

#### SNS COLLEGE OF TECHNOLOGY



# Coimbatore-35. An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade (Cycle III)
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
COURSE CODE & NAME: 23CSB201 - Object Oriented Programming

II YEAR/ III SEMESTER

**UNIT – I INTRODUCTION TO OOP** 

**Topic: Abstraction** 





- Abstraction in Java is the process in which we only show essential details/functionality to the user.
- The non-essential implementation details are not displayed to the user.
- In Java, abstraction is achieved by interfaces and abstract classes.
- We can achieve abstraction using interfaces.
- Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.
- The properties and behaviours of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.





#### Java Abstract classes and Java Abstract methods

- 1. An abstract class is a class that is declared with an abstract keyword.
- 2. An abstract method is a method that is declared without implementation.
- 3. An abstract class may or may not have all abstract methods. Some of them can be concrete methods
- 4. A method-defined abstract must always be redefined in the subclass, thus making overriding compulsory or making the subclass itself abstract.
- 5. Any class that contains one or more abstract methods must also be declared with an abstract keyword.
- 6. There can be no object of an abstract class. That is, an abstract class can not be directly instantiated with the <u>new operator</u>.
- 7. An abstract class can have parameterized constructors and the default constructor is always present in an abstract class.





#### When to use abstract classes and abstract methods?

- There are situations in which we will want to define a superclass that declares the structure of a given abstraction without providing a complete implementation of every method.
- Sometimes we will want to create a superclass that only defines a generalization form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.
- Consider a classic "shape" example, perhaps used in a computer-aided design system or game simulation.
- The base type is "shape" and each shape has a color, size, and so on. From this, specific types of shapes are derived(inherited)-circle, square, triangle, and so on each of which may have additional characteristics and behaviors.
- For example, certain shapes can be flipped.
- Some behaviors may be different, such as when you want to calculate the area of a shape.
- The type hierarchy embodies both the similarities and differences between the shapes.



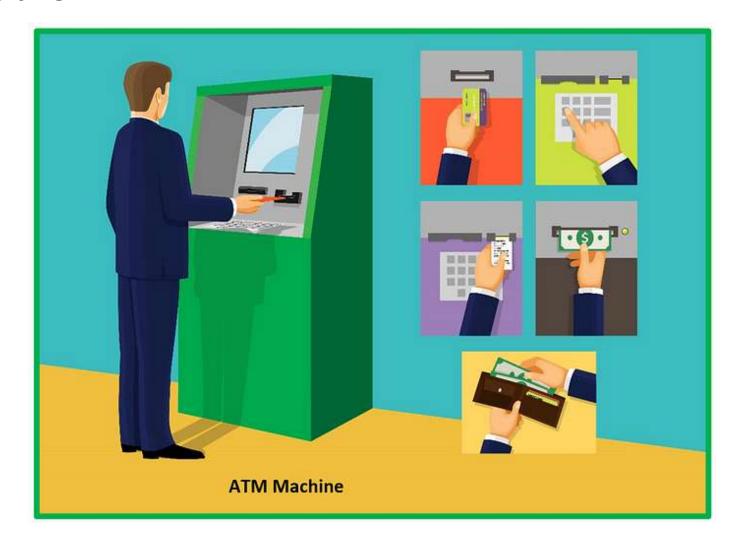


#### Example 1:

- Let's take the ATM machine. In an ATM machine, we can perform functions like withdraw cash, deposit cash, check balance, print bills, and so on.
- Even though it performs a lot of actions it doesn't show us the process.
- It has hidden its process by showing only the main things like getting inputs and giving the output.











#### Example 2:

- The next example is the most commonly used mobile phones. On mobile phone, we can perform so many actions like making a call, sending messages, take pictures, download software and etc.
- We perform a lot of things but here also we don't know the inside process of these things.
- Which means the implementation parts are hidden.











