Lesson 8.1

Abstraction

By the end of this lesson you will learn:

- Introduction to Abstraction
- Using Abstract Class
- Using Abstract Method
- Final Keyword

What is Abstraction

It means ‘Hiding the Details’. It is a process where implementation is hidden and its functionality is shown to user.

When you go to atm booth. You use the machine to withdraw your money. But you do not know how the machine is actually working. So it is hiding its details from the user.

Another example is when you use your tv remote you do not know how the button inside the remote is actually working.

So all these are example of abstraction.

In programming, this concept of hiding the details is known as abstraction.

Abstract class and abstract method:

We can hide the details of a class or a method.

- Abstract Class
- Abstract Method
- **Abstract Class**: An abstract class is a class that we can not instantiate (can not create object). However, we can use Object Reference of an abstract class.
  The keyword abstract is used to denote abstract class.

public abstract class MyClass{. . .}

An abstract class may have attributes. An abstract class may have constructors but we can not call the constructors using the new keyword.

However, super( ), super(…), this( ), this(….) can be used to call constructors of an abstract class.

An abstract class may have regular methods. An abstract class may have abstract methods. It is not necessary that an abstract class must have an abstract method.

So, it possible to have both regular methods and abstract methods in any number combination in a abstract class.

An abstract class MUST have a child class. This child class may be a regular class or may be another abstract class. If the child class is another abstract class, that child class will have another child class which has to be a regular class.

So, at least one regular class will inherit the abstract class/classes in one way or the other.

Abstract Method: An abstract method is like a regular method but it does not have anybody. An abstract method is denoted with the keyword abstract.

public abstract void show( );

As, an abstract method does not have any body, it does not have the '{ }'. Instead, it has a ' ; ' .

An abstract method must be inside an abstract class. As, an abstract class must have a child class, that child class should override/implement (give body to) all the abstract methods (if any) of that abstract class. If it does not override/implement one/any of the abstract methods, the child class will have to be declared as another abstract class.

So from above discussion we can write the following:

```
public abstract class Alpha
{
 public void show( ){. . . }
 public void print( ){. . . }
}
public class MyAlpha extends Alpha
{
public void display( ){. . . }
}
```

```
public abstract class Bravo
{
public abstract void show( );
public void print( ){. . . }
}
public class MyBravo extends Bravo
{
public void display( ){. . . }
public void show( ){. . .}
}
```

- ✓ In the Alpha class there is no abstract method. So in the child class "MyAlpha" method overriding/implementation is not mandatory.
- ✓ In the Bravo class there is a abstract method named "show". So in the child class it must be overridden.

But we can not write the following:

```
public abstract class Charlie{
    public abstract void show( );
    public abstract void print( );
}

public class MyCharlie extends Charlie{
     public void display( ){. . .}
     public void show( ){. . . }
}
```

- ✓ In the "Charlie" class there are two abstract method. So the child class "MyCharlie" must implement both the method.
- ✓ But in this case only show is implemented in the child class.

We can also write the following:

public abstract class Alpha{
public void show( ){. . . }
public void print( ){. . . }
}
	public class MyAlpha extends Alpha{
	public void display( ){. . . }
	}

public abstract class Bravo{
public abstract void show( );
public void print( ){. . . }
}
 	
public abstract class Charlie{
	public class MyBravo extends Bravo{
	public void display( ){. . . }
	public void show( ){. . .}
	}

public abstract void show( );
public abstract void print( );
}
	public abstract class MyCharlie extends Charlie{
	public void display( ){. . .}
	public void show( ){. . . }
	}

Exercise

- ✓ Design the following classes
- ✓ You have to use super in every child class parameterized constructor.