What is Junit ?  
  
Junit is basically a unit testing framework in JAVA which is used for writing and running tests.  
 It provides some annotations to identify the test methods. We will see about some annotations a bit later when we start writing test cases.  
  
So, all of you just guess one thing.  
What may be the steps for a test execution ?

1) Preparation – we need to write the test cases ?  
2) Provide the test inputs.  
3) Provide the expected result  
4) Run the test  
5) Verify the result  
6) And if the test fails we need to alert the developer about it right ?

So, here when we use Junit framework we don’t need to take any tension about running the test, verifying the result and to alert the developer. Junit will take care of this.   
We just need to concentrate on what ?  
Correct!

Here, we will be learning about Junit-5, So the question now arises why did we need Junit-5, Basically what was the issue with Junit-4?   
Let me give you some brief about it.  
You may take it down if you wish might be helpful in the future.  
1) Junit was actually almost 10 years old. The Junit-4 wasn’t updated according to the new testing standards/patterns.  
You can just imagine right 10 years in this IT market is a lot of time. So, basically the bugs and defects all piled up.  
2) Also, it was not up to date with the new JAVA features.  
JAVA as we all know has evolved significantly over the years right, we now have Lambda, stream API, functional interfaces and what not. So, Junit-4 wasn’t at all compatible with those features, which created a problem for the developers as Junit is like the bread and butter for every developer. Every developer uses it.  
3) Also, there is one more important point that Junit-4 was based on a monolithic architecture. Does anyone know what is a monolithic architecture ?  
Okay, so basically what it means that Junit-4 had only one JAR file. So, if you want only a small portion of the JAR to conduct a particular test, no it din’t have the ability to do that. You had to drop the whole JAR file to perform the test.  
This also led to one thing that is the advancement of the new features. So, if any new features had to be added the JAR file grows significantly. I suppose you getting where the problem lies. Right ?

JUNIT ARCHITECTURE:  
  
Platform, JUPITER, VINTAGE and IDE /Tools  
PLATFORM:   
It is the core which comprises of the library of Junit in running the tests, calling the test runners, proving an execution context and a bunch of those things.

We as developers don’t directly interact with the platform.

What we typically interact with is the Junit API which is called JUPITER.  
So, basically all the annotations which I talked about earlier are a part of this API and it consist of many more things like assertions, assumptions. We will be covering all of them in the later part of the session.  
  
So, let me tell you this interesting thing, so why is this called JUPITER, why did the Junit people named it JUPITER, can anyone take a guess ?  
  
JUPITER- 5th planet, junit 5  
starts with ju  
so if you think about it, it is a pretty good API name.  
  
So now we know that we are going to use this JUPITER API going forward.   
  
VINTAGE: Vintage is also an API, we can use it to run your older tests as the name suggests. So basically we don’t use much of vintage, but it’s there.

IDE/Tools: IDE uses the platform to run the tests. So, we will write the test cases using Jupiter and the platform knows that OH! These are the test this guy wants me to run. So, the IDE’s will use the platform to run those tests. Got it ?