Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/

Character String (C++)

```
char *str="IITRoorkee";
    for (int i=0; i<10; i++)
    {     cout<<str+i;
          cout<<*str;
    }     cout << *(str+i);
```

str[0] = I
str[1] = I
str[2] = T
.
.
.

0	1	2	3	4	5	6	7	8	9
I	I	T	R	O	O	r	k	e	e

How to print the base address of string

```
(int *) str
&str
```

i=0
IITRoorkee
I

i=1
ITRoorkee
I

i=2
TRoorkee
T

i=3
Roorkee
R

```
int a[2] = { 1, 2 };
cout << a;
```

Strings in Java

- Char charArray[ ] = new char[4];

charArray[0]='J';

charArray[1]='A';

charArray[2]='V';

charArray[3]='A';

- String str;

str=new string("IITRoorkee");

- Write a C++ program to find whether given string is Palindrome or not.

malayalam

nitin

ada

What is the output of following strange code? Why?

```
int a[10];
cout<<a<<endl;
cout<<&a;
```

Sum = a + b,

a + b = sum;

l-value required.

```
Terminal
sh-4.3$ g++ -o main *.cpp
sh-4.3$ main
0x7fffbb8f65b0
0x7fffbb8f65b0sh-4.3$
```

```
int a[10];
cout<<a+0<<endl;
cout<<&(a+0);
```

l-value required

Garbage collection in Java

- In java, garbage means unreferenced objects.

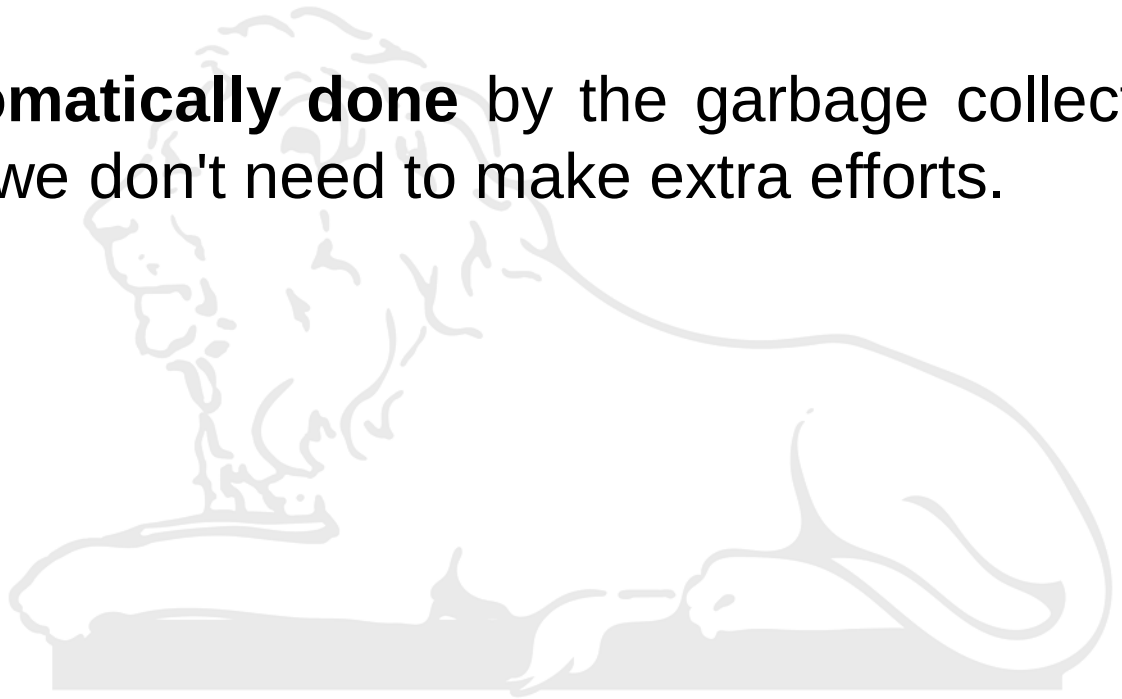
- Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

- To do so, we were using **`free()`** function in C language and **`delete()`** in C++. However, in java it is performed automatically. So, java provides better memory management.

Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.

- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

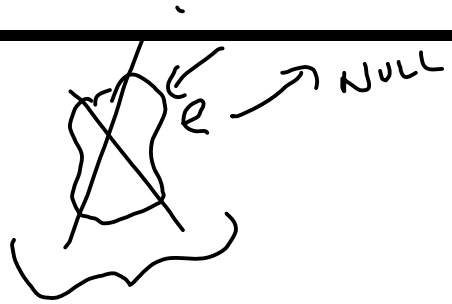


How can an object be unreferenced?

- By making the reference Null
- By assigning a reference to another
- By anonymous object etc.

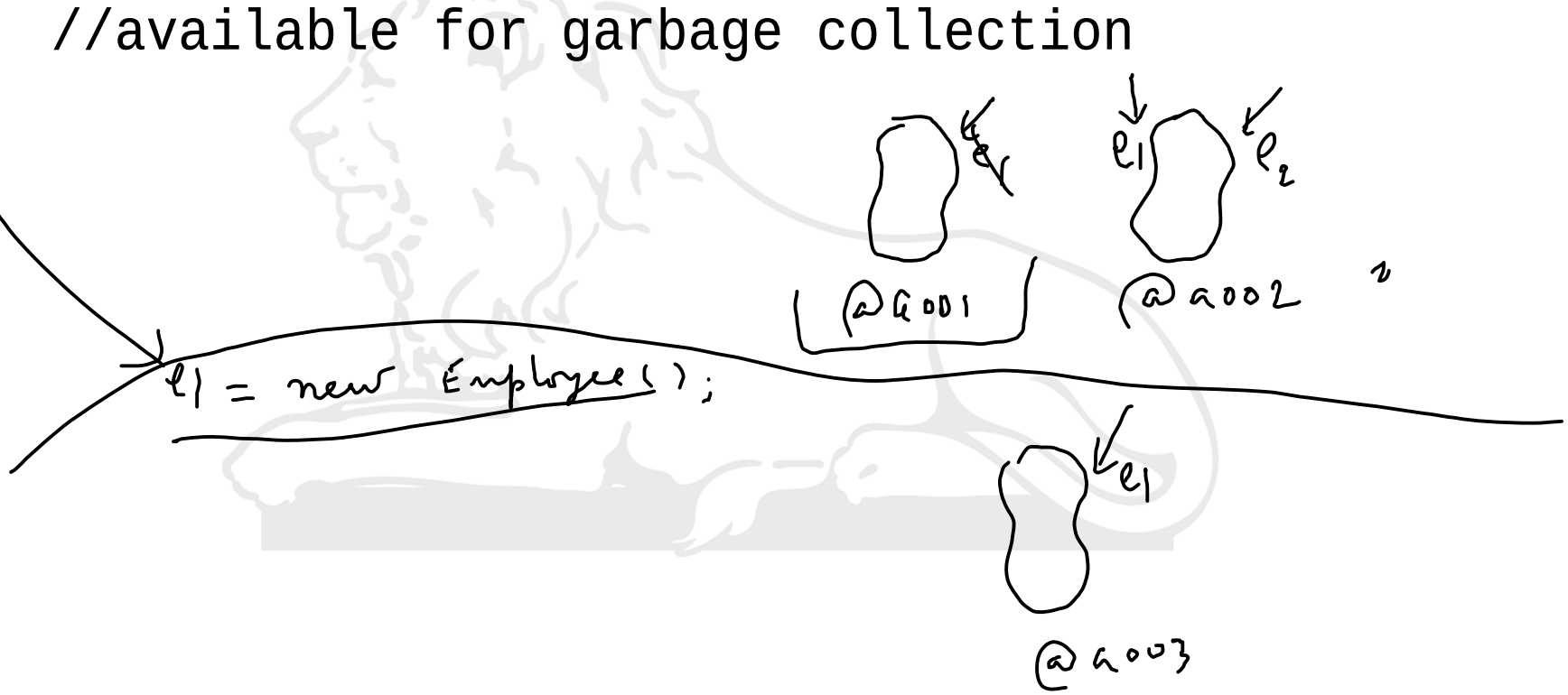
By nulling a reference:

```
Employee e=new Employee();
e=null;
```



By assigning a reference to another:

```
Employee e1=new Employee();
Employee e2=new Employee();
e1=e2;//now the first object referred by e1 is
    //available for garbage collection
```



By anonymous object:

```
new Employee();
```

`finalize()` method

`gc()` method

finalize() method

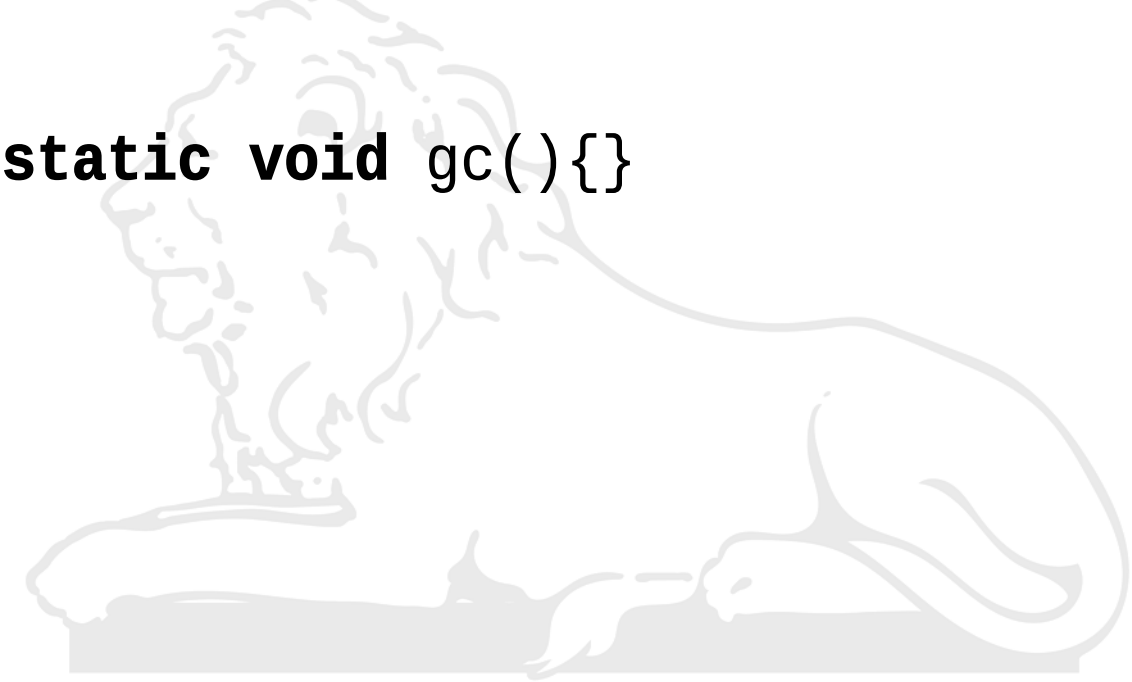
- The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
public void finalize(){}
```

gc() method

- The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Run-time classes.

```
public static void gc(){}
```



```java
/* package whatever; // don't place package name! */

import java.util.*;
import java.lang.*;
import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */
class Ideone
{
public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
   Ideone s1=new Ideone();
   Ideone s2=new Ideone();
   s1=null;
   s2=null;
   System.gc();
}
}

```

stdout

```
object is garbage collected
object is garbage collected
```

https://ideone.com/dN2zOU

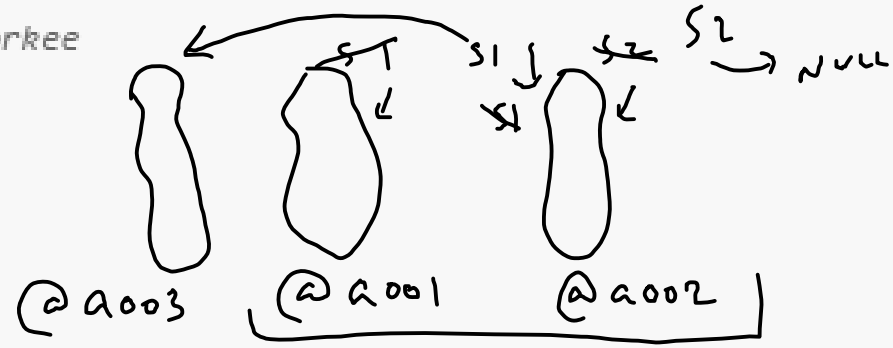
```java
/* package whatever; // don't place package name! */
//Program for CSN-103, IIT Roorkee

import java.util.*;
import java.lang.*;
import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */
class Ideone
{
public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
   Ideone s1=new Ideone();
   Ideone s2=new Ideone();
   s1=s2;
   s2=null;
   System.gc();
}
}
```

stdout

```
object is garbage collected
```

- https://ideone.com/pa48VC

```java
/* package whatever; // don't place package name! */
//Program for CSN-103, IIT Roorkee

import java.util.*;
import java.lang.*;
import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */
class Ideone
{
public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
   Ideone s1=new Ideone();
   Ideone s2=new Ideone();
   s1=s2;
   s2=null;
   s1=new Ideone();
   System.gc();
}
}
```

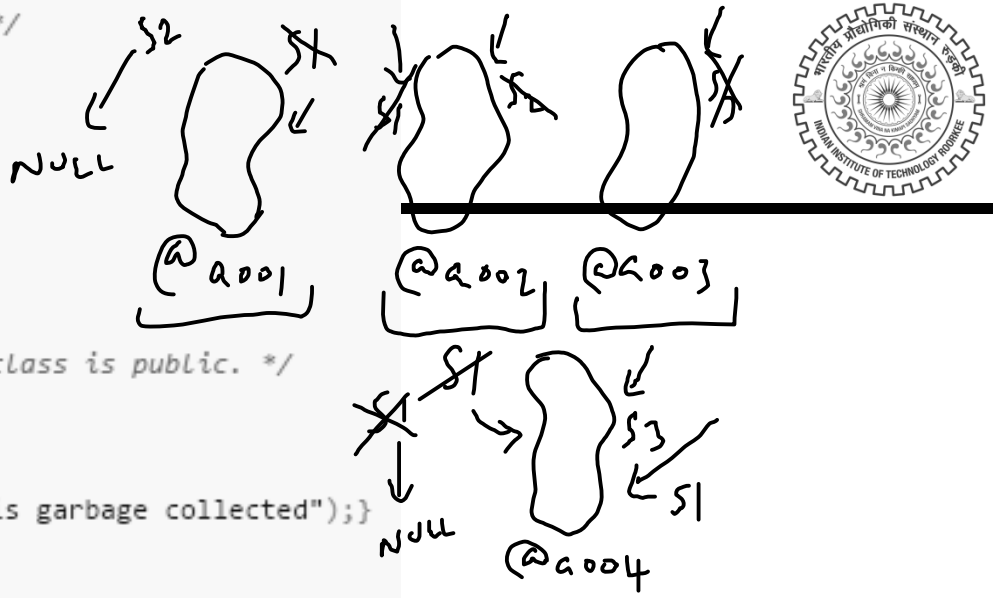


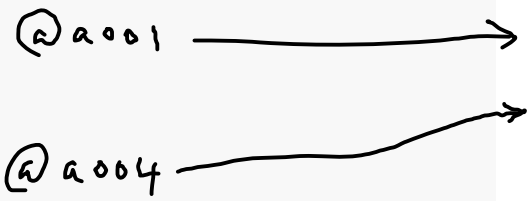
stdout

```
object is garbage collected
object is garbage collected
```

- https://ideone.com/BstbO6

```java
/* package whatever; // don't place package name! */
/* Exercise for CSN-103, IIT Roorkee */

import java.util.*;
import java.lang.*;
import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */
class Ideone
{
public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
  Ideone s1=new Ideone();
  Ideone s2=new Ideone();
  Ideone s3=new Ideone();
  System.out.println(s1);
  s1=s2;
  s1=new Ideone();
  System.out.println(s1);
  s2=null;
  s3=s1;
  s1=null;
  s1=s3;
  System.gc();
}
}
```



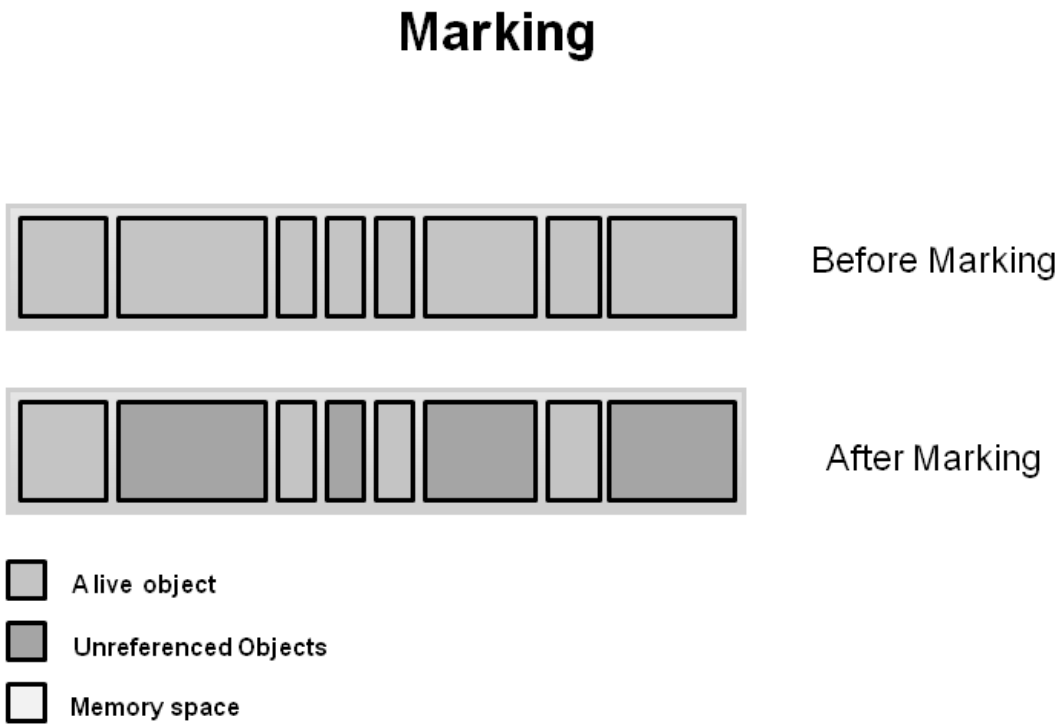


stdout

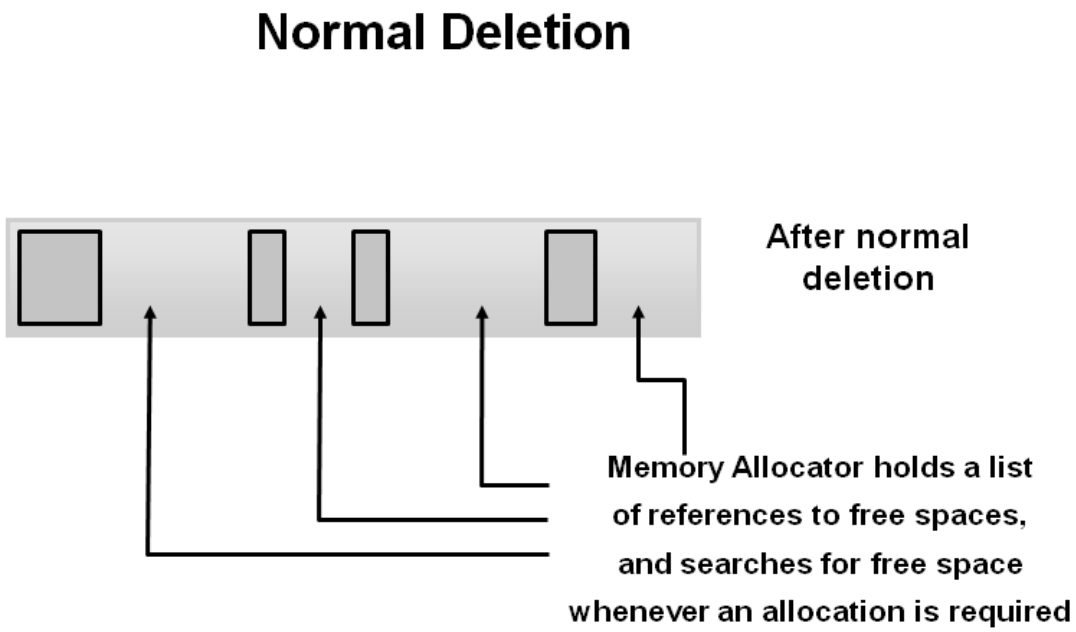
```
Ideone@106d69c
Ideone@52e922
object is garbage collected
object is garbage collected
object is garbage collected
```

- https://ideone.com/aD4PEs

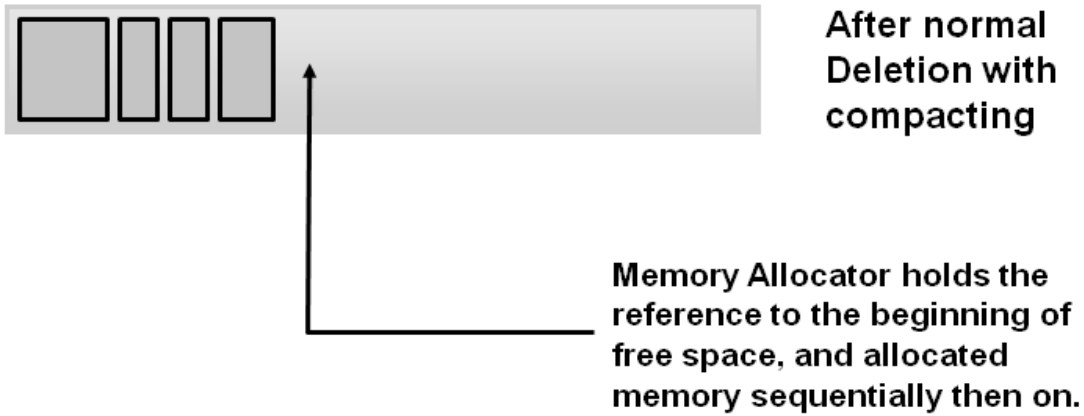
Marking

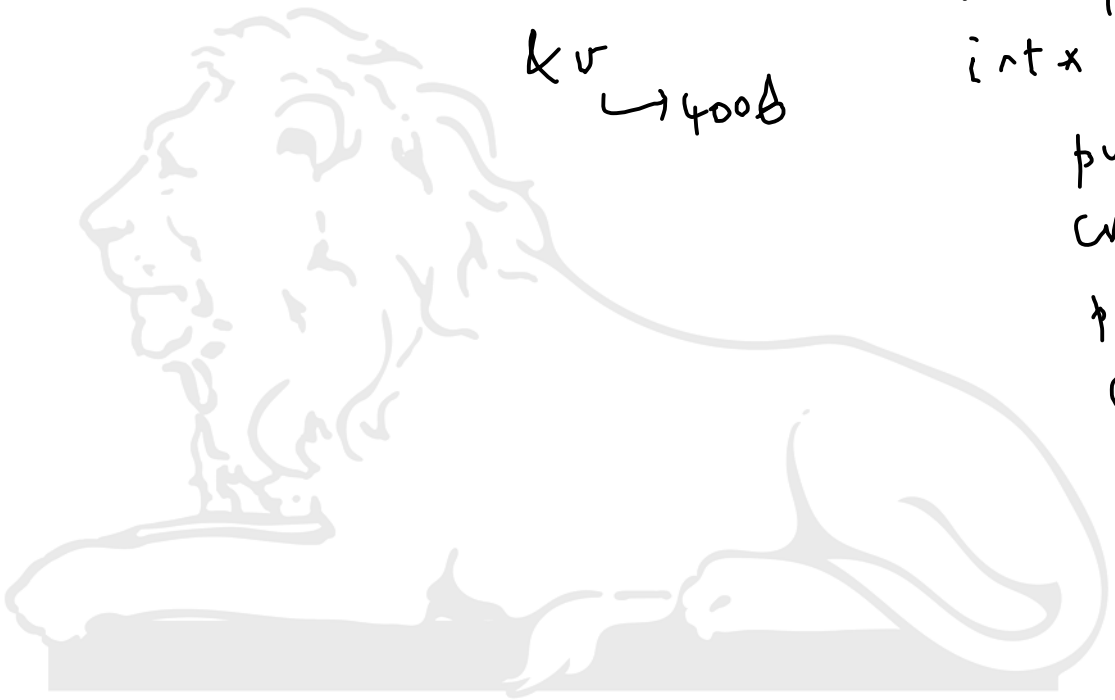

Marking
Before Marking
After Marking
A live object
Unreferenced Objects
Memory space

Normal Deletion



Deletion with Compacting





```
int u=10;
int v=10;
int* pu;
int* pv;
pu = &u;
cout << pu, 4002
pv = &v;
cout << pv; 4006
pv = pu;
cout << pv;
          4002
```

&u → 4002

&v → 4006

Objects invoke methods


```
1  class distance{
2          int feet;
3          int inches;
4          distance()
5          { }
6          distance(int x , int y)
7          {
8                  feet=x;
9                  inches=y;
10         }
11         void displaydistance()
12         {
13                 System.out.println(feet+" feet" + " " +inches+" inchess");
14         }
15         distance addDistance(distance two)
16         {
17                 distance df3=new distance();
18                 df3.feet=feet+two.feet;
19                 df3.inches=inches+two.inches;
20                 if(df3.inches>=12)
21                 {
22                         df3.feet++;
23                         df3.inches=df3.inches-12;
24                 }
25                 return df3;
26         }
27 }//distance type created
28
```

```
29  class Executedistance2{
30
31      public static void main(String[] args) {
32          distance d1=new distance(10,9);
33          System.out.println("the first distance is :");
34          d1.displaydistance();
35          distance d2=new distance(9,10);
36          System.out.println("the second distance is :");
37          d2.displaydistance();
38          distance d3=new distance();
39          d3=d1.addDistance(d2);
40          System.out.println("the sum of their distance is :");
41          d3.displaydistance();
42
43      }
44
45  }
46
```

Terminal

```
sh-4.3$ javac Executedistance2.java
sh-4.3$ java Executedistance2
the first distance is :
10 feet 9 inchess
the second distance is :
9 feet 10 inchess
the sum of their distance is :
20 feet 7 inchess
sh-4.3$
```

Returning the invoked object

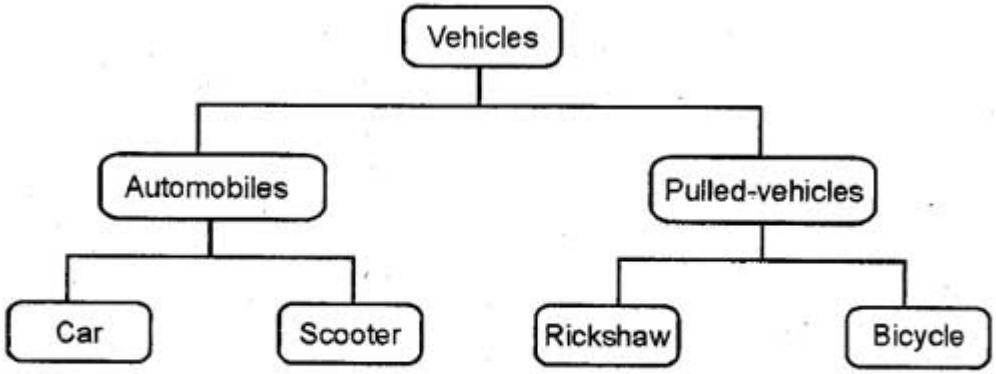
```
class distance{
        int feet;
        int inches;
        distance()
        { }
        distance(int x , int y)
        {
                feet=x;
                inches=y;
        }
        void displaydistance()
        {
                System.out.println(feet+" feet" + " " +inches+" inchess");
        }
        distance addDistance(distance two)
        {//Example for returning invoked object, not addition
         //Testing
                distance df3=new distance();
                df3.feet=feet+two.feet;
                df3.inches=inches+two.inches;
                if(df3.inches>=12)
                {
                        df3.feet++;
                        df3.inches=df3.inches-12;
                }
                //return df3;
                return this;
        }
}//distance type created

```

```
29 class Executedistance2{
30
31     public static void main(String[] args) {
32         distance d1=new distance(10,9);
33         System.out.println("the first distance is :");
34         d1.displaydistance();
35         distance d2=new distance(9,10);
36         System.out.println("the second distance is :");
37         d2.displaydistance();
38         distance d3=new distance();
39         d3=d1.addDistance(d2);
40         System.out.println("the sum of their distance is :");
41         d3.displaydistance();
42
43     }
44
45 }
46
```

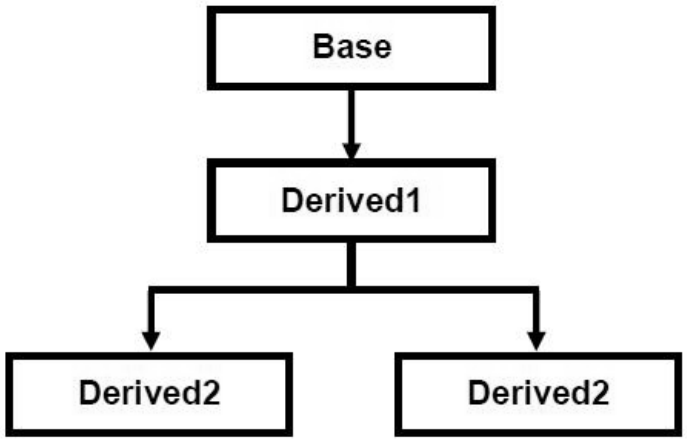
Terminal

```
sh-4.3$ javac Executedistance2.java
sh-4.3$ java Executedistance2
the first distance is :
10 feet 9 inchess
the second distance is :
9 feet 10 inchess
the sum of their distance is :
10 feet 9 inchess
sh-4.3$
```

Inheritance



Inheritance



Inheritance

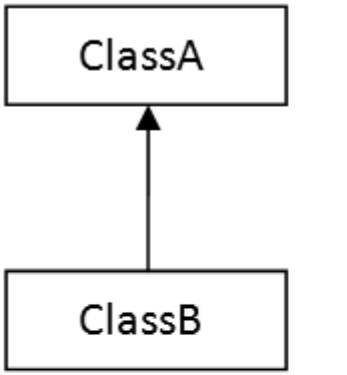
- Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another.
- With the use of inheritance the information is made manageable in a hierarchical order.

- The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

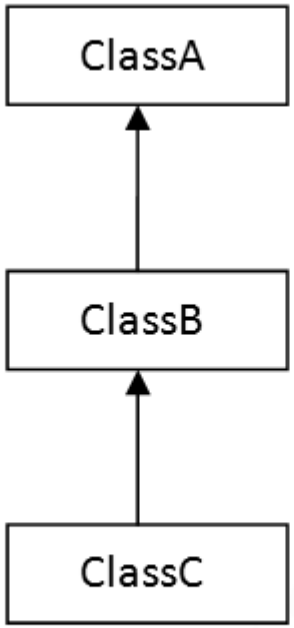
Use of inheritance in java

- For Method Overriding (runtime polymorphism can be achieved).
- For Code Reusability.

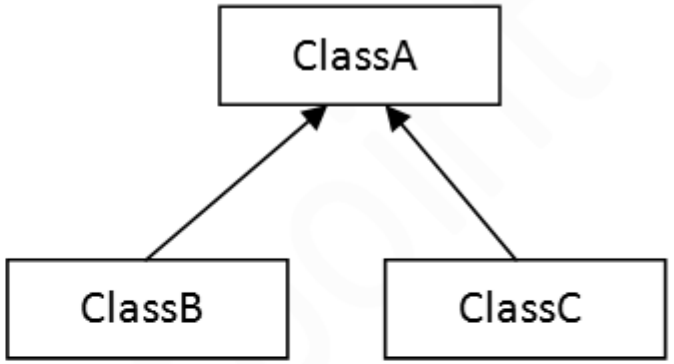
Types of Inheritance



1) Single



2) Multilevel



3) Hierarchical

Syntax of Java Inheritance

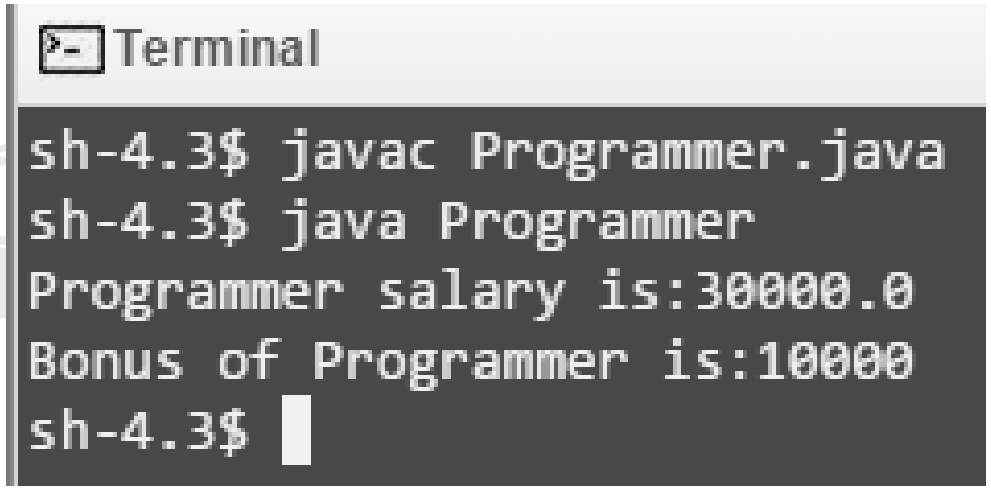
```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

ClassB

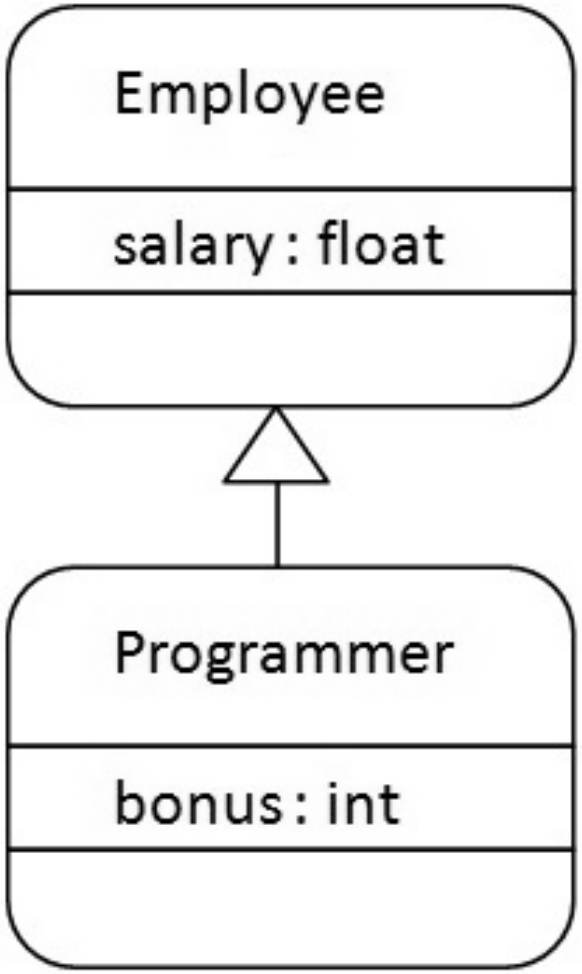
ClassA

Simple or Single Inheritance

```
1  class Employee{
2   float salary=30000;
3  }
4  class Programmer extends Employee{
5   int bonus=10000;
6   public static void main(String args[]){
7     Programmer p=new Programmer();
8     System.out.println("Programmer salary is:"+p.salary);
9     System.out.println("Bonus of Programmer is:"+p.bonus);
10 }
11 }
```



Understanding Simple Inheritance



Simple Inheritance

```
1  class Calculation{
2     int z;
3     public void addition(int x, int y){
4        z=x+y;
5        System.out.println("The sum of the given numbers:"+z);
6     }
7     public void Substraction(int x,int y){
8        z=x-y;
9        System.out.println("The difference between the given numbers:"+z);
10    }
11
12 }
13
14 public class My_Calculation extends Calculation{
15
16    public void multiplication(int x, int y){
17       z=x*y;
18       System.out.println("The product of the given numbers:"+z);
19    }
20    public static void main(String args[]){
21       int a=20, b=10;
22       My_Calculation demo = new My_Calculation();
23       demo.addition(a, b);
24       demo.Substraction(a, b);
25       demo.multiplication(a, b);
26
27    }
28
29 }
```

Terminal

```
sh-4.3$ javac My_Calculation.java
sh-4.3$ java My_Calculation
The sum of the given numbers:30
The difference between the given numbers:10
The product of the given numbers:200
sh-4.3$
```

Understanding Pointers

```
1.  //Understanding Pointers, CSN-103, IIT Roorkee
2.  #include <iostream>
3.  using namespace std;
4.
5.  int main() {
6.      int* pta;
7.     int a=10;
8.     pta=&a;
9.     int b=5;
10.    int* ptb;
11.    ptb=&b;
12.    cout<<pta<<endl;
13.    cout<<ptb<<endl;
14.    cout<<pta-ptb<<endl;
15.    cout<<&b-&a<<endl;
16.
17.    return 0;// your code goes here
18. }
```

stdout

```
0xbfe4a828
0xbfe4a82c
-1
1
```