|  |
| --- |
| [Contribute Us](http://www.javatpoint.com/contribute-us) | [Ask Question](http://www.javatpoint.com/asknewquestion.jsp) | [login](http://www.javatpoint.com/login.jsp) [RSS Feed](http://feeds.feedburner.com/javatpointsonoo) [Subscribe to Get Email Alerts](http://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo) [Facebook Page](https://www.facebook.com/javatpoint) [Google Page](https://plus.google.com/u/0/113973042385691981314/posts) [Twitter Page](https://twitter.com/pagejavatpoint) |
| Javatpoint Logo  Top of Form    Bottom of Form  [**Subscribe Us**](http://www.javatpoint.com/subscribe.jsp)  **91-9990449935**  **0120-4256464** |

* [**Home**](http://www.javatpoint.com/)
* [**Core Java**](http://www.javatpoint.com/java-tutorial)
* [**Servlet**](http://www.javatpoint.com/servlet-tutorial)
* [**JSP**](http://www.javatpoint.com/jsp-tutorial)
* [**EJB**](http://www.javatpoint.com/ejb-tutorial)
* [**Struts2**](http://www.javatpoint.com/struts-2-tutorial)
* [**Mail**](http://www.javatpoint.com/java-mail-api-tutorial)
* [**Hibernate**](http://www.javatpoint.com/hibernate-tutorial)
* [**Spring**](http://www.javatpoint.com/spring-tutorial)
* [**Android**](http://www.javatpoint.com/android-tutorial)
* [**Design P**](http://www.javatpoint.com/design-patterns-in-java)
* [**Quiz**](http://www.javatpoint.com/examaccess)
* [**Projects**](http://www.javatpoint.com/free-java-projects)
* [**Interview Q**](http://www.javatpoint.com/corejava-interview-questions)
* [**Comment**](http://www.javatpoint.com/comment.jsp)
* [**Forum**](http://www.javatpoint.com/forum)
* [**Training**](http://www.javatpoint.com/training)

[**Basics of Java**](http://www.javatpoint.com/java-tutorial)

Java OOPs Concepts

[**Advantage of OOPs**](http://www.javatpoint.com/java-oops-concepts)[Naming Convention](http://www.javatpoint.com/java-naming-conventions)[Object and Class](http://www.javatpoint.com/object-and-class-in-java)[Method Overloading](http://www.javatpoint.com/method-overloading-in-java)[Constructor](http://www.javatpoint.com/constructor)[static keyword](http://www.javatpoint.com/static-keyword-in-java)[this keyword](http://www.javatpoint.com/this-keyword)[Inheritance(IS-A)](http://www.javatpoint.com/inheritance-in-java)[Aggregation(HAS-A)](http://www.javatpoint.com/aggregation-in-java)[Method Overriding](http://www.javatpoint.com/method-overriding-in-java)[Covariant Return Type](http://www.javatpoint.com/covariant-return-type)[super keyword](http://www.javatpoint.com/super-keyword)[Instance Initializer block](http://www.javatpoint.com/instance-initializer-block)[final keyword](http://www.javatpoint.com/final-keyword)[Runtime Polymorphism](http://www.javatpoint.com/runtime-polymorphism-in-java)[Dynamic Binding](http://www.javatpoint.com/static-binding-and-dynamic-binding)[instanceof operator](http://www.javatpoint.com/downcasting-with-instanceof-operator)[Abstract class](http://www.javatpoint.com/abstract-class-in-java)[Interface](http://www.javatpoint.com/interface-in-java)[Abstract vs Interface](http://www.javatpoint.com/difference-between-abstract-class-and-interface)[Package](http://www.javatpoint.com/package)[Access Modifiers](http://www.javatpoint.com/access-modifiers)[Encapsulation](http://www.javatpoint.com/encapsulation)[Object class](http://www.javatpoint.com/object-class)[Object Cloning](http://www.javatpoint.com/object-cloning)[Java Array](http://www.javatpoint.com/array-in-java)[Wrapper class](http://www.javatpoint.com/wrapper-class-in-java)[Call By Value](http://www.javatpoint.com/call-by-value-and-call-by-reference-in-java)[strictfp keyword](http://www.javatpoint.com/strictfp-keyword)[API Document](http://www.javatpoint.com/creating-api-document)[Command Line Arg](http://www.javatpoint.com/command-line-argument)[Object vs Class](http://www.javatpoint.com/difference-between-object-and-class)[Overloading vs Overriding](http://www.javatpoint.com/method-overloading-vs-method-overriding-in-java)[Array Quiz-1](http://www.javatpoint.com/directload.jsp?val=86)[Array Quiz-2](http://www.javatpoint.com/directload.jsp?val=87)[OOPs Quiz-1](http://www.javatpoint.com/directload.jsp?val=10)[OOPs Quiz-2](http://www.javatpoint.com/directload.jsp?val=83)[OOPs Quiz-3](http://www.javatpoint.com/directload.jsp?val=84)[OOPs Quiz-4](http://www.javatpoint.com/directload.jsp?val=85)

[**Java String**](http://www.javatpoint.com/java-string)[**Java Regex**](http://www.javatpoint.com/java-regex)[**Exception Handling**](http://www.javatpoint.com/exception-handling-in-java)[**Java Inner classes**](http://www.javatpoint.com/java-inner-class)[**Java Multithreading**](http://www.javatpoint.com/multithreading-in-java)[**Java I/O**](http://www.javatpoint.com/java-io)[**Java Networking**](http://www.javatpoint.com/java-networking)[**Java AWT**](http://www.javatpoint.com/java-awt)[**Java Swing**](http://www.javatpoint.com/java-swing)[**Java Applet**](http://www.javatpoint.com/java-applet)[**Java Reflection**](http://www.javatpoint.com/java-reflection)[**Java Date**](http://www.javatpoint.com/java-date)[**Java Conversion**](http://www.javatpoint.com/java-string-to-int)[**Java Collection**](http://www.javatpoint.com/collections-in-java)[**Java JDBC**](http://www.javatpoint.com/java-jdbc)[**Java New Features**](http://www.javatpoint.com/New-features-in-java)[**RMI**](http://www.javatpoint.com/RMI)[**Internationalization**](http://www.javatpoint.com/internationalization-in-java)[**Interview Questions**](http://www.javatpoint.com/corejava-interview-questions)

|  |
| --- |
| space |
|  |
| [**next →**](http://www.javatpoint.com/java-naming-conventions)[**← prev**](http://www.javatpoint.com/operators-in-java)  Java OOPs Concepts   1. [Object Oriented Programming](http://www.javatpoint.com/java-oops-concepts#oops) 2. [Advantage of OOPs over Procedure-oriented programming language](http://www.javatpoint.com/java-oops-concepts#oopsadvantage) 3. [Difference between Objcet-oriented and Objcet-based programming language.](http://www.javatpoint.com/java-oops-concepts#oopsdifference)   In this page, we will learn about basics of OOPs. Object Oriented Programming is a paradigm that provides many concepts such as**inheritance**, **data binding**, **polymorphism** etc.  **Simula** is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.  **Smalltalk** is considered as the first truly object-oriented programming language.  OOPs (Object Oriented Programming System)  java oops concepts**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:   * Object * Class * Inheritance * Polymorphism * Abstraction * Encapsulation   **Object**  Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.  **Class**  **Collection of objects** is called class. It is a logical entity.  **Inheritance**  **When one object acquires all the properties and behaviours of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.  polymorphism in java oops concepts  **Polymorphism**  When **one task is performed by different ways** i.e. known as polymorphism. For example: to convense the customer differently, to draw something e.g. shape or rectangle etc.  In java, we use method overloading and method overriding to achieve polymorphism.  Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.  **Abstraction**  **Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing.  In java, we use abstract class and interface to achieve abstraction.  encapsulation in java oops concepts  **Encapsulation**  **Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.  A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.  Advantage of OOPs over Procedure-oriented programming language   |  | | --- | | 1)OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows. | | 2)OOPs provides data hiding whereas in Procedure-oriented prgramming language a global data can be accessed from anywhere. | | 3)OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language. |  |  |  | | --- | --- | | Global Data | Object Data |   **What is difference between object-oriented programming language and object-based programming language?**   |  | | --- | | Object based programming language follows all the features of OOPs except Inheritance. JavaScript and VBScript are examples of object based programming languages. |   Do You Know ?   |  | | --- | | * Can we overload main method ? * Constructor returns a value but, what ? * Can we create a program without main method ? * What are the 6 ways to use this keyword ? * Why multiple inheritance is not supported in java ? * Why use aggregation ? * Can we override the static method ? * What is covariant return type ? * What are the three usage of super keyword? * Why use instance initializer block? * What is the usage of blank final variable ? * What is marker or tagged interface ? * What is runtime polymorphism or dynamic method dispatch ? * What is the difference between static and dynamic binding ? * How downcasting is possible in java ? * What is the purpose of private constructor? * What is object cloning ? |   What we will learn in OOPs Concepts ?   |  | | --- | | * Advantage of OOPs * Naming Convention * Object and class * Method overloading * Constructor * static keyword * this keyword with 6 usage * Inheritance * Aggregation * Method Overriding * Covariant Return Type * super keyword * Instance Initializer block * final keyword * Abstract class * Interface * Runtime Polymorphism * Static and Dynamic Binding * Downcasting with instanceof operator * Package * Access Modifiers * Encapsulation * Object Cloning |   **Next Topic**[Naming Convention in Java](http://www.javatpoint.com/java-naming-conventions)  [← prev](http://www.javatpoint.com/operators-in-java)[next →](http://www.javatpoint.com/java-naming-conventions) | |

 

Like/Subscribe us for latest updates or newsletter[RSS Feed](http://feeds.feedburner.com/javatpointsonoo)[Subscribe to Get Email Alerts](http://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo)