1. Initially, JVM acts like any other interpreter -> running each line of code as it is needed.  
   However, by default this would make the code execution somewhat slow  
   certainly, if we compare this code to the code written in a language like C, which would be compiled to native machine code which OS can directly understand and needs no additional interpreter to run the code which makes it quick to run compared to **interpreted languages**.
2. But using the languages that are compiled **natively** means that we will lose the “Right Once And Run Anywhere” feature of Java.
3. To help get around this problem of slower execution of interpreted **languages than the compiled languages**, the JVM has a feature called **JIT** **(Just-In Time) Compilation**.
4. The JVM will monitor which branches of the code are run the most frequently (which methods or parts of methods, specifically loops are executed most frequently).
5. Then the JVM will compile that piece of code to native language to makes its execution faster.  
   So, at this point, some of our code is being run as an **interpretive mode** as byte codeand some is running **compiled native machine** **code**.
6. JVM is **multithreaded**. So, one thread is responsible for interpreting the byte code and execute it and another thread for doing JIT compilation.   
   So, the JIT process will not stop JVM to execute the code and once the code is compiled natively, the JVM will start executing that code.  
   If our app is heavy on processing using all of the available CPU resources, then we could potentially see a temporary reduction in performance if **JIT compilation** is taking place but it is worth the slight dip in processing power to get the benefits of native code version.
7. One **impact of JIT Compilation** is that if we are **assessing the performance** of a particular piece of code, we actually need to think about when we are going to do that assessment as if we try to assess the performance of two methods soon as they start, we might get different result if we start assessing the methods after our app has been running for a short while.  
   So, basically we need to check if the assessment is being performed before JIT Compilation or after.
8. As a programmer, it might be interesting to see which part of the code is being compiled to native code.