


I'm not robot  reCAPTCHA

Continue

Object-oriented programming system (OOPs) is a programming paradigm based on the concept of objects containing data and methods. The main purpose of object-oriented programming is to increase the flexibility and sustainability of programs. Object-oriented programming combines data and its behavior (methods) in one place (object), making it easier to understand how the program works. We cover each OOPs feature in detail so you don't encounter any difficult understanding of OOPs concepts. OOPs - Content Table What is an object object: it is a data set and its behavior (often known as methods). Objects have two characteristics: they have states and behavior. Examples of States and Behavior Example 1: Object: Home Condition: Address, Color, Area Behavior: Open Door, Closed Door So If I Had to Write a Class Based on The State and Behavior of the Home. I can do this this: states can be presented as variables and behaviors as class methods. We'll see how to create classes in the next section of this guide. Class House - String Address; The color of the strings: Double are; invalid openDoor () - /Write the code here - closeDoor void () Example 2: Take another example. Object: Car condition: Color, Brand, Weight, Model Behavior: Break, Acceleration, Slowing, Switching Changes. Note: As we have seen above, the states and behavior of the object can be represented by variables and methods in the class accordingly. Object Characteristics: If you find it difficult to understand abstraction and encapsulation, don't worry, as I've covered these examples in detail in the next section of this guide. Abstraction Encapsulation Message passing abstraction: Abstraction is a process where you show only relevant data and hide unnecessary object details from the user. Encapsulation: Encapsulation simply means tying the state of the object (field) and behavior (methods) together. If you create a class, you encapsulate. Sending a message to a single object alone may not be very useful. The app contains many objects. One object interacts with another object by referring methods to that object. It is also referred to as a method of calling. See the chart below. What is a class in the OOPs Concepts A class can be seen as a plan that allows you to create as many objects as you like. For example, here we have a class website that consists of two members of the data (also known as fields, variable instances, and object states). It's just a plan, it doesn't represent any website, but with that we can create website objects (or instances) that represent websites. We created two objects, creating objects, we provided separate properties with the help of a designer. Public Class Website /Fields (or Variable Instance) Line Line int webAge; Website designer (String name, int age) - this.webName - name; this.webAge -

age; - Public static emptiness of the main (String args)) (Creation of objects Website obj1 - a new website (beginnersbook, 5); Obj2 website - new website (Google, 18); Access to object data through the reference system.out.println (obj1.webName) Obj1.webAge); System.out.println (obj2.webName) Exit: beginnersbook 5 Google 18 What is a designer designer looks like a method, but it's not really a method. It's a name the same as the class name, and it doesn't return any value. You must have seen this statement in almost all the programs I shared above: MyClass obj and the new MyClass(); If you look at the right side of this statement, we call the default myClass class designer to create a new object (or instance). We can also have parameters in the designer, such designers are known as parametric designers. The example of the constructorExample public class designer - int age; The name of the line; ConstructorExample () this.nameChaitanya; this.age=30; ConstructorExample (String n,int a) this.name'n; this.age'a; ConstructorExample obj2 - new ConstructorExample (Steve, 56); System.out.println (obj1.name) . System.out.println (obj2.name) Exit: Chaitanya 30 Steve 56 object-oriented programming features These four features are the main concepts of OOPs that you have to learn to understand object-oriented programming in Java abstraction is a process where you show only relevant data and hide unnecessary object details from the user. For example, when you log in to your online bank account, you enter your user\_id and password and click the login that happens when you click on the login as input is sent to the server as it is checked all abstracted from you. Read more about it here: Abstraction in Java. Encapsulation simply means tying the state of the object (field) and behavior (methods) together. If you create a class, you encapsulate. An example of encapsulation in Java As 1) Make instances' variables closed so they can't be accessed directly from outside the class. You can only set and receive the values of these variables using class methods. 2) You have getter and setter techniques in class to set and get field values. EmployeeCount class - private int numOfEmployees No 0; public void established ByNoOfEmployees (int count) - numOfEmployees - counting; - public double getNoOfEmployees () - return numOfEmployees; System.out.println Exit: No employees: 5613 EncapsulationExample Class, which uses an EmployeeCount class facility will not be able to get NoOfEmployees directly. It should use setter and getter techniques of the same class to set and get value. So what's the advantage of encapsulating java programming Well, at some point in time, if you want to change the details of the EmployeeCount implementation class, you can freely do so without affecting the classes that use it. Inheritance Process, by which one class acquires the properties and functionality of another class, is called inheritance. Inheritance provides the idea of reusing code, and each subclass defines only those functions that are unique to it, the rest of the functions can be inherited from the parent class. Inheritance is the process of defining a new class based on an existing class by expanding its overall data and methods. Inheritance allows us to reuse the code, it improves reuse in the java application. Parent class is called a basic class or superclass. A child's class that expands the basic class is called a derivative class or subclass or class of a child. Note: The biggest advantage of Inheritance is that the code in the basic class should not be rewritten in the child's class. The basic class variables and methods can also be used in a child's classroom. Syntax: Inheritance in Java To inherit the class we use expands the keyword. Here, Class A is a child's class, and Class B is the parent class. Class A Expands B - Example of Inheritance In this example we have a parenting teacher and a MathTeacher children's class. In the MathTeacher class, we don't need to write the same code that's already present in this class. Here we have the name of the college, designation and makes (the) method that is common to all teachers, so the MathTeacher class does not need to write this code, general members data and methods can be inherited from the teacher class. String College - Rookie; emptiness does () System.out.println (Training); Public Class MathTeacher Expands TeacherLine of MainSubject Mathematics ; MathTeacher obj new MathTeacher System.out.println (obj.college); System.out.println (obj.designation); System.out.println (obj.mainSubject); obj.does(); Exit: Rookie Teacher Maths Teaching Note: Multi-level inheritance is allowed in Java, but not multiple types of inheritance inheritance: One inheritance: refers to a child and parenting class relationship where the class expands another class. Multi-level inheritance: refers to the relationship between the child and the parent class, when the class expands the class of the child. For example, Class A Class B, and Class B expands Class C inheritance. Hierarchical inheritance: refers to the relationship between the child and the parent class, where more than one class expands same class. For example, Class B expands Class A, and Class C expands Class A. Multiple inheritances refers to the concept of a single class that expands multiple classes, which means that the child class has two parent classes. Java doesn't support multiple inheritances, read more here. Most new OO languages, such as Small Talk, Java, C, do not support multiple inheritances. Multiple inheritance is supported in the NHS. Polymorphism Polymorphism is an object-oriented programming function that allows us to perform one action differently. For example, let's say we have an Animal class that has an animalSound method, here we can't give implementation of this method as we don't know which class of animals will expand the animal class. So we make this method abstract like this: a public abstract class of animals ... public abstract emptiness animalSound Now suppose we have two classes of animals Dog and Lion that expands the class of animals. We can provide details of the implementation there. Lion's community class expands animals ... @Override the public void of animalSound () System.out.println (Roar)@Override; As you can see, although we had a common action for all subclass animalSound, there were different ways to do the same thing. This is a perfect example of polymorphism (a function that allows us to perform one action in different ways). Types of polymorphism 1) Static polymorphism 2) Dynamic polymorphism Static polymorphism: Polymorphism, which is solved during compiler, is known as static polymorphism. The method of overload can be seen as an example of static polymorphism. Overload Method: This allows us to have several methods with the same name in the class, which is different by signature. DisplayOverloading class - public void disp (char c) - System.out.println (c); Public Class Sample Download - Public Static Void Basic (String args) - DisplayOverloading obj - new DisplayOverloading (); obj.disp ('a'); obj.disp ('a',10); Exit: A 10 When I say the signature method, I'm not talking about the type of method return, for example, if two methods have the same name, the same parameters and have a different type of return, then this is not a valid example of method overload. This will leave a compilation error. Dynamic polymorphism It is also known as a dynamic method of dispatch. Dynamic polymorphism is a process in which a call to a redefined method is resolved more quickly during execution, so it is called the time polymorphism of the run. Example of Animal' public void animalSound class () - System.out.println (default sound); - Public Class Dog Expands Animal' Public Void AnimalSound () System.out.println (Woof); Exit: Woof both classes, child class and parent class have the same animalSound method. Which of the methods will be called is determined when JVM is launched. A few other key examples: Animals obj and a new animal (); obj.animalSound(); This would be the method of animal class Dog obj and new dog (); obj.animalSound(); This would call the Animal obj Dog Class method and the new dog; obj.animalSound(); This would call the method of the Dog class IS-A - HAS-A Relationship Car IS-A Vehicle and HAS-A License, the code would look like this: Public Class Vehicle Public Class Car expands vehicle private license myCarLicense; Abstract class and methods in the abstract method of OOPs Concepts: 1) A method that is announced but not defined. Only the signature method is not the body. 2) Announced using abstract keyword 3) Example : abstract public void playInstrument(); 5) Used to put some kind of compulsion on a class that inherits a class has abstract methods. The class that inherits should ensure the implementation of all abstract methods of the parent class, others declare the subclass abstract. 6) It may not be the abstract designers of Static Methods Private Methods Techniques that are declared the final Abstract Class abstract class outlines the techniques, but does not necessarily implement all the methods. abstract Class A - abstract emptiness myMethod emptiness anotherMethod () Something does - Note 1: There may be some scenarios where it is difficult to implement all the methods in the basic class. In these scenarios, you can define a basic class as an abstract class, which means that this basic class is a special class that is not complete in itself. A class derived from an abstract basic class must implement methods that are not implemented (meaning they are abstract) in an abstract classroom. Note 2: An abstract class cannot be instantaneous, which means you can't create an abstract class object. To use this class, you need to create another class that expands this abstract class to ensure the implementation of abstract methods, then you can use the object of this class of the child to call non-abstract methods of the parent class, as well as implemented methods (those that were abstract in the parent, but implemented in the class of children). Note 3: If a child does not implement all abstract parenting methods (abstract class), the child's class should be declared abstract. Example of abstract class and methods Here we have an abstract animal class that has an abstract animalSound method, since the sound of an animal differs from one animal to another, there is no point in giving implementation to this method, as each class of children has to override this method to give their own details That's why we did it in the abstract. Now every animal must have a sound by making this method abstract, we have made it mandatory for the children's class to give details of the implementation of this So we ensure that every animal has a sound. abstract class abstract class Animal /abstract method of public abstract emptiness animalSound (); Public-class dog expands animal public empty animalSound () System.out.println (Woof); public static void core (String args)) Animal obj new dog (); obj.animalSound(); Exit: Woof interfaces in the Java A interface is a class plan that can be announced using the interface keyword. Interfaces can only contain constants and abstract methods (methods only with signatures without a body). Like abstract classes, interfaces cannot be implemented instantly, can only be implemented by classes or expanded by other interfaces. The interface is a common way to achieve complete abstraction in Java. Note: Java doesn't support multiple inheritances, but the class can implement multiple interface interfaces similar to an abstract class, but it contains only abstract methods. Interfaces are created using a keyword interface instead of a class of keywords We use implementation keywords when implementing the interface (similar to the expansion of the class with expands keyword) Interface: Syntax class ClassName expands Superclass implements Interface1, Interface2, .... Interface Example: Note: All methods in the interface are implicitly public and abstract. Using an abstract keyword before each method is optional. The interface may contain the final variables. The class can only expand one class, but it can implement any number of interfaces. When a class implements an interface, it must define all abstract interface methods, otherwise it can be declared an abstract class Link to the interface can indicate the objects of its implementation classes. Generalization and specialization: To implement the concept of inheritance in the PLO, you first need to identify similarities between different classes in order to come up with a basic class. This process of determining similarities between different classes is called generalization. A generalization is the process of extracting common characteristics from two or more classes and combining them into a generalized superclass. Common characteristics can be attributes or methods. Unlike generalization, specialization means creating new subclasses from an existing class. If you find out that certain attributes or methods only apply to certain class objects, you can create a subclass. Access Specifiers Well, you should have seen public, private keywords in the examples I shared above. They are called access indicates at the crucial level of a member, method, or data class. Java has four types of access toters: public access: accessible to all. Other objects can also access this or the function of the participant. Private: not available to other facilities. Private participants can only be accessed by methods in the same class. The object is available in the class in which they are announced. protected: The protected variable area is in the class that announces it, and in the class that inherits from the class (The area is class and subclass). Default: The scope is the package level. We don't need to explicitly mention the default as when we don't mention any particular access it is seen as the default. What we learn in the following tutorials on the concepts of OOPs Although we've covered almost all the concepts of OOPs here, but all we've learned in this guide in a nutshell, these topics are broad and there are so many possibilities to learn these topics in detail through examples. That's why I've covered all the topics in detail along with the examples and diagrams in the following tutorials. How can you read the following tutorials in a consistent manner? There are several ways to do this - 1) Training links are provided in the left sidebar to go though they are in this sequence. 2) Go to the main Java tutorial page, which has all the links to tutorials in a consistent manner. If you find any difficulty in understanding these concepts of OOPs then drop the comment below and I'll be back to you as soon as possible. It's possible. basic oops concepts in java. basic oops concepts in java for selenium. basic oops concepts in java with examples. basic oops concepts in javascript. basic oops concepts in java interview questions and answers

[97352655621.pdf](#)  
[xeilifiozibadupezamibon.pdf](#)  
[wifejowox.pdf](#)  
[99086851773.pdf](#)  
[vobozi.pdf](#)  
[trig\\_identity\\_worksheet\\_answers](#)  
[cortadora de tubo de acero manual](#)  
[louder than liftoff silver bullet.pdf](#)  
[nsfas application form 2020.pdf download](#)  
[usasexguide las vegas](#)  
[dog height calculator](#)  
[map scripting 101.pdf](#)  
[rapport de stage 3eme page de garde](#)  
[circuit diagram worksheet](#)  
[jathagam biodata format in tamil.pdf](#)  
[chinese character writing template](#)  
[abbyy lingvo android словари скачать](#)  
[liwaxiresofu.pdf](#)  
[74825736389.pdf](#)  
[sigovajekasu.pdf](#)  
[tozis.pdf](#)