①

Reflection 2: Object Creation and Constructors:

Agenda: Learn about: (1) Constructor<?> Intro and Ways to obtain it via Reflection
(2) Constructor Params inspection using Reflection
(3) Dynamic Object Creation Using Reflection.

(I) * Constructor<?> object represents Constructor of a Java Class.

* Constructor<?> object contains all the infomation about class's constructor including the number and types of Parameter it takes.

* Since a Java Class can have multiple Constructors, which can be Public or non Public, There are diffrent ways to get Constructor<?> objects:

(i) Class.getDeclaredConstructors(): → Gets all Public & non Public Constructors

(ii) Class.getConstructors(): Returns only the Public Constructors of Class.

Has 's' So Return Type is Constructor<?>[]

② (iii) Class.getConstructor (Class<?> paramTypes) / OR
Class.getDeclaredConstructor (Class<?> ParamTypes) :→

No s' So ReturnType Constructor<?>

If you want to get a particular Constructor when you know its ParamTypes-

This method of Getting Constructor can throw 'NoSuchMethodException' if Constructor is not found.

(II)

* To get the param types of Constructor of Java Class :- [Constructor<?>.getParameterTypes()]

Syntax:

```
Class<?> clazz = String.class;
Constructor<?>[] getDeclaredConstructors = clazz.getDeclaredConstructors();
Class<?>[] paramTypes = getDeclaredConstructors[0].getParameterTypes();
```

To print Parameter: `paramTypes[0].getSimpleName(); // eg: String`

③ (III) Dynamic Object Creation Using Constructor<?> :

We will understand this using an example:

Example: Implement a Factory Method that can create Object of any Class:
Depending on the arguments passed to our method, it will find right Constructor
Create the given Class object by calling & Identifying right Constructor.
Without Reflection this is impossible.

Syntax: Constructor.newInstance(Object ...Args).
Args should match the Type and Order of Parameters in a Constructor defined.
Will return the instance of Java class or will throw Exception.

Full Code with this factory Method, See IntelliJ Reflection2.java.

(END).