



Abstraction in Java



gaurav miglani

Read

Discuss

Courses

Practice

Video

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essential units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.

What is Abstraction in Java?

In Java, abstraction is achieved by [interfaces](#) and [abstract classes](#). We can achieve 100% 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 behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

Real-Life Example:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of a car or applying brakes will stop the car, but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.



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.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

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 [new operator](#).
7. An abstract class can have parameterized constructors and the default constructor is always present in an abstract class.

Algorithm to implement abstraction in Java

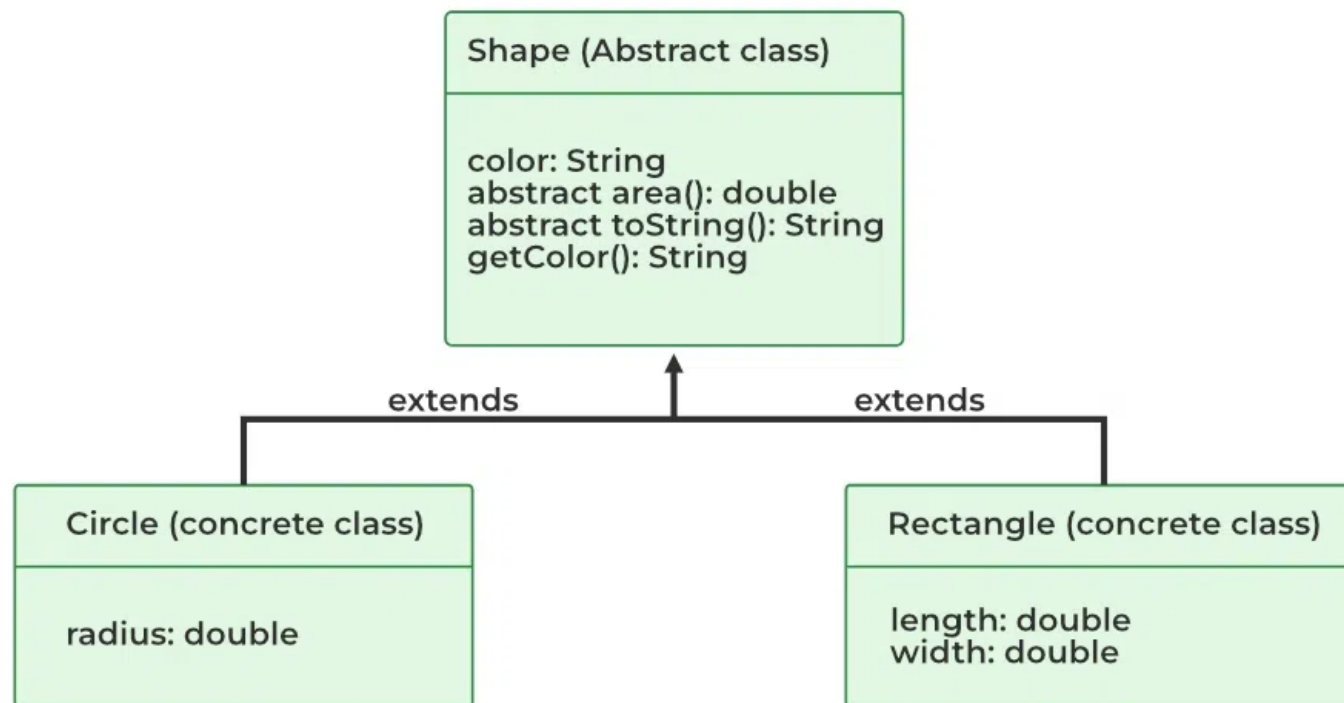
1. Determine the classes or interfaces that will be part of the abstraction.
2. Create an abstract class or interface that defines the common behaviors and properties of these classes.
3. Define abstract methods within the abstract class or interface that do not have any implementation details.
4. Implement concrete classes that extend the abstract class or implement the interface.
5. Override the abstract methods in the concrete classes to provide their specific implementations.
6. Use the concrete classes to implement the program logic.

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.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

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.



Java Abstraction Example

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Java

```
// Java program to illustrate the
// concept of Abstraction
abstract class Shape {
    String color;

    // these are abstract methods
    abstract double area();
    public abstract String toString();

    // abstract class can have the constructor
    public Shape(String color)
    {
        System.out.println("Shape constructor called");
        this.color = color;
    }

    // this is a concrete method
    public String getColor() { return color; }
}

class Circle extends Shape {
    double radius;

    public Circle(String color, double radius)
    {

        // calling Shape constructor
        super(color);
        System.out.println("Circle constructor called");
    }
}
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
@Override double area()
{
    return Math.PI * Math.pow(radius, 2);
}

@Override public String toString()
{
    return "Circle color is " + super.getColor()
        + "and area is : " + area();
}
}

class Rectangle extends Shape {

    double length;
    double width;

    public Rectangle(String color, double length,
                     double width)
    {
        // calling Shape constructor
        super(color);
        System.out.println("Rectangle constructor called");
        this.length = length;
        this.width = width;
    }

    @Override double area() { return length * width; }

    @Override public String toString()
    {
        return "Rectangle color is " + super.getColor()
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
public class Test {  
    public static void main(String[] args)  
    {  
        Shape s1 = new Circle("Red", 2.2);  
        Shape s2 = new Rectangle("Yellow", 2, 4);  
  
        System.out.println(s1.toString());  
        System.out.println(s2.toString());  
    }  
}
```

Output

```
Shape constructor called  
Circle constructor called  
Shape constructor called  
Rectangle constructor called  
Circle color is Redand area is : 15.205308443374602  
Rectangle color is Yellowand area is : 8.0
```

Example 2:

Java

```
// Java Program to implement  
// Java Abstraction  
  
// Abstract Class declared  
- - - - -
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
public Animal(String name) {
    this.name = name;
}

public abstract void makeSound();

public String getName() {
    return name;
}
}

// Abstracted class
class Dog extends Animal {
    public Dog(String name) {
        super(name);
    }

    public void makeSound()
    {
        System.out.println(getName() + " barks");
    }
}

// Abstracted class
class Cat extends Animal {
    public Cat(String name) {
        super(name);
    }

    public void makeSound()
    {
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).


```
// Driver Class
public class AbstractionExample {
    // Main Function
    public static void main(String[] args)
    {
        Animal myDog = new Dog("Buddy");
        Animal myCat = new Cat("Fluffy");

        myDog.makeSound();
        myCat.makeSound();
    }
}
```

Output

Buddy barks


Fluffy meows

Explanation of the above program:

This code defines an Animal abstract class with an abstract method makeSound(). The Dog and Cat classes extend Animal and implement the makeSound() method. The main() method creates instances of Dog and Cat and calls the makeSound() method on them.

This demonstrates the abstraction concept in Java, where we define a template for a class (in this case Animal), but leave the implementation of certain methods to be defined by subclasses (in this case makeSound()).

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).



**Virtual Interaction
Made Real**

One platform for seamless voice, video
and live streaming integration

[LEARN MORE >](#)

ZEGOCLOUD

**Better Outcomes Start with Better
Communication**

[Java Arrays](#)[Java Strings](#)[Java OOPs](#)[Java Collection](#)[Java 8 Tutorial](#)[Java Multithreading](#)[Java Exception Handling](#)[Java Programs](#)[Java Project](#)

Why do we use abstract?

In Java, the abstract keyword is used to declare a class or a method as an abstract. An abstract class cannot be instantiated, meaning we cannot create objects of it, but we can use it as a template for creating other classes that extend it. Abstract methods, on the other hand, are declared in abstract classes, but they have no implementation. The implementation is left for the concrete subclasses to define.

Here are some reasons why we use abstract in Java:

1. Abstraction:

Abstract classes are used to define a generic template for other classes to follow. They define a set of rules and guidelines that their subclasses must follow. By providing an abstract class, we can ensure that the classes that extend it have a consistent structure and behavior. This makes the code more organized and

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

3. Polymorphism:

Abstract classes and methods enable polymorphism in Java. Polymorphism is the ability of an object to take on many forms. This means that a variable of an abstract type can hold objects of any concrete subclass of that abstract class. This makes the code more flexible and adaptable to different situations.

3. Frameworks and APIs:

Java has numerous frameworks and APIs that use abstract classes. By using abstract classes developers can save time and effort by building on existing code and focusing on the aspects specific to their applications.

In summary, the abstract keyword is used to provide a generic template for other classes to follow. It is used to enforce consistency, inheritance, polymorphism, and encapsulation in Java.

Encapsulation vs Data Abstraction

1. [Encapsulation](#) is data hiding (information hiding) while Abstraction is detailed hiding (implementation hiding).
2. While encapsulation groups together data and methods that act upon the data, data abstraction deal with exposing the interface to the user and hiding the details of implementation.
3. Encapsulated classes are Java classes that follow data hiding and abstraction. We can implement abstraction by using abstract classes and interfaces.
4. Encapsulation is a procedure that takes place at the implementation level, while abstraction is a design-level process.

Advantages of Abstraction

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

3. Helps to increase the security of an application or program as only essential details are provided to the user.
4. It improves the maintainability of the application.
5. It improves the modularity of the application.
6. The enhancement will become very easy because without affecting end-users we can able to perform any type of changes in our internal system.
7. Improves code reusability and maintainability.
8. Hides implementation details and exposes only relevant information.
9. Provides a clear and simple interface to the user.
10. Increases security by preventing access to internal class details.
11. Supports modularity, as complex systems can be divided into smaller and more manageable parts.
12. Abstraction provides a way to hide the complexity of implementation details from the user, making it easier to understand and use.
13. Abstraction allows for flexibility in the implementation of a program, as changes to the underlying implementation details can be made without affecting the user-facing interface.
14. Abstraction enables modularity and separation of concerns, making code more maintainable and easier to debug.

Disadvantages of Abstraction in Java:

1. Abstraction can make it more difficult to understand how the system works.
2. It can lead to increased complexity, especially if not used properly.
3. This may limit the flexibility of the implementation

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

4. Abstraction can add unnecessary complexity to code if not used appropriately, leading to increased development time and effort.
5. Abstraction can make it harder to debug and understand code, particularly for those unfamiliar with the abstraction layers and implementation details.
6. Overuse of abstraction can result in decreased performance due to the additional layers of code and indirection.

A real-life example of data abstraction:

A real-life example of data abstraction is a car's dashboard. The dashboard provides a simplified interface to the driver, abstracting away the complexity of the car's internal systems.

The dashboard provides key information such as the speed, fuel level, and engine temperature, which are necessary for the driver to operate the car safely. The driver does not need to know the intricate details of how the car's engine or transmission works in order to drive the car. Instead, the dashboard presents a simplified, high-level view of the car's status, allowing the driver to focus on the task of driving.

In this example, the car's internal systems represent the underlying implementation details that are hidden from the driver. The dashboard provides the abstraction layer that simplifies the interface between the driver and the car, making it easier to operate. The abstraction layer also provides flexibility in the implementation of the car's internal systems, allowing changes to be made without affecting the driver's experience.

References:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

2. Java Design Patterns: Abstraction: <https://java-design-patterns.com/patterns/abstraction/>
3. Head First Design Patterns: Abstraction: <http://shop.oreilly.com/product/9780596007126.do>

Related articles

- [Interfaces in java](#)
- [Abstract classes in Java](#)
- [Difference between abstract class and interface](#)
- [abstract keyword in Java](#)

Last Updated : 20 May, 2023

290

Similar Reads

1. Control Abstraction in Java with Examples
2. Abstraction by Parameterization and Specification in Java
3. Difference Between Data Hiding and Abstraction in Java
4. Difference between Abstraction and Encapsulation in Java with Examples

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

5. Understanding OOPs and Abstraction using Real World Scenario

6. Understanding Encapsulation, Inheritance, Polymorphism, Abstraction in OOPs

7. Difference Between java.sql.Time, java.sql.Timestamp and java.sql.Date in Java

8. How to Convert java.sql.Date to java.util.Date in Java?

9. Different Ways to Convert java.util.Date to java.time.LocalDate in Java

10. How to Convert java.util.Date to java.sql.Date in Java?

Related Tutorials

1. Spring MVC Tutorial

2. Spring Tutorial

3. Spring Boot Tutorial

4. Java 8 Tutorial

5. Java IO Tutorial

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[Previous](#)[Next](#)

Inheritance in Java

Encapsulation in Java

Article Contributed By :



gaurav miglani
gaurav miglani

Vote for difficulty

Current difficulty : [Easy](#)

[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

Improved By : [vaibhav jain 20](#), [gabaa406](#), [sg4ipiafwot258z3lh6xa2mjq2qtxd89f49zgt7g](#), [anishcode19](#), [avinashrat55252](#), [anikettchpiow](#)

Article Tags : [Java-Abstract Class and Interface](#), [Java-Object Oriented](#), [Java](#)

Practice Tags : [Java](#)

[Improve Article](#)[Report Issue](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305

feedback@geeksforgeeks.org

Company

[About Us](#)

[Careers](#)

[In Media](#)

[Contact Us](#)

[Terms and Conditions](#)

[Privacy Policy](#)

[Copyright Policy](#)

[Third-Party Copyright Notices](#)

[Advertise with us](#)

Languages

[Python](#)

[Java](#)

[C++](#)

Explore

[Job Fair For Students](#)

[POTD: Revamped](#)

[Python Backend LIVE](#)

[Android App Development](#)

[DevOps LIVE](#)

[DSA in JavaScript](#)

Data Structures

[Array](#)

[String](#)

[Linked List](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[R Language](#)[Android Tutorial](#)

Algorithms

[Sorting](#)[Searching](#)[Greedy](#)[Dynamic Programming](#)[Pattern Searching](#)[Recursion](#)[Backtracking](#)

Computer Science

[GATE CS Notes](#)[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)

Data Science & ML

[Tree](#)[Graph](#)

Web Development

[HTML](#)[CSS](#)[JavaScript](#)[Bootstrap](#)[ReactJS](#)[AngularJS](#)[NodeJS](#)

Python

[Python Programming Examples](#)[Django Tutorial](#)[Python Projects](#)[Python Tkinter](#)[OpenCV Python Tutorial](#)[Python Interview Question](#)

DevOps

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

Top 50 Graph Problems

Top 50 Array Problems

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

Interview Corner

Company Preparation

Preparation for SDE

Company Interview Corner

Experienced Interview

Internship Interview

Docker

Kubernetes

Azure

GCP

System Design

What is System Design

Monolithic and Distributed SD

Scalability in SD

Databases in SD

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Commerce

Accountancy
Business Studies
Microeconomics
Macroeconomics
Statistics for Economics
Indian Economic Development

SSC/ BANKING

SSC CGL Syllabus
SBI PO Syllabus
SBI Clerk Syllabus
IBPS PO Syllabus
IBPS Clerk Syllabus
Aptitude Questions
SSC CGL Practice Papers

UPSC

Polity Notes
Geography Notes
History Notes
Science and Technology Notes
Economics Notes
Important Topics in Ethics
UPSC Previous Year Papers

Write & Earn

Write an Article
Improve an Article
Pick Topics to Write
Write Interview Experience
Internships
Video Internship

@geeksforgeeks , Some rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).