Abstract Class and Abstract Method: Rules:

1) A method without body (no implementation) is known as abstract method.
   Example
   ```
   Class Vehicle
   {
    Int no_of_tyres;
   abstract void start();
   }
   ```

2) A abstract method must always be declared in an abstract class

   Example
   ```
    abstract class Vehicle
   {
    Int no_of_tyres;
   abstract void start();
   }
   ```
3) If a regular class extends an abstract class , then the class must be have to implement all the abstract method of abstract parent class or it has to be declared abstract as well.
   ```
   Class Car extends  Vehicle
   {

    void start();
   {
    System.out.println (“ Start with key”);
   }
   }
   ```

4) Abstract methods in an abstract class are meant to overridden in concrete classes otherwise compile time error will be thrown.
   ```
   Class Scooter extends  Vehicle
   {
    void start();
   {
    System.out.println (“ Start with kick”);
   }
   }
   ```

5) Abstract classes cannot be instantiated means we can’t create an abject of abstract class.

```
abstract class Vehicle {
   int no_of_tyres;

   abstract void start();
}

class Car extends Vehicle {
   void start() {
      System.out.println(" Start with key");
   }
}

class Scooter extends Vehicle {
   void start() {
      System.out.println("Start with kick");
   }

   public static void main(String[] args) {
      Car c = new Car();
      c.start();
      Scooter s = new Scooter();
      s.start();
   }
}
```

6) Let us make a program where the abstract class “MyClass” have one abstract method which has got various implementations in sub-classes

```
abstract class MyClass {
    abstract void calculate(double x);
}
class sub1 extends MyClass{
    void calculate(double x){
        System.out.println("Sqaure : "+(x*x));
    }
}
class sub2 extends MyClass{
    void calculate(double x){
        System.out.println("Sqaure root : "+Math.sqrt(x));
    }
}
class sub3 extends MyClass{
    void calculate(double x){
        System.out.println("Cube : "+(x*x*x));
    }
}
public class app6{
    public static void main(String[] args) {
        sub1 s1 = new sub1();
        sub2 s2 = new sub2();
        sub3 s3 = new sub3();
        s1.calculate(12);
        s2.calculate(144);
        s3.calculate(4);
    }
}
```

Abstract method using Reference variable

Class **CAR**

```
abstract class car{
    int regno;
    car(int r){
        regno=r;
    }
    void open_tank(){
        System.out.println("Fill the tank");
    }
    abstract void steering(int direction, int angle);
    abstract void braking(int force);
}
```

```
Compilation : javac car.java
```

Class **Maruti**:

```
public class maruti extends car {
    maruti(int regno){
        super(regno);
    }
    void steering(int direction, int angle){
        System.out.println("Take a turn");
        System.out.println("This car uses ordinary
steering");
    }
    void braking(int force){
        System.out.println("Braking applied");
        System.out.println("This car uses hydraulic
brakes");
    }
}
```

Compilation : javac Maruti.java

Class santro:

```
public class santro extends car {
    santro(int regno){
        super(regno);
    }
    void steering(int direction, int angle){
        System.out.println("Take a turn");
        System.out.println("This car uses power 
steering");
    }
    void braking(int force){
        System.out.println("Braking applied");
        System.out.println("This car uses gas brakes");
    }
}
```

Compilation : javac santro.java

The main class **use_car**:

```
//taking reference of maruti

public class use_car {
    public static void main(String[] args) {

        maruti m= new maruti(2020);
        santro s= new santro(3020);
        car ref;
        ref=m;
        ref.open_tank();
        ref.steering(1,50);
        ref.braking(500);

    }
}
```

Output :

```
Fill the tank
Take a turn
This car uses ordinary steering
Braking applied
This car uses hydraulic brakes
```

```
//taking reference of santro

public class use_car {
    public static void main(String[] args) {

        maruti m= new maruti(2020);
        santro s= new santro(3020);
        car ref;
        ref=s;
        ref.open_tank();
        ref.steering(1,50);
        ref.braking(500);

    }
}
```

Output :

```
Fill the tank
Take a turn
This car uses power steering
Braking applied
This car uses gas brakes
```

Key points on abstract class & methods

- An abstract class is a class that contains zero or more abstract methods
- An abstract class can contain instance variable and concrete methods in addition to abstract method.
- An abstract class and abstract method should be declared by using the keyword “abstract”
- All abstract method of the abstract class should be implemented (body) in its sub-classes
- If any abstract method is not implemented, then that sub-class should be declared as abstract. In this case, we cannot create an object to the sub-class. We should create another sub-class to this sub-class and implement the remaining abstract method there.
- We cannot create an object to abstract class
- But we can create a reference of abstract class type
- The reference of abstract class can be used to refer to object(s) of its subclasses.
- The reference of abstract class cannot refer to individual methods of its sub-classes
- We cannot declare a class as both abstract and final

    For example :

    - Abstract final class app