Java Reflection API is an advanced feature of Java used for introspecting and modifying application from class levels during runtime.

Three main things that we need to take care of while using reflection API

* Performance : Reflection API has slower performance as it is dynamically resolved.
* Exposure of Internals : Private methods or fields can be reached by using reflection API .

It also breaks encapsulation & abstraction of internals.

* Security : It requires runtime persmissions to interospect & modify the application.

Main features of reflection API :

* Extensibility feature : application make use of external , user -defines classes by creating instances of extensibility objects using fully qualified names.
* Debugger & Test tools : Debuggers even need private members of class to understand the level of coverage by using reflection API.
* Visual development environment : IDE can use type information or class members by using reflection.

Get instance of class in 3 ways

* ForName() method – It loads the class dynamically and returns the instance of the class. It should by used if you know the exact name of the class.
* getClass() method – it returns the instance of class and should be used if type of class is known
* .class syntax – If theres is no instance of class , .class syntax can be used to get the instance of the calss.

Text

Description automatically generated

Methods in reflection API

* getConstructors() → Used to get constructors represented by the class
* getDeclaredConstructors() → Used to get declared constructors represented by the class
* getDeclaredConstructor(@Nullable Class<?> …parameterTypes) → Used to get declared constructor represented by the class. parameterTypes is an array of Class objects that identify the class constructor in the declared order.
* getMethods() → Used to get methods in the initialized class
* getDeclaredMethods() → Used to get declared methods in the initialized class
* getDeclaredMethod(@NonNls @NotNull String name, Class<?> …parameterTypes) → Used to get declared method represented by the class. name is the exact name of the method, parameterTypes is an array of Class objects that identify the method signature in the declared order.
* getDeclaredPublicConstructor(@NonNls @NotNull String name, Class<?> …parameterTypes) → Used to get declared only public method represented by the class. name is the exact name of the method, parameterTypes is an array of Class objects that identify the method signature in the declared order.
* getDeclaredFields() → Used to get declared fields represented by the class. This includes public, protected, default (package) access, and private fields, but excludes inherited fields.
* getDeclaredField(@NonNls @NotNull String name) → Used to get declared field with the name parameter represented by the class.
* setAccessible(boolean flag) → Used to set the accessibility of the reflected object.
* invoke(Object obj, Object… args) → Used to invoke the underlying method on the specified object with the given parameters. Parameters must be given to the invoke method with the declared order in the underlying method.
* isDefault() → Used to understand whether the underlying method is a default method or not.