Learning Goals

Click to Continue

This session will help you to understand,

- Abstract classes and methods.
- How to implement abstract classes?
- Abstract class usage.

Abstract definition

Click to Continue

Abstract:

1. *Something which is conceptual*
2. *Something which is theoretical*

Abstract methods

Click to Continue

Abstract methods are methods which do **not** have any **implementation**.

Abstract Method Example:

public abstract void divide(int number, int divisor);

This is a ***method*** with just the signature and no implementation.

Abstract class

Click to Continue

An class which contains one or more abstract methods is called a abstract class.

This is a abstract class with two abstract methods.

```
abstract class Automobile{
        public abstract void drive();
        public abstract void turnRight();
}
```

- "***Abstract***" keyword is used to define a abstract class.
- Abstract class can contain non abstract methods also.
- If a class has one or more abstract methods then class should be marked as ***abstract***.

Abstract Class Salient Points

Click to Continue

- Abstract classes cannot be instantiated. You will get an compilation error.
- Classes can extend abstract class.
- The subclass of abstract class should implement all the abstract methods in the base class.
- If the abstract methods in the parent class is not implemented in the subclass the subclass should also be marked as **abstract**.

Extending Abstract class

Click to Continue

Parent Class

```
package com.interfaces;

public abstract class Vehicle {

    public abstract void applyBrake();

}
```

This is the Abstract method.

Class extends abstract class

Subclass

```
package com.interfaces;

public class TwoWheeler extends Vehicle {

    public void applyBrake() {
        System.out.println("Apply Oil Brake");
    }

}
```

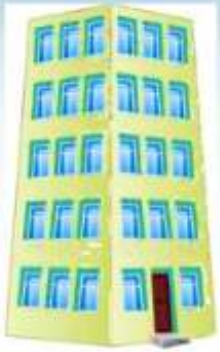
Abstract method implemented.

If the **applyBrake** method not implemented, Java will throw a compilation error.

Real world analogy

Click to Continue

Let u look at a analogy.



RBI is a central banking authority which defines the processes which the other banks needs to follow.

Say for example, for a customer to open an account, RBI slates few rules pertaining to the personal information to be gathered from the customer.

These rules needs to be adhered by all the other banks.

Abstract classes Usage:

Click to Continue

- Abstract class can be used if you want to impose method implementations in sub class.
  - Say for example you have a method **sayHello()** in parent class and you want all the subclasses to implement it. You can mark this method as ***abstract***.

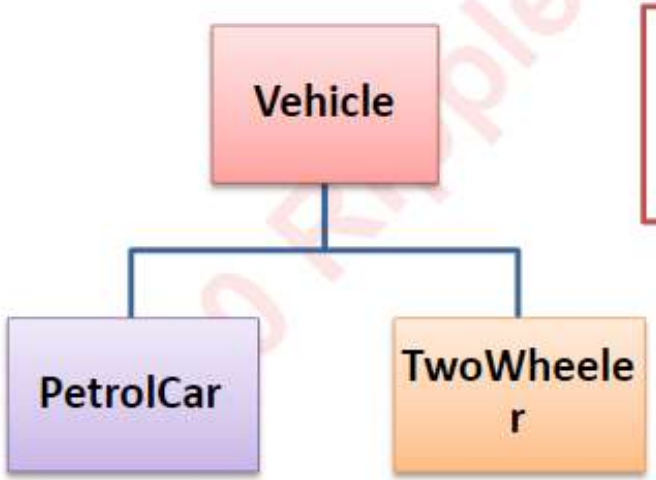
Let us look at the illustration in the next slide.

Usage Illustration

Click to Continue

Assume there is a abstract class **Vehicle** with two subclasses "**PetrolCar**" and "**TwoWheeler**". Assume there are two methods in Vehicle.

- applyBrake
- fillFuel



The method **applyBrake** will be marked as abstract to enforce all vehicles to implement their own brakes namely air or oil brakes.

But **fillfuel** method is same for both the vehicles since both runs on petrol. So this method will be implemented in the abstract class as a common method to be reused by both the sub classes.

Try it Out - Abstract classes

Click to Continue

Let us simulate a banking Application: RBI bank dictates the process and rules of other banks. Assume ABC and XYZ are two banks. RBIBank defines two process one

- openAccount – This is a common process for all bank.
- closeAccount – This is a process which is very specific to banks.

1. Create an Abstract class **RBIBank** with the below methods:
    - openAccountProcess(); -- Add an SOP "RBI opened account" in this method.
    - closeAccountProcess(); -- mark this method abstract.
2. Create a class for ABCbank extend the abstract class. Implement the abstract method with SOP "ABC bank closed the account"
3. Create a class **XYZBank.java** extend the abstract class implement the abstract method with SOP "XYZ bank closed the account"

Now create a **BankManager** Class with a main method this should perform the following actions,

- Create an object of ABCBank and invoke method openAccountProcess().
- Create an object of ABCBank and invoke method closeAccountProcess().

Repeat the same for XYZBank and observe the SOP getting printed.

Try it Out-Solution

Click to Continue

```
package com.demo.abstractclass;

public abstract class RBIBank {

    public void openAccountProcess() {
        System.out.println("RBI opened process...");
    }

    public abstract void closeAccountProcess();
}
```

openAccountProcess is created as a method which can be reused by the sub classes.

closeAccountProcess is created as a abstract method needs to be implemented by the sub classes.

```
package com.demo.abstractclass;

public class ABCBank extends RBIBank {
    public void closeAccountProcess() {
        System.out.println("ABC Bank Closed the Account...");
    }

}
```

ABCBank extends **RBIBank** abstract class and implements **closeAccountProcess** method.

Try it Out-Solution

Click to Continue

```
package com.demo.abstractclass;

public class XYZBank extends RBIBank {
    public void closeAccountProcess() {
        System.out.println("XYZ Bank Closed The Account...");
    }

}
```

XYZBank extends **RBIBank** abstract class and implements **closeAccountProcess** method.

If abstract method is not implemented you will get a compilation error.

```
package com.demo.abstractclass;

public class BankManager {

    public static void main(String[] args) {
        ABCBank abc = new ABCBank();
        abc.openAccountProcess();
        abc.closeAccountProcess();
        XYZBank xyz = new XYZBank();
        xyz.openAccountProcess();
        xyz.closeAccountProcess();

    }

}
```

Create a object of **XYZBank & ABCBank** and invoke the methods.

The appropriate bank classes **openAccountProcess** method will be trigerred.
closeAccountProcess defined in base RBIBank lcass will be invoked.