Hands-on Assessment 13 Sep

Create a Java application based on the use case described below:

A company assigns bundle of software licenses to the newly joined employees based on their roles and on the availability of licenses. There are different types of bundles as follows:

1. Generic bundle: which contains MsOffice & Zoom software
2. Developer bundle: which contains softwares that are part of Generic bundle and also contains a Programming Language, a WebServer and a Database software
3. Tester bundle: which contains the softwares that are part of the Developer Bundle and also includes additional Testing Software.

The application maintains the details of each software, its license number and its allocation status using a data structure from the Collection framework in Java. For example, if the software is a programming language (say Python), then the data structure would contain the following information Python; "P12122"; "Allocated" if the license is allocated to a bundle, or Python; "P12122"; "Available" if the license is not allocated to any bundle. (Note, the format <<Software>>;<<License>>;<<Status>> is only representational and it is not a specification for actual storage)

The LicenseRepository component is responsible for adding new software licenses to the collection, removing software licenses from the collection, and allocating/deallocating software licenses as applicable. Appropriate exceptions should be raised in case of attempt to add a software with a duplicate license number, attempt to remove a software with an invalid license number, attempt to allocate a license if all licenses are exhausted, and attempt to deallocate a license with an invalid license number.

The BundleManagement component is responsible for allocating and deallocating bundles to employees and contains an inventory of which employees (based on id) are allocated with bundles. This inventory is currently a collection framework data structure to store this data. The bundle management component controls the business logic. To allocate a bundle, the component needs the employee id and the role that the employee is playing. To deallocate a bundle, the component needs the employee id of the employee. When a request to allocate a bundle is raised, the component should interact with LicenseRepository and fetch the licenses for all the software in the bundle and allocate the bundle to the employee and record the allocation in the BundleRepository. Similarly, when a request for deallocating a bundle is raised, the component should interact with the LicenseRepository and deallocate the licenses that are part of the bundle and record the deallocation of bundle in the BundleRepository. Appropriate exceptions should be raised/re-thrown depending on the context. For example, If an employee is already allocated with a bundle a new bundle cannot be allocated. In another example, if there are no licenses available for a particular software in a bundle, then the bundle can't be created and allocated to the employee.

In the long term, the LicenseRepository will be migrated from collection framework to other data storage mechanisms such as files or databases. Ensure that your design caters to this change in a way that minimal changes to the application is required when this happens.

Finally, create a UserInterface class with main method which can help in executing the following flows:

1. Procuring several licenses and stocking up the license repository
2. Allocating bundles to employees (generic, developers and testers)
3. Deallocating a bundle from a generic employee and allocating that employee with a tester bundle assuming that the employee's role changes.

Ensure that you follow the best practices in naming conventions, layered architecture, and exception handling.