What's up programmer and welcome to the Singleton tutorial! We are going   to explain one of the simplest yet one of the most used design   patterns out there: The Singleton Pattern. The singleton is a creational design pattern  this pattern ensures that only  one instance of its kind exists  and provides a single point of access to  it from any other part of our application|As a real-world analogy take the  government of a country or its president,  you see a given country can have only one  official government or one official president  regardless of the individuals who form  this government or the person that   is currently the president of this country, so the title "the government or the president   of X" is a global identifier it is a global point of access   used to represent or reference the president  or the individuals in charge of this country.|Singletons represent exactly the same idea, just like global variables they let you access   your object from anywhere in your application but on top of that, they encapsulate the   attributes of this object in one single class and guarantee that only one instance of this class   will be available at any point in time but why would anyone want to control   how many instances a class has?| Well think of a database instance,  only one instance of a database is required and  should be used by all the parts of our application  So instead of creating a new instance  every time we need to query our database  we create a singleton database instance and reuse  this same instance every time we access our tables  by doing this we didn't only guarantee a  faster access and retrieval of the instance  but we also reduced the possibility  of encountering an error or exception   while creating these several instances.  Now let's go ahead and try to  implement the Singleton class together.|The first thing we need to do is create a  private static field inside our singleton,  this field will store the singleton instance under this instance you can go ahead and add   the attributes of your Singleton class in case of a database these attributes   can be the username and password needed  to connect for example or anything else|Now what we need to do is add a constructor  to create an object based on this class  make sure that the access level of  this constructor is set to private  inside this constructor we can go ahead and  initialize all the attributes of our object  in our case we only have one the "data" field.|This constructor was made private because we   only need access to it using a public static  method created inside the singleton class  usually this method is named getInstance() to  refer to the same single instance it returns  so this method should return an  instance of the Singleton class  the instance we declared at the  beginning of our implementation  but how do we guarantee that the same  instance is returned every single time? | Well the created instance is wrapped inside an   if-statement that checks if the static  instance of our singleton is null or not  and only if it is we proceed by creating a new  instance of it using our private constructor  however if it isn't we return the same instance  already stored inside our Singleton class.|So to recap we can say that the Singleton Pattern  should be used when a class must have a single   instance available to all its clients as this pattern disables all other means   of creating objects of a class except  for the special static creation method  This method either creates a new object or returns  the existing one if it has already been created.|So that's it for this tutorial, I hope it was   helpful, don’t forget to take the quiz to get your badge!