**Hospital Management Application Development**

**Important Instructions:**

Please read the document thoroughly before you code.

Import the given skeleton code into your Eclipse.

Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers, etc.

You can create any number of private methods inside the given class.

You can test your code from the main() method of the program.

Using Spring Core, develop the application using **JavaConfig.**

**Time: 1 hour**

**Assessment Coverage:**

**Classes, Objects, and Constructor Injection**

**Interface, Autowire, and Annotation**

Application Scenario: Create a simple simulation for the workflow of a hospital's patient registration and treatment process. The Head Hospital sets rules for Regional Hospitals on how to process patient documents and provide treatment.

**Skeleton File for Development:**

Import the below attached skeleton code into your eclipse project and implement the required functionalities



**Technical Requirements:**

You are required to develop an app following the below conditions.

**Step 1:** Create an Interface **HeadHospital** with the below-mentioned public methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Specifier/ Modifier** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| Public abstract | doDocumentVerification | nil | Void | This method should do verification of the patient based on ID proof like insurance, ID card, or health card. This method should print which ID proof was used for verification. |
| Public abstract | provideTreatment | Nil | Void | This method provides treatment to the patient. This method will print the name and condition of the patient. |

**Step 2:** Create class **NewYorkHospital** which implements **HeadHospital** and gives implementation for **doDocumentVerification** and **provideTreatment** methods. Annotate the class with **@Component.**

**Variable:**newYorkDocument of type Document

**Parameterized Constructor:**   
Create a parameterized constructor that takes a Document object. Annotate the constructor with @Autowired.

**Step 3:** Create class **LosAngelesHospital** which implements **HeadHospital** and gives implementation for **doDocumentVerification** and **provideTreatment** methods. Annotate the class with @Component.

**Variable:** losAngelesDocument of type Document

**Parameterized Constructor:** Create a parameterized constructor that takes a Document object. Annotate the constructor with @Autowired.

**Step 4:** Create class **Document** which has the following variables.

**Variables:** name of type String, idProof of type String, condition of type String

**Step 5:** Create clas**s AppConfig** which will be used as a configuration class. Annotate this class with **@Configuration** and **@ComponentScan** and create below methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Annotation** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| @Bean | newYorkDocument | nil | Document | This method will create and return a document object for New York patient, set name as Alice, idProof as Insurance, and condition as "Flu" |
| @Bean | losAngelesDocument | nil | Document | This method will create and return a document object for Los Angeles patient, set name as Bob, idProof as ID Card, and condition as "Sprain" |

**General Design Constraints:**

Ensure that all the Java Coding Standards are followed.

Assume that the method inputs are valid always, hence exceptional blocks are not needed to be included in the development.

**Sample Input Output 1:**

Select location:

1. New York

2. Los Angeles

**1**

Document verification done using Insurance

Treatment is in progress for Alice with condition Flu at New York Hospital

**Sample Input Output 2:**

Select location:

New York

Los Angeles

**2**

Document verification done using ID Card

Treatment is in progress for Bob with condition Sprain at Los Angeles Hospital