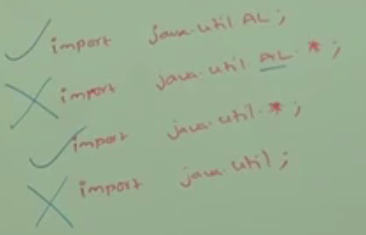
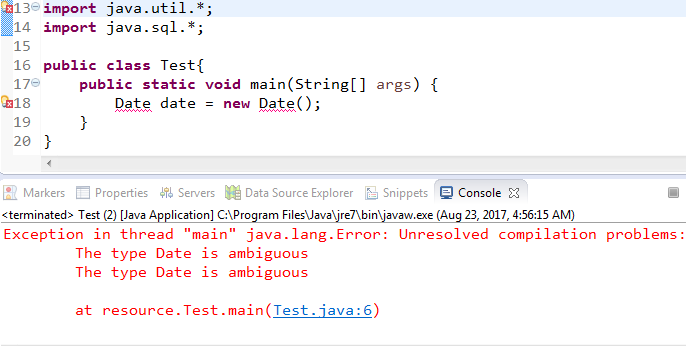
Types of import statement

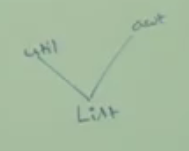
1. **Case 01:**
   1. **2 types**
      1. **Explicit class import**:  
          java.util.ArrayList
      2. **Implicit class import**:  
          java.util.**\***
   2. Which import type is recommended?
      1. Explicit Class Import because it improves readability of the code.
      2. Best suitable for hi-tech where readability is important.
2. **Case01**: which of the following import statements are meaningful?
   1. 
3. **Case03**: Consider the following code.  
   

The code compiles fine even though we are not writing import statement because we used fully-qualified name.

NOTE: Whenever we are using fully-qualified name, it’s not required to write corresponding import statement. Similarly, whenever we are writing import statement, no need to write fully-qualified name.

1. **Case04**: A bit dangerous case ☺  
   

Because both package util and sql have Date class.

NOTE: Even in the case of **List** also, we can get the same abmiguity problem.  


1. **Case05**: While resolving class name, compiler always gives the preference in the following order
   1. Explicit Class Import.
   2. Classes present in current working directory (Default Package).   
      Jatin: I think. It means that if class A is in a.b and A class has reference to Date then the Date .class in a.b will be preferred.
   3. Implicit Class Import.
2. **Case06**: Whenever we are importing a java package, all classes and interfaces present in that package by default are available but not sub-package classes.  
   If we want to use sub-package classes, we need to write import statement till sub-package level.
3. **Case07**: Two packages are imported automatically.
   1. **java.lang** package
   2. **default packge (current working directory)**
4. **Case08**:

No other difference   
b/w these two classes

import statements  
short name

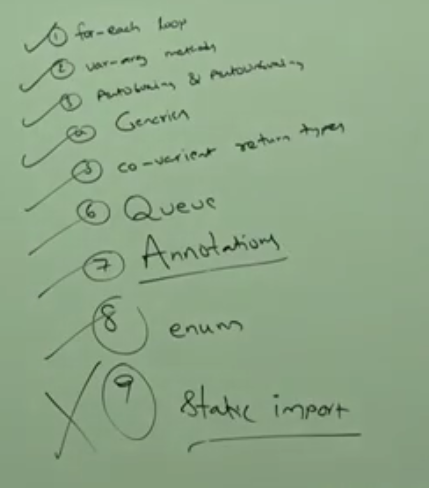
only fully-qualified class name

The import statement would take more time as compared to class having fully-qualified class name to resolve the name to exact class.

* 1. import statement is totally compile time related concept. The more no of import statements, the more compile time but there is no effect on execution time.

1. **Case09**: Difference b/w C Language #include and Java Language import statement.
   1. **#include**: All header files will be loaded at the beginning when the program is translated (preprocessing). Hence, it’s static include.
   2. **import statement**: No class file will be loaded at the beginning. Whenever we are using a particular class, only then corresponding .class file will be loaded.  
      Called🡪 Dynamic Include or Load on Demand or Load on Fly
2. d

static import

1. In JDK 1.5 Version new features were introduced.   
   
2. Static import came into JDK 1.5 version. According to Sun, usage of static import reduces the length of the code and improves readability but according to World Wide Programming Experts (like us), usage of static import creates confusion and reduces readability. Hence, if there is no specific requirement then it’s not recommended to use static import statement.
3. **How to write static import statement?**
   1. import static java.lang.Math.\*; // importing all static members statically
   2. import static java.lang.Math.sqrt; // importing only sqrt() statically.
   3. Usually we can access by using class name such as Math.sqrt(36) but whenever we are static import, we can access static member directly without class name such as sqrt(36)