Abstraction

Abstraction is the concept of exposing only the required essential characteristics and behavior with respect to a context.
Hiding of data is known as data abstraction. In object oriented programming language this is implemented automatically while writing the code in the form of class and object.
Real Life Example of Abstraction in Java
Abstraction shows only important things to the user and hides the internal details, for example, when we ride a bike, we only know about how to ride bikes but cannot know how it work? And also we do not know the internal functionality of a bike.

Another real life example of Abstraction is ATM Machine; All are performing operations on the ATM machine like cash withdrawal, money transfer, retrieve mini-statement…etc. but we can't know internal details about ATM.

Real Life Example of Abstraction

Note: Data abstraction can be used to provide security for the data from the unauthorized methods.
Note: In Java language data abstraction can achieve using class.
Example of Abstraction

```
class Customer
{
int account_no;
float balance_Amt;
String name;
int age;
String address;
void balance_inquiry()
{
/* to perform balance inquiry only account number
is required that means remaining properties
are hidden for balance inquiry method */
```

```
}
void fund_Transfer()
{
/* To transfer the fund account number and
balance is required and remaining properties
are hidden for fund transfer method */
}
```

How to achieve Abstraction?

There are two ways to achieve abstraction in java

- Abstract class (0 to 100%)
- Interface (Achieve 100% abstraction)

Difference between Encapsulation and Abstraction

Encapsulation is not providing full security because we can access private member of the class using reflection API, but in case of Abstraction we can't access static, abstract data member of a class.

Abstract class

We know that every Java program must start with a concept of class that is without the class concept there is no Java program perfect.

In Java programming we have two types of classes they are

- Concrete class
- Abstract class

Concrete class in Java

A concrete class is one which is containing fully defined methods or implemented methods.

Example

```
class  Helloworld
{
void  display()
{
System.out.println("Good Morning. ......");
}
}
```

Here Helloworld class is containing a defined method and object can be created directly.

Create an object

```
Helloworld obj=new  Helloworld();
obj.display();
```

Every concrete class has specific features and these classes are used for specific requirement, but not for common requirement.

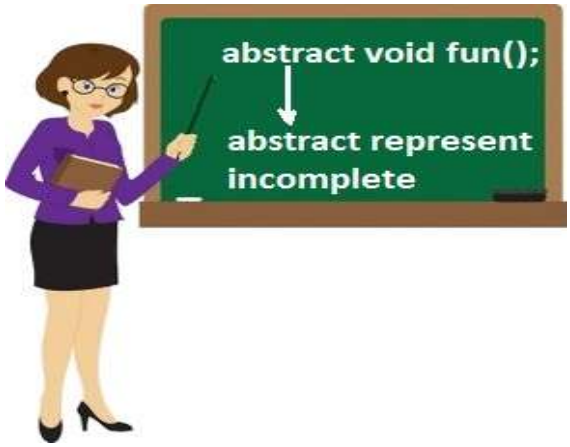
If we use concrete classes for fulfill common requirements than such application will get the following limitations.

- Application will take more amount of memory space (main memory).
- Application execution time is more.
- Application performance is decreased.

To overcome above limitation you can use abstract class.

Abstract class in Java

A class that is declared with abstract keyword is known as abstract class. An abstract class is one which is containing some defined methods and some undefined methods. In java programming undefined methods are known as un-implemented or abstract methods.



Syntax
abstract class className
{
.....
}
Example
abstract class A
{
.....
}
If any class has any abstract method then that class becomes an abstract class.
Example
class Vahicle
{
abstract void Bike();
}
Class Vehicle is become an abstract class because it has abstract Bike() method.
Make class as abstract class
To make the class as abstract class, whose definition must be preceded by an abstract keyword.
Example
abstract class Vehicle
{
.....
}
Abstract method
An abstract method is one which contains only declaration or prototype but it never contains body or definition. In order to make any undefined method as abstract whose declaration must be predefined by abstract keyword.
Syntax
abstract ReturnType methodName(List of formal parameter)
Example
abstract void sum();
abstract void diff(int, int);
Example of abstract class
abstract class Vehicle
{

```
 abstract void speed();  // abstract method
}
class Bike extends Vehicle
{
void speed()
{
System.out.println("Speed limit is 40 km/hr..");
}
public static void main(String args[])
{
 Vachile obj = new Bike(); //indirect object creation
 obj.speed();
 }
}
```

Output

Speed limit is 40 km/hr..

Create an Object of abstract class

An object of abstract class cannot be created directly, but it can be created indirectly. It means you can create an object of abstract derived class. You can see in above example

Example

```
Vachile obj = new Bike(); //indirect object creation
```

Important Points about abstract class

- Abstract class of Java always contains common features.
- Every abstract class participates in inheritance.
- Abstract class definitions should not be made as final because abstract classes always participate in inheritance classes.
- An object of abstract class cannot be created directly, but it can be created indirectly.
- All the abstract classes of Java makes use of polymorphism along with method overriding for business logic development and makes use of dynamic binding for execution logic.

Advantage of abstract class

- Less memory space for the application
- Less execution time
- More performance

Why abstract classes have no abstract static method?

In abstract classes we have the only abstract instance method, but not containing abstract static methods because every instance method is created for performing repeated operation where as static method is created for performing a onetime operations in other word every abstract method is instance but not static.

Abstract base class

An abstract base class is one which is containing physical representation of abstract methods which are inherited by various sub classes.

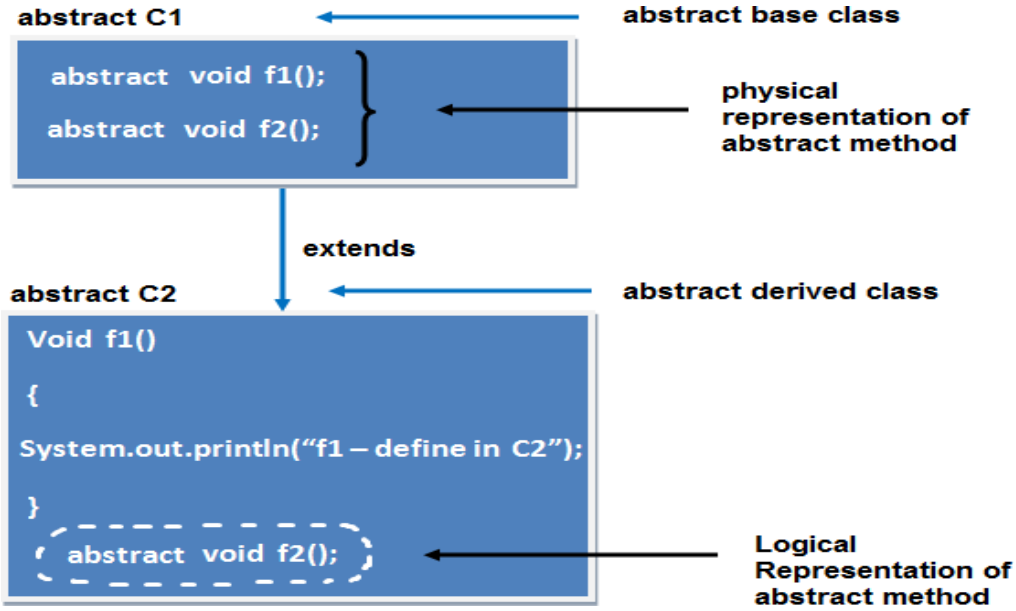
Abstract derived class

An abstract derived class is one which is containing logic representation of abstract methods which are inherited from abstract base class with respect to both abstract base class and abstract derived class one cannot create objects directly, but we can create their objects indirectly both abstract base class and abstract derived class are always reusable by various sub classes.

When the derived class inherits multiple abstract methods from abstract base class and if the derived class is not defined at least one abstract method then the derived class is known as

abstract derived class and whose definition must be made as abstract by using abstract keyword

If the derived class defined all the abstract methods which are inherited from abstract Base class, then the derived class is known as concrete derived class.



Example of abstract class having method body

```
abstract class Vehicle
{
 abstract void speed();
 void mileage()
{
 System.out.println("Mileage is 60 km/ltr..");
}
}
class Bike extends Vehicle
{
void speed()
{
System.out.println("Speed limit is 40 km/hr..");
}
public static void main(String args[])
{
 Vehicle obj = new Bike();
 obj.speed();
 obj.mileage();
 }
}
```

Output

```
Mileage is 60 km/ltr..
Speed limit is 40 km/hr..
```

Example of abstract class having constructor, data member, methods

```
abstract class Vehicle{
 int limit=40;
 Vehicle()
{
System.out.println("constructor is invoked");
}
```

```
 void getDetails()
{
System.out.println("it has two wheels");
}
 abstract void run();
}

class Bike extends Vehicle
{
 void run()
{
System.out.println("running safely..");
}
 public static void main(String args[])
{
  Vehicle obj = new Bike();
  obj.run();
  obj.getDetails();
  System.out.println(obj.limit);
 }
}
```

Output

```
constructor is invoked
running safely..
it has two wheels
40
```

Difference between Abstract class and Concrete class

Concrete class	Abstract class
A Concrete class is used for specific requirement	Abstract class is used to fulfill a common requirement.
Object of concrete class can create directly.	Object of an abstract class can not create directly (can create indirectly).
Concrete class containing fully defined methods or implemented method.	Abstract class has both undefined method and defined method.