Abstraction

Learn to hide the data with the abstraction technique in OOP.

We'll cover the following

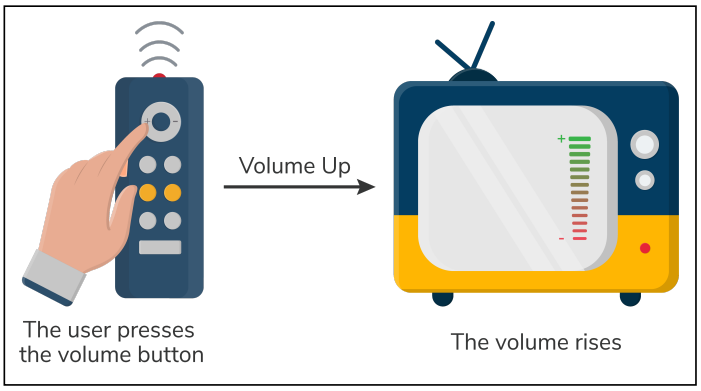
Definition

Abstraction is a technique used in object-oriented programming that simplifies the program's structure. It focuses only on revealing the necessary details of a system and hiding irrelevant information to minimize its complexity. In simpler words, we can say that it means to show what an object does and how it hides.

Example

There are countless real-life examples that follow the rules of abstraction. Take the "volume" button on a television remote. With one click, we can increase the TV's volume. Let's say the button calls the `volumeUp()` function. The TV responds with a sound louder than before. We are oblivious to the fact that the inner circuitry of the TV implements this, but we know how the exposed function interacts with the TV's volume.

Another instance of abstraction is our daily use of vehicles. To our general knowledge, the race peddle tells the car to consume fuel and increase its speed. We do not need to understand the mechanical process.



Abstraction in the form of the "volume up" button

Implementation of abstraction in programming languages

So, let's put all this theory into practice. In the code below, we have a basic class of a circle:

```
class Circle {
  #radius;
  #pi;
  constructor() {
    this.radius = null;
    this.pi = null;
  }
}
```

It has two variables, `radius` and `pi`. Now let's add the constructor and functions to calculate the area and perimeter:

```
class Circle {
  //define data attributes
  private double radius;
  private double pi;

  //define constructors
  public Circle() {
    radius = 0;
    pi = 3.142;
  }

  public Circle(double r) {
    radius = r;
    pi = 3.142;
  }

  //define methods
  public double area() {
    return pi * radius * radius;
  }

  public double perimeter() {
    return 2 * pi * radius;
  }

  public static void main(String[] args) {
    Circle circle = new Circle(5);
    System.out.printf("Area: %.2f %n", circle.area());
    System.out.printf("Perimeter: %.2f %n", circle.perimeter());
  }
```

Abstraction implementation in programming languages

As you can see, we only need to define the radius of the circle in the constructor. After that, the `area()` and `perimeter()` functions are available to us. This interface is part of encapsulation.

We use the functions to calculate the area and perimeter. Users do not need to know the implementation details of the functions. Even `pi` is hidden since it's a constant. This is how we can achieve abstraction using classes.

Advantages of abstraction

The following are some advantages of abstraction:

- It reduces the complexity of the system from a user's perspective.
- It makes the code extendable and reusable.
- It refines the modularity of the application or the system.
- It makes the code more maintainable.

Abstraction vs. encapsulation

Since abstraction and encapsulation are data hiding techniques of OOP, they are often confused with being the same. Let's look at some of the differences in the following table:

Abstraction	Encapsulation
It focuses on the design level of the system.	It focuses on the application level of the system.
It hides unnecessary data to simplify the structure.	It restricts access to data to prevent its misuse.
It highlights the work that the object performs.	It deals with the internal working of the object.
Abstraction means to hide implementation using interface and abstract classes.	Encapsulation means to hide data using getter and setter functions.

Next, let's look at another important principle of object-oriented programming—inheritance.

← Back
Encapsulation

Next →
Inheritance

Mark as Completed