Ingeniería Civil en Computación e Informática
Ingeniería Civil Industrial
Ingeniería en Tecnologías de Información

Prueba Sumativa de Laboratorio Nº 1

Patrones de software y programación - 09 de Mayo 2024

Nombre		RUT	
Paralelo	( ) - C1 ( ) - C2 ( ) - C3		

Antecedentes Generales:

Puntaje total de la prueba/Puntos para nota aprobatoria(4.0)	10 puntos	**Puntaje Obtenido**	
Duración de la prueba	1.5 horas	**Nota Final**	
Resultados de Aprendizaje a evaluar	RA1: Identificar posibles patrones de diseño de software en una solución ya implementada RA2: Caracterizar patrones de diseño de acuerdo a su nivel de aplicabilidad a los problemas		
Fecha de entrega de resultados	28 de Mayo 2024		

Instrucciones:

1. Esta evaluación tiene 2 páginas (incluyendo la portada) y 2 preguntas. Compruebe que dispone de todas las páginas. Responda a las preguntas en los espacios previstos para ello en las hojas de preguntas. Asegúrese de marcar apropiadamente sus respuestas.

2. Durante la prueba no se puede utilizar: teléfono móvil, calculadora, apuntes . Está prohibido intentar conectarse a internet de cualquier manera. Si es sorprendido obtendrá la calificación mínima. Tampoco puede utilizar dispositivos de almacenamiento externos o cualquier otro dispositivo relojes inteligentes, ábacos, etc.

3. Lea la prueba completamente DOS veces antes de hacer cualquier pregunta

4. Una prueba respondida correctamente en un 60 %, de acuerdo a las ponderaciones asignadas, corresponde a una nota 4,0.

5. Solamente se pueden realizar preguntas durante los primeros 10 minutos de la prueba. Solo se responderán preguntas respecto a los enunciados a viva voz.

6. La prueba es individual, cualquier sospecha de copia será calificada con la nota mínima y el caso será remitido al comité de ética.

7. En su espacio personal no debe haber nada más que hojas de papel en blanco, lápiz, goma y/o calculadora.

8. El resto de sus implementos debe guardarlos dentro de su mochila/bolso y ésta debe posicionarse al frente debajo de la pizarra. Si leyó hasta este punto, felicidades, para saber que lo hizo dibuje una estrella al final de esta página.

Acepto las condiciones firmando:_____

Enunciados/Problemas

Pregunta 1 [10].

Usted como ingeniero de software debe desarrollar la implementación de un **Bundle** OSGi, correspondiente con la persistencia de un sistema de biblioteca. Para tal efecto deberá seguir las siguientes instrucciones y contestar las preguntas correspondientes en Campus Virtual.

Set-up de los proyectos e Instrucciones

1. Descargue el desarrollo desde la siguiente URL:
   `https://github.com/IS-LAB-EIC-UCN/Lab-I-2024-SoftPatterns`
   Para la descarga, puede descargar el archivo *.zip* o utilizar git si lo tiene instalado en su computador con la instrucción:
   `git clone https://github.com/IS-LAB-EIC-UCN/Lab-I-2024-SoftPatterns.git` . Donde descargó el *.zip* deben aparecer cuatro proyectos: `Api-Biblioteca, Impl-Biblioteca, Serv-Biblioteca y Tui-Biblioteca`.

2. Intellij permite abrir solo un proyecto por ventana, por lo que deberá abrirlos en ventanas separadas o si prefiere en la misma ventana, tomando en cuenta que el proyecto abierto con anterioridad cerrará.

3. **Importante:** Usted solo deberá trabajar en el bundle `Impl-Biblioteca`.

Desarrollos y Preguntas

P1 - Revise el POM del proyecto y vea que archivo de configuración está faltando. Una vez identificado crearlo en el lugar que corresponda. ¿Que hace este archivo y porqué es importante?. (2pts)

1. En el paquete `..api.impl`, deberá:

   a) Crear la clase `LibroMutableImpl.java` (Código 1)

   b) Adicionar código a la clase `InventarioImpl.java` (Código 2)

2. En el paquete `..inventario.activador` deberá crear la clase: `InventarioActivador.java`, Código 3.

Código 1: LibroMutable

```
1
2  // implementar getters and setters
3
4
5  public String toString() {
6
7          StringBuffer buf = new StringBuffer();
8          buf.append(this.getCategoria()).append(": ");
9          buf.append(this.getTitulo()).append(" de ").append(this.getAutor());
10         return buf.toString();
11 }
```

Código 2: InventarioImpl

```
1          public LibroMutable cargarLibroParaEdicion(String isbn)
2                  throws ExcepcionLibroNoEncontrado {
3
4                  LibroMutable libro= this.libroByIsbn.get(isbn);
5                  if (libro == null) {
6                          throw new ExcepcionLibroNoEncontrado(isbn);
7                  }
8                  return libro;
9          }
10         public Set<String> getCategorias(){
11                 return this.categorias.keySet();
12         }
13         public String guardarLibro(LibroMutable libro) throws ExcepcionLibroInvalido {
```

```
14
15                  String isbn = libro.getIsbn();
16                  if (isbn == null)
17                          throw new ExcepcionLibroInvalido("ISBN no esta seteado");
18
19                  this.libroByIsbn.put(isbn, libro);
20                  String categoria = libro.getCategoria();
21                  if (categoria == null)
22                          categoria = CATEGORIA_DEFECTO;
23
24                  if (this.categorias.containsKey(categoria)) {
25
26                          int contador = this.categorias.get(categoria);
27                          this.categorias.put(categoria, contador + 1);
28                  }
29                  else {
30                          this.categorias.put(categoria, 1);
31                  }
32
33                  return isbn;
34          }
35          public void removerLibro(String isbn) throws ExcepcionLibroNoEncontrado {
36
37                  Libro libro = this.libroByIsbn.remove(isbn);
38                  if (libro == null)
39                          throw new ExcepcionLibroNoEncontrado(isbn);
40
41                  String categoria = libro.getCategoria();
42                  int contador = this.categorias.get(categoria);
43                  if (contador == 1)
44                          this.categorias.remove(categoria);
45                  else
46                          this.categorias.put(categoria, contador -1);
47
48          }
49          public Libro cargarLibro(String isbn) throws ExcepcionLibroNoEncontrado {
50
51                  return cargarLibroEditar(isbn);
52          }
53          public LibroMutable cargarLibroEditar(String isbn)
54                  throws ExcepcionLibroNoEncontrado {
55                  LibroMutable libro = this.libroByIsbn.get(isbn);
56                  if (libro == null)
57                          throw new ExcepcionLibroNoEncontrado(isbn);
58                  return libro;
59          }
```

Código 3: Activador

```
1
2  public class InventarioActivador implements BundleActivator {
3
4          private ServiceRegistration<Inventario> reg = null;
5
6          @Override
7          public void start(BundleContext context) throws Exception {
8                  System.out.println("\nComenzando implementacion de Inventario de Libros ");
9                  Inventario inventario = new InventarioImpl();
10                 this.reg = context.registerService(Inventario.class,  inventario , null);
11
12         }
13         @Override
14         public void stop(BundleContext context) throws Exception {
15                 System.out.println("\nParando implementacion de Inventario de Libros ");
16                 if (this.reg!=null) {
17                         context.ungetService(reg.getReference());
18                         this.reg = null;
19                 }
20         }
21 }
```

Para aumentar la flexibilidad en la integración de servicios existentes en un framework de trabajo OSGi (en general), un servicio puede ir acompañado de un activador de paquetes, que tomará el control temporal de la ejecución durante el inicio y la detención del paquete, realizando así las tareas necesarias como registrar servicios, que es justamente lo que realiza el Código 3.
P2 - Explique en que consiste el mecanismo de acceso entre los distintos bundles para este desarrollo. Dibuje un diagrama que muestre como son las dependencias y súbalo a la plataforma. (2pts)

P3 - Modifique el POM.xml para dar soporte de repositorio. ¿Donde realizó los cambios?. (2pts)

Cuando no tenga errores en ningún bundle, realizar el deployment de cada bundle mediante consola de Linux o terminal de windows CMD, con el siguiente comando:

```
mvn clean deploy
```

P4 - Al deployar cada bundle, ¿donde estos quedaron registrados?. Subir a la plataforma el archivo repository.xml. (2pts)

Ejecutar Apache Felix y en la consola deberá indicar el lugar donde se encuentran los bundles con el siguiente comando **repos add file:///path/to/RepoOsgi/repository.xml**. Recuerde que puede consultar los comandos de consola con la palabra reservada *help*.

Realizar el deployment en Apache felix para cada bundle con el comando `deploy -s "Bundle-Name"`.

P5 - Obtenga un screenshot de su consola donde se muestre que realizó el deployment de todos los bundles. (2pts)

Bonus (No Obligatorio): Si los pasos anteriores son correctos, entonces usted debería visualizar los dos comandos que utilizará para esta entrega `libro:adicionar` y libro:buscar. El primero adiciona un libro al inventario y el segundo buscar un determinado libro, por algún campo, incluso pudiendo usar wildcards.

Cada vez que usted haga un `help adicionar` o un `help buscar` en la consola de Apache Feliz, aparecerá la información del comando y los parámetros que debe ingresar para usar el comando. Usted deberá:

1. Ingresar por lo menos 5 libros, donde 2 pertenecerán a la misma categoría, usando `(username, password, ISBN, titulo, autor,categoría)`, donde username y password es *admin* para ambos (1pt).

2. Buscar los libros que usted ingresó, usando `(username, password, atributo de la búsqueda, autor, titulo o categoría, match like%)`, donde username y password es *admin* para ambos (1pt).