OOPs/ OOPS principles/ oops paradigm

* OOPS in java
* OOP: Object oriented programming
* OOPS: Object oriented programming system
* But calling oop is not good so we are calling as oops.
* In imperative programming oop is better than procedural, structural programming.
* Because in oops we are using code reusability.

Programming languages that follow oop?

* Java
* C++
* C#
* Python
* Simula
* Smalltalk.

What are the principles of oops?

* Inheritance

Definition:

* Inheritance is a mechanism in which one class all properties and behaviours are used in another class but there should be specific relationship.
* To use code reusability we introduce an inheritance concept.

Code-reusability:

* We are using one class code into another class is called code re usability.
* Here parent properties and behaviour inherit by child called inheritance.

Types of inheritance

They are two types of inheritance:

1. Multi-level inheritance.

* In java only multi-level inheritance applicable in class level.
* Use Multi level inheritance in class level only by using keyword "extends". Means we are extending code from parent/super class to child/sub class.

2. Multiple inheritance

* In java multiple inheritance won't be applicable in class level.
* Multiple inheritance applicable only in interface level.
* Encapsulation(getters/setters)

Definition:

* Describes the ability of an object and hide it data and methods from the rest of the world.
  + - Or
* Binding the variables and methods into single entity.
* In class we are declaring variables as private access modifier we can't assign and use them in another class.
* So that’s why we follow getters and setters method.
* When we want to assign a value to fields follows set.
* We can override for variables value.

Syntax:

Public void setVairableName ()

{

this.variableName = vairableName;

}

* If we want to use this method anywhere follows getters.
* If we need to print only particular values we use getters

Syntax:

public datatype getvaraibleName ()

{

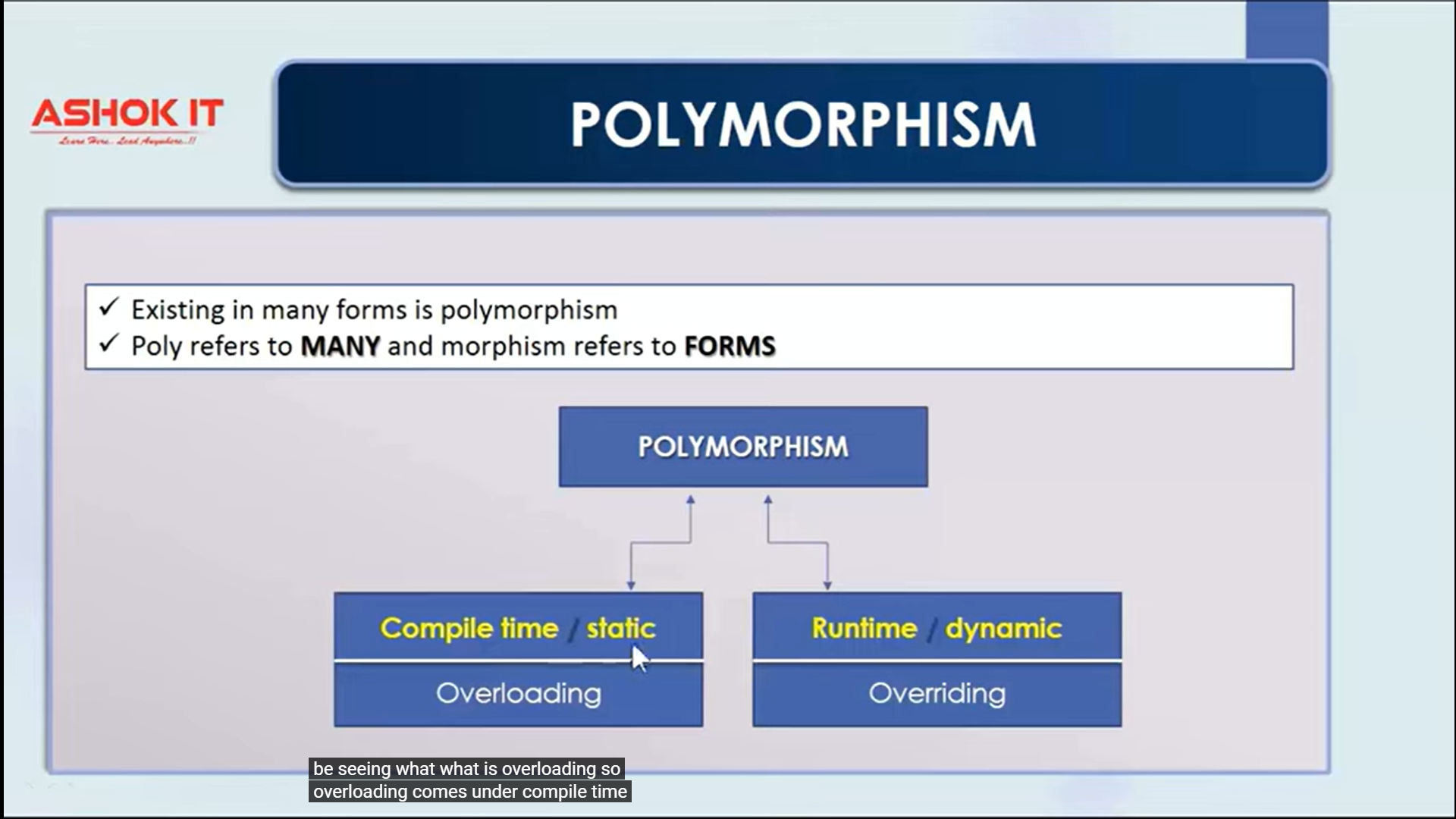
//If we want perform extra operations we can perform.

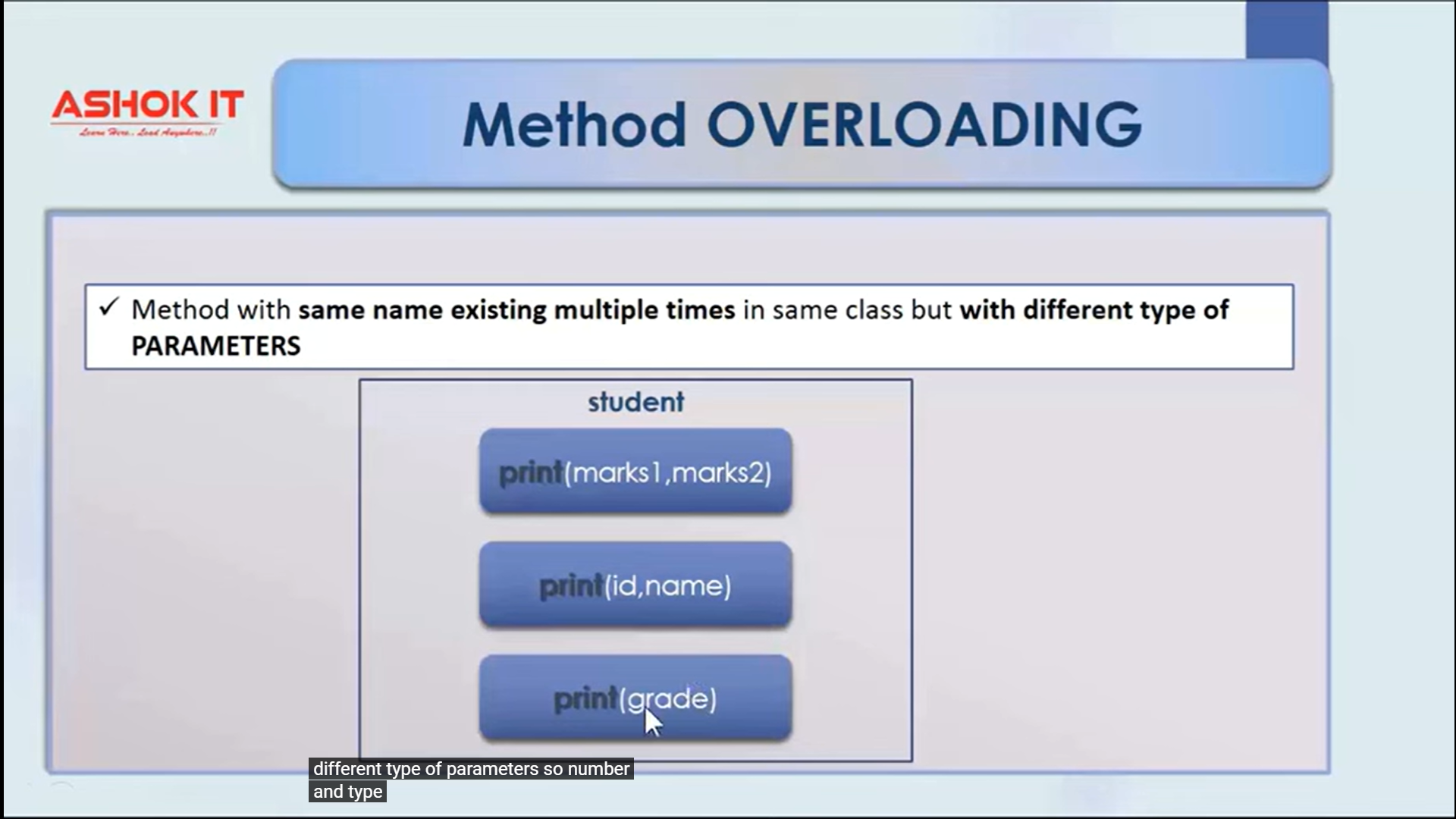
return variableName;

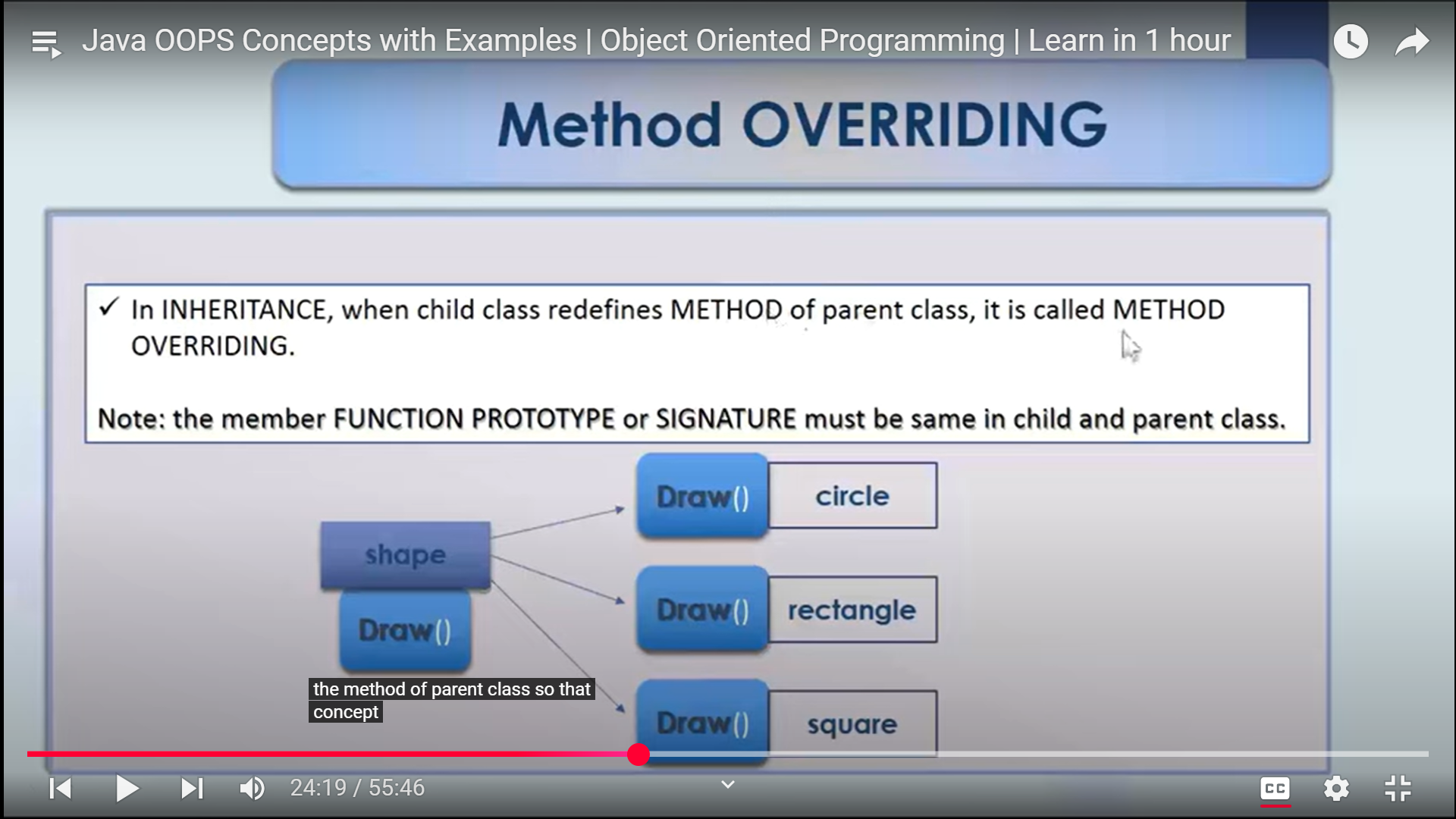
}

* We are securing the data by using methods.
* For data we are not giving access through objects

Polymorphism







* In Greek they is two words called poly, morph by combining both we get polymorphism.
* Poly: more than one or multiple.
* Morph: form
* Polymorphism is called more than one form.

Def.:

* Polymorphism is the ability of an object to take on many forms.
* In polymorphism we are achieving both method overloading and method overriding concepts.

There are two types of polymorphism

* Compile - time polymorphism
* It is a type of polymorphism which can be achieved at java compilation time.
* It is also referred as static polymorphism or early binding.
* To achieve compile time polymorphism we are using "method overloading" concept in java.

Method overloading:

* E.g.
* In class if we want to give two methods with same name java doesn't accept.
* To avoid that in parameters we can give different type of parameters/ number of parameters/ order of parameters.
* Here parameter name is not problem but parameter type is consider.

Method -1

public void Add (int i, int j)

{

//Statements;

}

Method-2

public void Add (int i, int j, int z)

{

//Statements;

}

* Runtime polymorphism.
* It is a type of polymorphism which can be achieved at java run time or execution time.
* It is also called as dynamic polymorphism or dynamic method dispatch.
* To achieve runtime polymorphism we are using "method overriding" concept in java.
* E.g.
* In class we written a code and that class extends in another class. In extends class with same method name we are creating is called method overriding.

Annotation: @Override keyword.

* In method if we declare with annotation it will check in parent class whether same method name is present or not.
* It is used to correct the user.

Abstraction.

Definition:

* It is process of hiding the implementation details and showing only functionality to the user.
* If we want to achieve abstraction concept based on interfaces and abstract classes we will do.
* And it is also used for two reason

1. Security purpose

* E.g.
* For example take an ATM machine in withdraw option if we leak background implementation user can collect more amount.
* To avoid those we hiding implementation by using abstraction concept.

2. Simplicity

* For user we always explain functionality, if we say about implementation user may get confuse and it is not necessary to explain about implementation to user.
* E.g.
* For user about withdraw button we can simply tell keep ATM card -->click withdraw

Button-->enter password -->and withdraw amount. This is fine.

* But explain about entire implementation is not necessary.

Super keyword:

* Is used to access parent properties.
* If we want to take specific method/variable from super class to child class than we use keyword super.
* If we pointing method/variable first it give preference or check in that class and if any related method or variable not present than goes to class and there it will check any parent class is present or not.
* If present it will check in that parent class.
* To avoid that above operation we use keyword super.

Interview question

1. Difference between object oriented programming and object based programming?

* In object based objects will be there but there is no inherit.
* If one object properties can inherit into another object called object oriented.
* If one object properties can't inherit into another object called object based programming.
* E.g. java is oops.

VB (visual basic programming language) is object based programming language.

2. Is java is 100% oops?

Ans:

* No, because in java for primitive data types, static keywords we are not creating objects. So java is not 100% oops.

3. Difference between class and interface in inheritance concept?

* In class for every method we provide implementation.
* Class doesn’t support multiple inheritance.
* In interface class for methods we won't provide any implementation.
* And in interface java support both multi-level and multiple inheritance.
* We can also use interfaces in class level by using keyword "implements".
* In that class we need to implement those UN implemented methods.

4. Best example for method overloading?

Ans:

* System.out.print is best example in java.

5. By using interface how much % we can achieve?

Ans:

* Before java 8 we can achieve 100% of abstraction.
* After java 8 100% abstraction may be may not be possible. It depend on user (who is design interface).

6. By using abstract how much % we can achieve?

Ans:

* We can't achieve 100% of abstraction.

7. Difference between abstraction and encapsulation?

Ans:

* In encapsulation we are just hiding the data. But in abstraction we are hiding implementation and giving functionality.