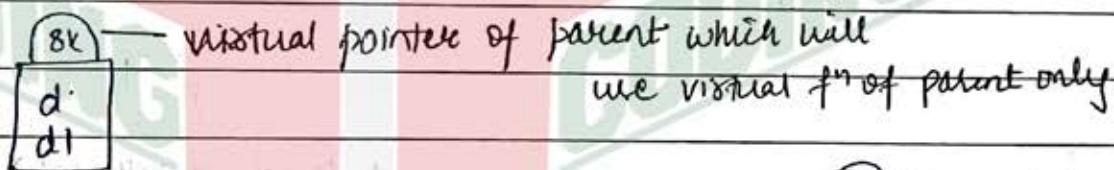
f1, f2 be virtual functions of class child
 (4K) (5K)

respective.

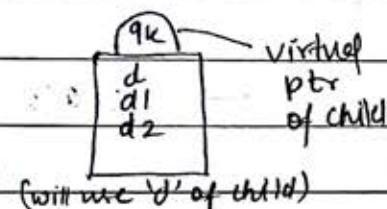
then these functions reside in VIRTUAL FUNCTION TABLE of respective classes and pointer to this virtual table is known as VIRTUAL POINTER.



when, Parent p0 = new Parent()



likewise, for Parent p0 = new Child()



en → class A {

(here sizeOf(A)
means size
of
an object
of class A)

int d1

func() {

} } ; ;

Is sizeOf(A) < sizeOf(B) ?

class B {

int d2

virtual func() {

} } ; ;

YES, because object of B has hidden virtual pointer
and hence its size is greater.

NOTE: Virtual pointer is called with the CONSTRUCTOR i.e. reference to virtual table is made when constructor of virtual functions is constructed.

- * without pointers, run-time polymorphism is not possible.

eg: → class A {

 virtual fun() { }

}

$$\text{sizeOf}(A) = 8$$

→ class A

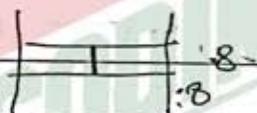
 { int a;

 virtual fun() { }

}

$$\text{sizeOf}(A) = 16$$

due to padding



4 used, 4 unused.

→ class A

 { int a;

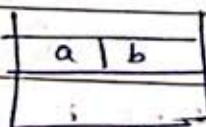
 int b;

 virtual fun() { }

}

$$\text{sizeOf}(A) = 16$$

4-4
used



1) by, left one more 'int' introduced / initialised
 $\text{sizeOf}(A) = 24$

Static data members

number → variable

data member

STATIC keyword → function

i) STATIC VARIABLE : (Only in C++)

↳ initialised and used within a function

eg: func()

```
{ int i = 10;
  static int j = 20;
  i++;
  j++;
  cout << i;
  cout << j;
```

y

main()

```
{ func();
  func();
  func(); }
```

Output: 11 21

11 22

11 23.

Each time, i is initialised by 10 but j is only initialised once, and gets incremented in next calls

because static variables are stored in DATA SEGMENT - separate space in heap.

life scope of static variable = life of program
 scope of static variable = function

GLOBAL VARIABLES

→ scope = entire program

→ life = life of program

⇒ A static variable is used as an instance variable

2) STATIC DATA MEMBER :

Ex: class Bank (Memory sharing)

{ int accountno;

int money;

static double roi;

// static data member

}

Bank bank1* = new Bank () { int acc.no; int money }

bank2* = new Bank ()

bank3* = new Bank ()

Each having its own set of value of account no and money but rate of interest for all banks will remain same.

If roi of a single bank changes, roi for all banks will automatically change.

This is an example of MEMORY SHARING.

Basically, "This memory is stored in class and not in general memory space" - Static ^{data member} ~~variable~~ (only in C++).

- 3) STATIC FUNCTIONS: These functions can be called directly on class without creating objects.
- * STATE OF AN OBJECT: Snapshot of data members inside it
- * STATELESS FUNCTIONS: Functions which ~~are~~ it can be used independently from and does not depend upon previous usage.
Ex - Transaction on a shop etc, Math.pow()
- * STATEFUL FUNCTIONS: Functions whose usage depends on previous usage / transaction.
Ex - Bank transactions
↳ depends on previous balance and so on.
- Can static function use 'this' pointer inside it?
No, because since no objects are created for the access of static functions, there is no concept of "this" pointer here.
- Can static function use a non-static data member?
No, because we need an object for calling and using non-static data members and no objects are created for static functions.

- Can a non-static data function use a static function / data?

Yes, using non-static functions means object is created but for usage of static functions, no even this object is required; static functions can be used anyway.

4) STATIC CLASS: Just like static functions can only access static data members, static class can be accessed through static functions without creating object of outer class.

ex → linked list class

static class Node

```
{ int data; Node next }
```

static main()

```
{ }
```

y

'main' can only use 'Node' class only if it is marked static.

IMP: * Objects of static class can be easily made

```
Node* node = new Node();
```

Points Only point to be noted is -

class linkedlist

```
{ Node class {
```

int data;

Node next;

}

→ If Node class is static
object of node can be made directly.

→ If Node class is non-static
To access Node, we need to first create object
of linkedlist class and then access Node
through that object.

(linkedlist object)

linkedlist ll = new linkedlist();

~~Node node = ll.createNode();~~

(node object)

linkedlist.Node node = ll.new ~~createNode~~ Node();

Basically,

FINAL Keyword

⇒ JAVA

final → variable, ~~data~~
 ↓ ↓
 / data member
 class function

1) VARIABLE: We cannot change value of these variables

2) DATA MEMBER: same as variable but have rules
for initialisation.

class A {

final int a;

A()

It needs to be initialised
either at declaration
or in constructor
else ERROR!

3) FUNCTION: final functions are opposite of virtual functions.

You cannot OVERRIDE final functions once written i.e. you cannot even write the body of a function which was marked FINAL which was otherwise VIRTUAL by default in JAVA.

class Parent

```
{ int d;
  final func() { ... }
```

class child

```
{ int d,
```

// here func() cannot be written

}

4) CLASS : Final classes cannot be inherited

e.g. also final class P

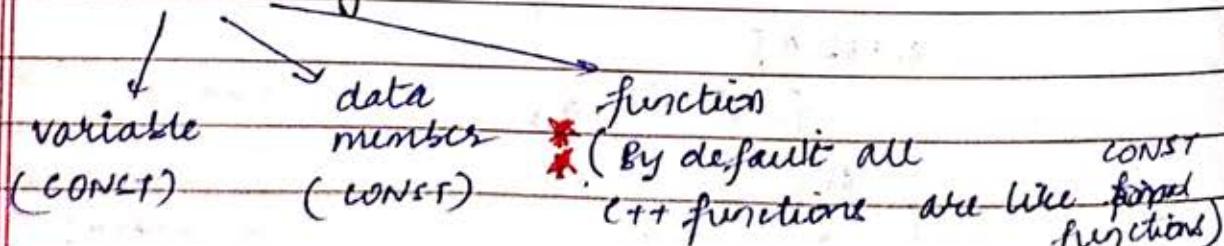
{

}

then, C extends P \Rightarrow not allowed

\Rightarrow C++

CONST Keyword



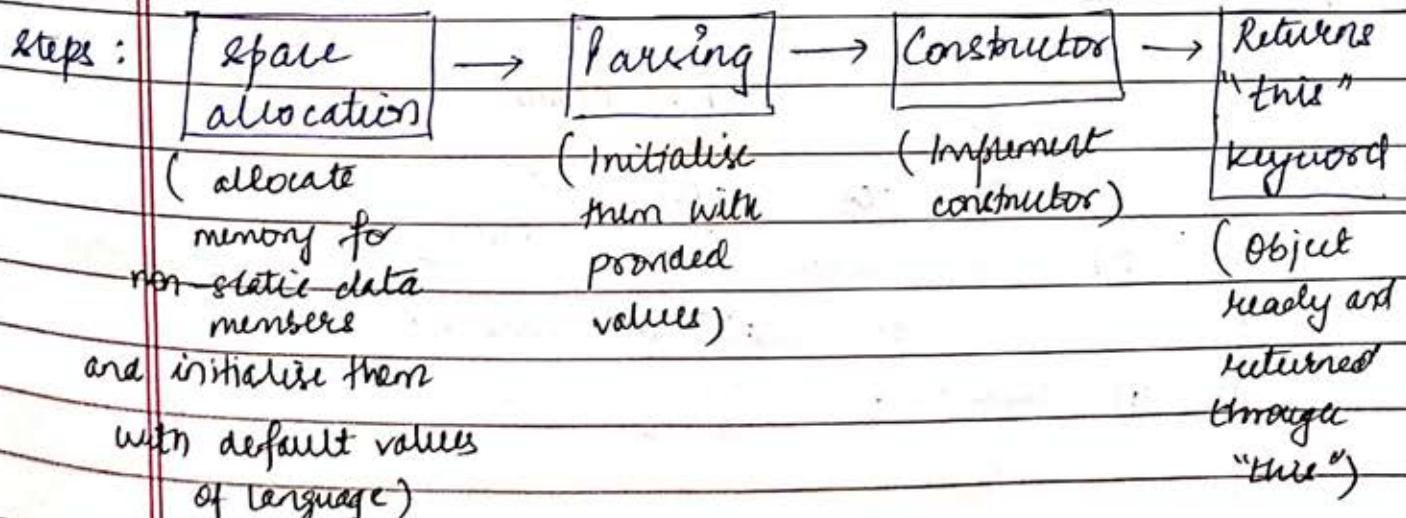
- * There is nothing available in C++ to achieve achieve the impact of FINAL class

	Java	C++
1) Variable	Final	CONST
2) Data member	FINAL Initialization → parsing constructor	CONST Initialization → only parsing
3) FUNCTION	FINAL (can't override)	By default C++ functions
4) Class	FINAL (can't inherit)	- NOT available -

Example of final classes in JAVA : ?
String, Double , int etc.

OBJECT CREATION

Person p = new Person()



class → public class account {
 int amount = 10;
 int ano = 3;
 static double ROI = 0.04;
 static int count = 0;

account ()

```
} ... count++;  
ano = count; } }
```

account (int amount)

```
{ ... this ();  
tha.amount = amount; }
```

Steps of Object creation

Step 1: Space allocated for amount and ano
and initialised with defaults

```
amount = 0  
ano = 0
```

NOTE :

- 1) Java defaults → space allocation.
- 2) class creator → Fixed defaults Parsing
~~class~~ eg: amount = 10.
- 3) class creator → Dynamic Defaults Default constructor
eg: static int count ++;
- 4) class user → previous values Parametrised constructor

ABSTRACT keyword.

functions class

JAVA
⇒ Abstract

C++

Step 2: Initialising with fixed defaults - Paring

amount = 10
ano = 0

Step 3:
Step 200.

Dynamic defaults - default constructor

amount = 10
ano = 1

Prunous values - parametrised values

amount = 10 + x
ano = 1

Note: Default constructor within parametrised constructor
account (int amt)
 { this();
 this.amount += amt }

over in class A

print pair

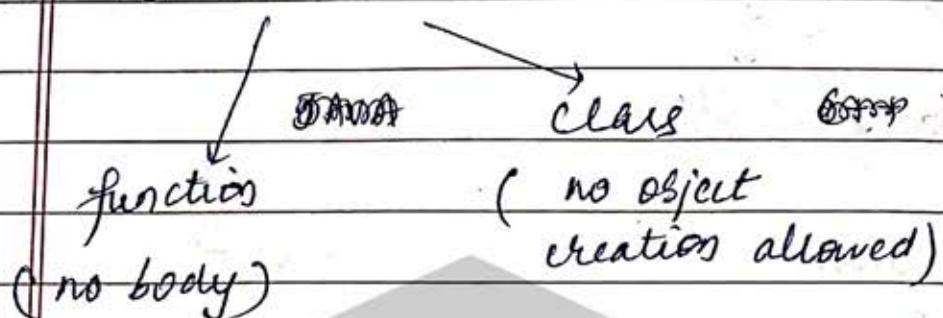
Account A(p)

Expresso

version before

Expresso

ABSTRACT KEYWORD



+ If a function is abstract then it is necessary for its class to be abstract?

Yes, because abstract function means function has no body and non-abstract class means its class can be created. Now calling an empty function via through an object would make no sense - null pointer.

- It is necessary for an abstract class to have atleast one abstract function?

No, an abstract class may have all non-abstract functions. It hardly matters.

NOTE: If an abstract class is inherited by some non-abstract class, ~~without providing definition for~~ and all the functions of abstract class are abstract then, —

either you define all the functions of abstract class or make the derived class as abstract too. and this will continue unless definition for all functions ~~are~~ provided.

→ Difference between library and framework.

Library: Called and used by application only.

Framework: Cross calls - framework can call and use application as well.

Eg - Priority queue is a framework which calls our app using comparator.

- Where do we use abstract class?

TEMPLATE DESIGN PATTERN - A parent template class is created with abstract functions which might ^{contain} have different logics for different applications.

→ Basically, it provides empty body / skeleton of an operation in terms of a number of high-level steps and leaves the details to be implemented by child class.

The overall structure and sequence of the algorithm is preserved by parent class.

Template - Preet format

- Can abstract class have constructor?

Yes, in the scenario of inheritance, to initialise the properties which may get derived down the chain.

→ abstract class A {

p1

} abstract fun() { }

abstract class B {

g

p2:

abstract class C {

g

p3
fun() { }

In the given scenario, if body of `func()` is defined in class C where

A

B

T

C

then we need to have constructor in abstract classes A and B to initialize p1 and p2.

Constructor will be called in the sequence —

class A →

A (p1) {

 this.p1 = p1;

class B → B (p1, p2)

{ super (p1)

 this.p2 = p2;

// super means
constructor of
class just above it.

class C →

C (p1, p2, p3)

{ super (p1, p2)

 this.p3 = p3;

// if parametrized
constructor super(p1, p2)

is not called,
then its default
constructor will be
fixed automatically

NOTE: `super(p1, p2)` ≠ `this.p3 = p3`

`this.p3 = p3` `super(p1, p2)` } THIS IS WRONG!

Constructor chaining : In an inheritance scheme of type → A

↓

B

↓

C

constructor will be fired in sequence ⇒ A → B → C
whereas destructor sequence ⇒ C → B → A

Abstract keyword → Abstract (Java)

→ Pure virtual (C++)

⇒ Example of constructor chaining -

```
class A {
    A()
    {
        cout << "A" << endl;
    }
}
```

OUTPUT ⇒ A

```
class B {
    B()
    {
        cout << "B" << endl;
    }
}
```

B

```
class C {
    C()
    {
        cout << "C" << endl;
    }
}
```

A

```
class D {
    D()
    {
        cout << "D" << endl;
    }
}
```

C

```
main()
{
```

```
    D d = new D();
}
```

A

B

A

B

A

C

A

B

A

PEPCODING - PURSUIT OF EXCELLENCE AND PEACE

ITERATOR

Consider the question of finding pair sum equal to 100 in binary tree.

Approach	BST	Binary	Space
1) Recursion in recursion ('find' function)	$n \log n$	n^2	$O(h)$
2) sorted arraylist	n (Inorder of BSF)	$n \log n$	$O(n)$
3) Hashmap	$\Theta(n)$	n	$O(n)$
4) Iterator (recursion from both side - "freezable")	n	n	$O(h)$

INTERFACE : Pure abstract class with all body-less functions and no data members.

Interface \rightarrow implements

Abstract class \rightarrow extends

NOTE:

- Interface may have data members but of. STATIC type.
- C++ does not have interfaces.

Use of interface in Java:

Since java doesn't support multi-inheritance, so to achieve that purpose/ functionality - interfaces can be used.

ITERATOR VS ITERABLE (JAVA)

Both of them are interfaces

→ Iterable - to be implemented by main class (BST)

↳ It has a single function " iterator() "

↓ returns
ITERATOR

(iterator is again an interface)
which provides composition by

Since we need to return an Iterator on the iterator() function, so we will be writing a BSTIterator class and it will have 2 functions of ITERATOR interface

hasNext() next()

So basically,

1) Our class implements Iterable < ? >
It's function → iterator() will return an

~~bstiterator (Node node)~~ Iterator named

~~stack.push(new Stack(nodes))~~ bstiterator

2) class bstiterator will have 2 functions

hasNext() next()

Algorithm for applying iterator on BST

USING STATES - For inorder traversal

state 0 → push in stack and

state 1 → push left

state 2 → point and push right

↓

state 3 → pop from stack

To implement the above, we can apply iterations on pair class,

class pair {

 Node node;

 int state;

 Pair (Node node, int state);

 { this.node = node;

 this.state = state;

 }

 }

For implementing iterator on BST,

For iterating BST, we need two iterators

bstiterator

(front
cursor)

reverseiterator

(reverse
cursor)

→ public Iterator < Integer > iterator ()
return new bstIterator (root);

static class bstIterator implements Iterator < Integer >
{ static class Pair {

Node node
int state; } }

Stack<Pair> st = new Stack ();
int val;

bstIterator (Node node) {
if (node != null) {
st.push (new Pair (node, 0));
next (); } }

public boolean hasNext ()
return val != -1;

public Integer next ()
{ int nv = val;
val = -1; }

while (stack.size > 0) {
Pair top = st.pop ();
if (top.state == 0)
 → add top.left if it exists
 and state ++

else if ($\text{state} == 1$):

$\rightarrow \text{state}++;$

$\text{val} = \text{top}.\text{node}.\text{data}$

BREAK;

else if ($\text{state} == 2$):

$\hookrightarrow \text{state}++$

add $\text{top}.\text{node}.\text{right}$ if it exists

else if ($\text{state} == 3$):

$\hookrightarrow \text{st.pop}();$

3

return $\text{arr};$

→ Similarly, in reverse iterator.

\hookrightarrow only difference \Rightarrow

at $\text{state} = 0$:

\hookrightarrow add height

at $\text{state} = 2$

\hookrightarrow add left

How to use these iterators for finding pair sum equal to x ?

Iterator <integer> $\text{ite} = \text{vec}.\text{iterator}();$

Iterator <integer> $\text{itr} = \text{tree}.\text{iterator}();$

```

int left = arr[0];
int right = arr[1];
while (left < right) {
    if (left + right == sum) {
        increment both and break;
    } else if (left + right < sum) {
        increment left;
    } else {
        right = arr[right];
    }
}

```

HASHCODE and EQUALS

Needed when our custom object is used as key in hashmap.

HashCode → which bucket to choose

equal → decides whether the given value is matching with any of the element present in that bucket or not.

Consider the problem of finding count of maximum no. of points lying on a straight line. —

class line {

 int dx;

 int dy;

}

Here class 'line' objects will be used as 'keys' ~~part of~~ of ~~in~~ hashmap.

Hence overriding `HASHCODE` and `EQUALS` functions -

⇒ `public boolean equals (Object o)`

```

    {
        Line lo = (Line) o;      ( typecasting
        if ( this.dx == 0 && o.dx == 0 )          of
        if ( this.dy == 0 && o.dy == 0 )
            return true;
        else if ( this.dx == 0 && o.dy == 0 )
            return true;
        else if ( this.dx == 0 && this.dy == 0 && this.dx == o.dy )
            return true;
        return false;
    }

```

⇒ `public int hashCode ()`

```

    {
        return this.dx + this.dy;
    }

```

TO STRING FUNCTION (`toString()`)

→ `toString()` method returns the string representation of the object.

It is overridden to print the desired output of custom class.

example:- class student {

int rollno;

string name;

string city;

student (int r, string n, string c)

{ this. rollno = r;

this. name = n;

this. city = c; }

y

~~main (void)~~

Now; if

Student s1 = new Student (*100, "ABC", "GDA");

and we want

"System.out.print (s1)" to print

rollno, name and city in sequence,

then we need to override toString function like -

public string toString ()

{ return rollno + " " + name + " " + city; }

Otherwise it will print the hashCode values of
these objects.