Factory is a creational design pattern, i.e., related to object creation.

Factory pattern hides the creation process of objects from being exposed to the caller. Caller of Factory doesn't know how the object gets created. This will make the code loosely coupled.

Example:

// Factory classpublic class FruitFactory {

public Object createFruit(String name){

Object obj = null;

switch(name){

case "apple":

obj = new Apple();

break;

case "orange":

obj = new Orange();

break;

}

return obj;

}}

// Client Code

FruitFactory factory = new FruitFactory();Apple apple = (Apple)factory.createFruit("apple"); // from her the caller doesn't know how the apple object got created.

Second way:

// Factory classpublic class FruitFactory2 {

Map<String,Object> fruitMap = new HashMap<>();

public FruitFactory2(){

fruitMap.put("apple", new Apple());

fruitMap.put("orange", new Orange());

}

public Object createFruit(String name){

return fruitMap.get(name);

}}

// Client code

FruitFactory2 f2 = new FruitFactory2();

f2.createFruit("apple");

Using Factory Pattern, we can achieve lots of techniques like Dependency Inject, Inversion Of Control. that's how Spring IOC does it.