```
Abstraction:
--------------
Hiding Internal implemetation and just highlet the set
of services what we are offering is the concept of Abstraction.
by using interfaces and abstract classes we can implement abstraction.

1.Normal Method
contain method declaration
constain method implementation
Ex:
void m1()
{}{
    //logic
}

2.abstract  methods
--contain only method declaration
--abstract methods ends with semicolon
--to represent method is a abstract we use abstract modifier.
------------------------------

1.Normal class

constain normal methods
we can create object
class Test
{}{

    void m1()
    {}{
    }
    void m2()
    {}{
    }
    void m3()
    {}{
    }
}

2.abstract classes

also known as Helper classes
contain abstract methods as well as normal methods
we can't create object for abstract classes
Ex:

abstract class Test
{}{
    //Normal Methods
```

```
50      void m1()
51      {}{
52      }
53      void m2()
54      {}{
55      }
56      void m3()
57      {}{
58      }
59      //abstract methods
60      abstract void m4();
61      abstract void m5();
62
63  }
64
65  Ex1:
66  //Ex1 Abstract class object creation is not possible
67
68  abstract class Parent
69  {}{
70  abstract void m1();
71  abstract void m2();
72  abstract void m3();
73
74  void m4()
75  {}{
76  System.out.println("Normal Method m4");
77  }
78  }
79  class Child extends Parent
80  {}{
81      void m1()
82      {}{
83      System.out.println("Overriding m1 method");
84      }
85      void m2()
86      {}{
87      System.out.println("Overriding m2 method");
88      }
89      void m3()
90      {}{
91      System.out.println("Overriding m3 method");
92      }
93      void m5()
94      {}{
95      System.out.println("Child m5 method");
96      }
97  public static void main(String[] args)
98  {}{
```

```
 99      //Parent p=new Parent();
100      //Child c=new Child();
101      Parent c=new Child();
102      Child c1=(Child)c;
103      c1.m1();
104      c1.m2();
105      c1.m3();
106      c1.m4();
107      c1.m5();
108  }
109  }
110  output:
111
112  D:\KLUniversity\OOPCourse\S15Abstract>javac Child.java
113  D:\KLUniversity\OOPCourse\S15Abstract>java Child
114  Overriding m1 method
115  Overriding m2 method
116  Overriding m3 method
117  Normal Method m4
118  Child m5 method
119
120  D:\KLUniversity\OOPCourse\S15Abstract>
121  ----------------------------------------
122  Ex2:
123  //Ex2:abstract class with instance variable,constructor,Concrete Method
124
125  abstract class Car
126  {}{
127      //inastance variable
128      int regno;
129
130      //constructor
131      Car(int no)
132      {}{
133      regno=no;
134      }
135      //concrete method
136      void fillTank()
137      {}{
138      System.out.println("Open Tank and Fill it");
139      }
140      //abstract methods
141      abstract void steering(int direction);
142      abstract void braking(int force);
143
144  }
145  class Maruti extends Car
146  {}{
147      Maruti(int no)
```

```
148     {}{
149         super(no);
150     }
151     void steering(int direction)
152     {}{
153         System.out.println("maruti has manual steering");
154         System.out.println("Please Drive the Car");
155     }
156     void braking(int force)
157     {}{
158         System.out.println("maruti has Gas brakes");
159         System.out.println("Apply the brakes");
160     }
161 }
162
163 class Santro extends Car
164 {}{
165     Santro(int no)
166     {}{
167         super(no);
168     }
169     void steering(int direction)
170     {}{
171         System.out.println("Santro has power steering");
172         System.out.println("Start the Car");
173     }
174     void braking(int force)
175     {}{
176         System.out.println("Santro has Hydralic brakes");
177         System.out.println("Stop the Car");
178     }
179 }
180 class AbstractTest
181 {}{
182 public static void main(String[] args)
183 {}{
184
185     Maruti m=new Maruti(9999);
186     Santro s=new Santro(7777);
187     Car c,d;
188     c=m;
189
190     d=s;
191 System.out.println("Reg.No:"+c.regno);
192 c.fillTank();
193 c.steering(2);
194 c.braking(46);
195 System.out.println("------------------");
196 System.out.println("Reg.No:"+d.regno);
```

```
d.fillTank();
d.steering(2);
d.braking(46);

}
}
output:

D:\KLUniversity\OOPCourse\S15Abstract>javac AbstractTest.java

D:\KLUniversity\OOPCourse\S15Abstract>java AbstractTest
Reg.No:9999
Open Tank and Fill it
maruti has manual steering
Please Drive the Car
maruti has Gas brakes
Apply the brakes
------------------
Reg.No:7777
Open Tank and Fill it
Santro has power steering
Start the Car
Santro has Hydralic brakes
Stop the Car

D:\KLUniversity\OOPCourse\S15Abstract>
--------------------------------------------------
Ex 3:Bank as conatin customers who hold accounts within the bank.
An Account can be of Two Types:
    Saving Account
      or
    Current Account
Customers can perform deposit or withdrawl operation on their respectiv
Banks provide interest to the saving account holders which it can chang
Banks provide a upper limit to the current account holders.
In case the balance in their account is less than what needs to be with
so when ever the current account account holder deposits the amount in
A customer will have a name and id and will holding an account(either s
Every account will have an id and balance.
Saving account will have an interest rate apart from id and balance.
current accounts will have an overdraft limit apart from id and balance

So we have created five classes:
    Customer class.
    Account class
        Subclasses
        -----------
        SavingAccount
        CurrentAccount
```

```
246     Bank class to hold customers
247 ------------------------------------
248 package klu.bank.sbh;
249 abstract class Account
250 {}{
251     float balance;
252 
253     private String accountNumber;
254 
255     Account(float bal,String acno)
256     {}{
257         balance=bal;
258         accountNumber=acno;
259     }
260 
261     float getBalance()
262     {}{
263         return balance;
264     }
265     void setBalance(float b)
266     {}{
267         balance=b;
268     }
269 
270     String getAccountNumber()
271     {}{
272         return accountNumber;
273     }
274 
275     void setAccountNumber(String acn)
276     {}{
277         accountNumber=acn;
278     }
279     void display()
280     {}{
281         System.out.println("Account Number:="+accountNumber);
282         System.out.println("Account Balance:="+balance);
283 
284     }
285 
286     abstract void debit(float amount);
287     abstract void credit(float amount);
288 
289 }
290 package klu.bank.sbh;
291 class CurrentAccount extends Account
292 {}{
293     float overdraftborrowed;
294     float overdraftlimit;
```

```
295
296     CurrentAccount(float b,String acno,float od)
297     {}{
298         super(b,acno);
299         overdraftlimit=od;
300
301     }
302     void setOverdraft(float o)
303     {}{
304         overdraftlimit=o;
305
306     }
307
308     public void credit(float amount)
309     {}{
310         System.out.println("Amount to be credited:"+amount);
311         System.out.println("Old Balance:"+balance);
312         System.out.println("Overdraft Borrowed:"+overdraftborrowed);
313
314         if(amount>overdraftborrowed)
315         {}{
316             amount=amount-overdraftborrowed;
317             overdraftborrowed=0;
318             balance=balance+amount;
319
320         }
321         else if(amount<overdraftborrowed)
322         {}{
323             overdraftborrowed=overdraftborrowed-amount;
324         }
325         System.out.println("New Overdraft Borrowed:"+overdraftborrowed)
326         System.out.println("New Balance:"+balance);
327
328     }
329     public void debit(float amount)
330     {}{
331         System.out.println("Amount to be debited:"+amount);
332         System.out.println("Old Balance:"+balance);
333         if(amount<=balance)
334         balance=balance-amount;
335
336         else if((amount>balance)&&(amount<(balance+overdraftlimit)))
337         {}{
338             overdraftborrowed=amount-balance;
339             balance=0;
340             System.out.println("Overdraft Borrowed:"+overdraftborrowed)
341         }
342         else
343             System.out.println("Reguest Denied");
```

```
344         System.out.println("New Balance:"+balance);
345         System.out.println("Overdraft Borrowed:"+overdraftborrowed);
346 
347     }
348     public void display()
349     {}{
350         super.display();
351         System.out.println("Overdraft limit:"+overdraftlimit);
352 
353     }
354     public String toString()
355     {}{
356         return "Current Account No:"+getAccountNumber()+"Balance :"+bal
357 
358     }
359 
360 }
361 package klu.bank.sbh;
362 class SavingAccount extends Account
363 {}{
364     static float interest=4;
365 
366     SavingAccount(float b,String acno)
367     {}{
368         super(b,acno);
369     }
370     SavingAccount()
371     {}{
372         super(0,"");
373     }
374 
375     static void setInterest(float i)
376     {}{
377         interest=i;
378 
379     }
380     public void display()
381     {}{
382         super.display();
383         System.out.println("Interest rate:"+interest);
384     }
385     public void credit(float amount)
386     {}{
387         System.out.println("Amount to be credited:"+amount);
388         System.out.println("OLD Balance:"+balance);
389         balance=balance+amount;
390         System.out.println("New Balance:"+balance);
391 
392     }
```

```
393     public void debit(float amount)
394     {}{
395         System.out.println("Amount to be debited:"+amount);
396         System.out.println("OLD Balance:"+balance);
397         if(amount<balance)
398         {}{
399 
400         balance=balance-amount;
401         System.out.println("New Balance:"+balance);
402         }
403         else
404             System.out.println("Request Denied");
405 
406     }
407     public void creditInterest()
408     {}{
409         float temp=balance*interest/100;
410         System.out.println("Interest paid:"+temp);
411         balance=balance+temp;
412         System.out.println("New Balance:"+balance);
413     }
414 
415     public String toString()
416     {}{
417         return "Saving Account No:"+getAccountNumber()+"Balance:"+balar
418     }
419 
420 }
421 package klu.bank.sbh;
422 class Customer
423 {}{
424 String custName;
425 String custId;
426 private Account account;
427 
428 Customer(String custName,String custId,Account account)
429     {}{
430     this.custName=custName;
431     this.custId=custId;
432     this.account=account;
433     }
434 public void deposit(float amt)
435     {}{
436     if(account instanceof SavingAccount)
437         ((SavingAccount)account).credit(amt);
438     else
439         if(account instanceof CurrentAccount)
440         ((CurrentAccount)account).credit(amt);
441     }
```

```
442     void depositInterest()
443     {}{
444         System.out.println("Depositing Interest in:"+custId);
445         if(account instanceof SavingAccount)
446         ((SavingAccount)account).creditInterest();
447 
448     }
449     public void withdrawl(float amt)
450     {}{
451         if(account instanceof SavingAccount)
452         ((SavingAccount)account).debit(amt);
453         else
454             if(account instanceof CurrentAccount)
455         ((CurrentAccount)account).debit(amt);
456     }
457     public void display()
458     {}{
459         System.out.println("Customer Name:"+custName);
460         System.out.println("Customer Id:"+custId);
461         account.display();
462         System.out.println(account);
463     }
464 }
465 package klu.bank.sbh;
466 public class Bank
467 {}{
468     private Customer c[]=new Customer[3];
469     public Bank()
470     {}{
471         c[0]=new Customer("Yellaswamy","C001",new SavingAccount(12000,'
472         c[1]=new Customer("Ashok","C002",new SavingAccount(12000,"A002'
473         c[2]=new Customer("Shashank","C003",new SavingAccount(12000,"A0
474     }
475     void changeInterestRate(float i)
476     {}{
477         SavingAccount.setInterest(i);
478     }
479     public static void main(String[] args)
480     {}{
481         Bank b=new Bank();
482         b.changeInterestRate(6);
483         b.demo();
484         b.c[0].depositInterest();
485         b.c[1].depositInterest();
486 
487     }
488     public void demo()
489     {}{
490         c[0].display();
```

```
491         c[0].deposit(1000);
492         c[0].withdrawl(15000);
493 
494 
495         c[1].display();
496         c[1].deposit(2000);
497         c[1].withdrawl(8000);
498 
499 
500 
501     }
502 }
503 output:
504 
505 
506 D:\KLUniversity\OOPCourse\S15Abstract\Bank>javac -d . *.java
507 
508 D:\KLUniversity\OOPCourse\S15Abstract\Bank>java klu.bank.sbh.Bank
509 Customer Name:Yellaswamy
510 Customer Id:C001
511 Account Number:=A001
512 Account Balance:=12000.0
513 Interest rate:6.0
514 Saving Account No:A001Balance:12000.0
515 Amount to be credited:1000.0
516 OLD Balance:12000.0
517 New Balance:13000.0
518 Amount to be debited:15000.0
519 OLD Balance:13000.0
520 Request Denied
521 Customer Name:Ashok
522 Customer Id:C002
523 Account Number:=A002
524 Account Balance:=12000.0
525 Interest rate:6.0
526 Saving Account No:A002Balance:12000.0
527 Amount to be credited:2000.0
528 OLD Balance:12000.0
529 New Balance:14000.0
530 Amount to be debited:8000.0
531 OLD Balance:14000.0
532 New Balance:6000.0
533 Depositing Interest in:C001
534 Interest paid:780.0
535 New Balance:13780.0
536 Depositing Interest in:C002
537 Interest paid:360.0
538 New Balance:6360.0
539 
```

```
D:\KLUniversity\OOPCourse\S15Abstract\Bank>
```