|  |  |
| --- | --- |
| Native | Native: Designed to run on a particular platform (usually an OS)  -Java Byte code is not natively executable on OS  -Byte code is system independent (can run on Mac or Windows), meaning it can on any system regardless of the platform  -WORA (Written Once Run Anywhere) thanks to JVM |
| Native part2  -Java apps are not native apps, they have to be run in a Java environment. To create java environment, we use java command (will launch an app into that java env) | -Java apps can’t run directly in the host environment (Windows, Mac, Linux) because its platform agnostic. Just needs JRE on the host’s environment  --Java apps are not native apps, they have to be run in a Java environment. To create java environment, we use java command (will launch an app into that java env) Example: Can’t use Main -> java Main? |
| JDK  A screenshot of a computer  Description automatically generated  Jakarta EE: API libraries towards web and db access (not included in JDK) | -Also includes jar, Javadoc  -builds java code and develops apps  Needs Java SE because of imports (System class is part of Java SE API) |
| JRE  A diagram of a library  Description automatically generated | -allows apps to run  -Because Java is a cross platform environment when I build my Java app, it doesn’t contain the native code that can run in a host environment (platform agnostic)  -It contains byte code and it has to be translated to execute within a particular host environment. JRE makes it possible |
| JVM  (java Hello.class)  java (Java runtime aka JVM?/Java Interpreter?)  -loads the .class file and executes it | -calls the main methods  -runs our Hello.class files (knows how to read Java bytecode) |
| JAVA\_HOME  PATH (System Env Variable: for all uers) | -point to the installation directory of JDK  -update OS’s PATH variable to include the JDK/bin (directory that contains javac and runtime) So we can use it on command prompt |
| Package  A close up of a sign  Description automatically generated | -a namespace, all lowercase?  -Use reverse domain name notation to assure global uniqueness  -Add further qualifiers to assure uniqueness within a company  A blue sign with white text  Description automatically generated |
| Type Names | -Since package names are unique, it allows our type names to be globally unique  Running    -While the simple class name is Hello, the real class name is a.b.c.Hello.  -Because class names might repeat in different libraries |
|  |  |
|  |  |
|  |  |
|  |  |