**JAVA BEST PRACTICES**

* **Using Naming Conventions :**

**Self-explanatory**: a name must reveal its intention so everyone can understand and change the code easily. For example, the names d or str do not reveal anything; however the names daysToExpire or inputText do reveal their intention clearly. Note that *if a name requires a comment to describe itself, then the name is not self-explanatory.*

**Meaningful distinctions**: If names must be different, then they should also mean something different. For example, the names a1 and a2 are meaningless distinction; and the names source and destination are meaningful distinction.

**Pronounceable**: names should be pronounceable as naturally as spoken language because we are humans - very good at words. For example, which name can you pronounce and remember easily: genStamp or generationTimestamp?

**Some general naming rules :**

Class and interface names should be nouns, starting with an uppercase letter. For example: Student, Car, Rectangle, Painter, etc.

Variable names should be nouns, starting with a lowercase letter. For example: number, counter, birthday, gender, etc.

Method names should be verbs, starting with a lowercase letter. For example: run, start, stop, execute, etc.

Constant names should have all UPPERCASE letters and words are separated by underscores. For example: MAX\_SIZE, MIN\_WIDTH, MIN\_HEIGHT, etc.

Using camelCase notation for names. For example: StudentManager, CarController, numberOfStudents, runAnalysis, etc.

* **Class Members should be private**
* **Avoid Empty Catch Blocks**
* **Use Logging**
* **Use Service Layer**
* **Create service interface and ServiceImpl class to use service methods**
* **Keep your @Controller ‘s clean and focused**
  + **Controllers should be stateless**! Controllers **are by default singletons** and giving them any state can cause massive issues.
  + Controllers should not execute business logic but rely on delegation.
  + Controllers should deal with the HTTP layer of the application. This should not be passed down to Services.
  + Controllers should be oriented around a use-case / business-capability.
* **Use JPA/Crud Repository and Use PagingAndSortingRepository for Datatable pagination**
* **Use jwt security in every api**