# Spring Core Hands On – PassportProcess

**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 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 created should be a simple simulation for workflow of passport office. Head Passport office

Setting rules for Regional Passport office how to process document and issue of passport.

## Skeleton File for Development:

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

PassportService_CodeSkeleton.zip

## Technical Requirements:

You are required to develop an App following below conditions.

**Step 1:** Create an Interface HeadPassportOffice with below mentioned public methods :

| Access Specifier/ Modifier | Method Name | Input Parameters | Output Parameters | Logic |
|---|---|---|---|---|
| Public abstract | doPhotoVerification | nil | Void | This method should do verification of applicant based on photo id like driving licence , aadhaar card or pan card .This method should print which id proof was used for verification |
| Public abstract | issuePassport | Nil | Void | This method issues passport to the applicant. This method will print the name and location of applicant. |

**Step 2:** Create class **ChennaiPassportOffice** which implements HeadPassportOffice and gives implementation for **doPhotoVerificatioin and issuePassport** methods .Annotate the class with @Component.

**Variable:**

chennaiDocument of type Document

**Parameterized Constructor:**

Create a parameterized constructor which takes a Document object. Annotate the constructor with @Autowire.

**Step 3**: Create class **BangalorePassportOffice** which implements HeadPassportOffice and gives implementation for **doPhotoVerificatioin and issuePassport** methods .Annotate the class with @Component.

**Variable:**

bangaloreDocument of type Document

**Parameterized Constructor:**

Create a parameterized constructor which takes a Document object. Annotate the constructor with @Autowire

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

**Variables:**

name of type String , idProof of type String, city of type String

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

| Annotation | Method Name | Input Parameters | Output Parameters | Logic |
|---|---|---|---|---|
| @Bean | chennaiDocument | nil | Document | This method will create and return document object for Chennai candidate, set name as Joe , idProof as driving licence and city as Chennai |
| @Bean | bangaloreDocument | nil | Document | This method will create and return document object for Bangalore candidate , set name as John , idProof as Aadhaar Card and city as Bangalore |

**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.Chennai
2.Bangalore
1
Photo verification done using Driving Licence
Passport issue is in progress for Joe from Chennai office
```

**Sample Input Output 2:**

```
Select location:
1.Chennai
2.Bangalore
2
Photo verification done using Aadhaar Card
Passport issue is in progress for John from Bangalore office
```