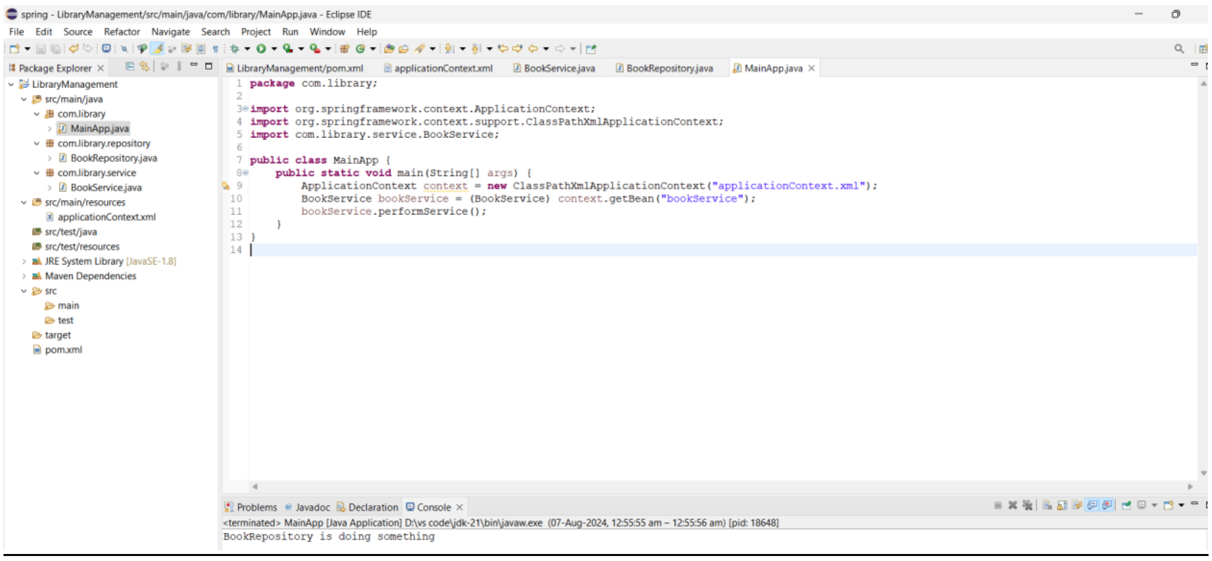
Digital Nurture 3.0 I Deep Skilling (WEEK 2 SOLUTIONS)

Superset ID:5021661

Name: Priya Hazra

Exercise 1: Configuring a Basic Spring Application



```java
package com.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.performService();
    }
}
```

```
<terminated> MainApp [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 12:55:55 am – 12:55:56 am) [pid: 18648]
BookRepository is doing something
```

CODE:

APPLICATIONCONTEXT.XML:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

    <bean id="bookRepository" class="com.library.repository.BookRepository"/>
</beans>
```

BookService.java

```java
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
public void performService() {
        bookRepository.doSomething();
    }
}
```

MainApp.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.performService();
    }
}
```

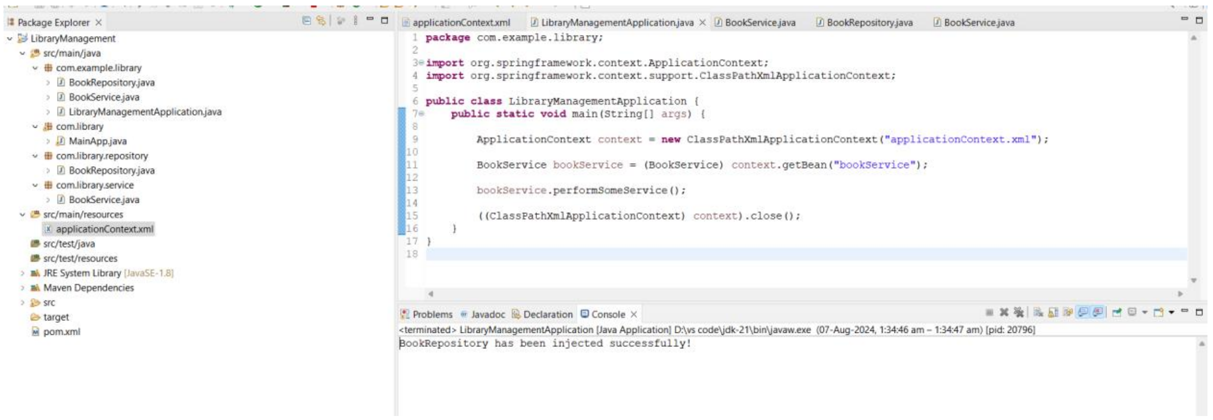
BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public void doSomething() {
        System.out.println("BookRepository is doing something");
    }
}
```

LibraryManagement/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.22</version>
    </dependency>
</dependencies>
</project>
```

Exercise 2: Implementing Dependency Injection



```java
package com.example.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {
    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");

        bookService.performSomeService();

        ((ClassPathXmlApplicationContext) context).close();
    }
}
```

```
<terminated> LibraryManagementApplication [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 1:34:46 am – 1:34:47 am) [pid: 20796]
BookRepository has been injected successfully!
```

CODE:

```java
package com.example.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {
   public static void main(String[] args) {

       ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

       BookService bookService = (BookService) context.getBean("bookService");

       bookService.performSomeService();

       ((ClassPathXmlApplicationContext) context).close();
   }
}
```

```java
package com.example.library;

public class BookRepository {

}
```

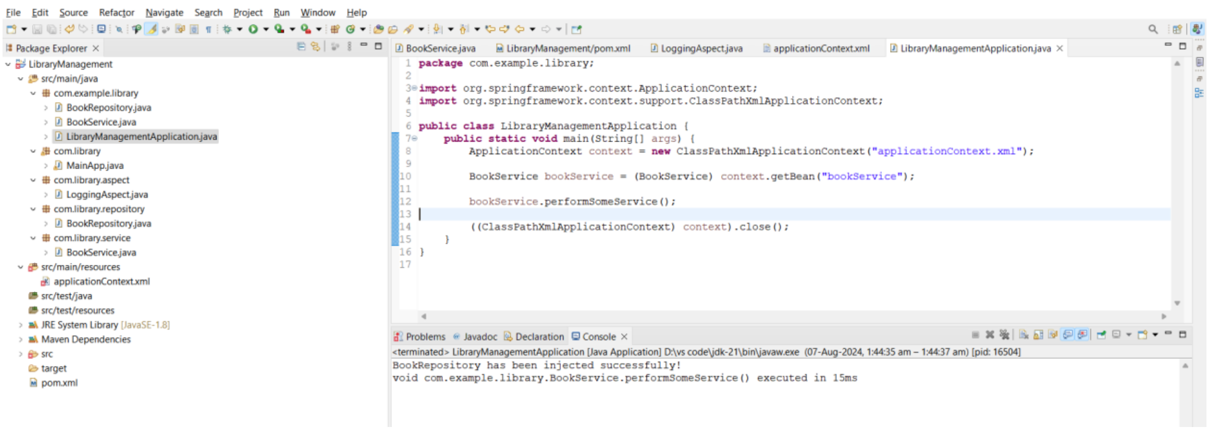
```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                  http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="bookRepository" class="com.example.library.BookRepository" />

   <bean id="bookService" class="com.example.library.BookService">
       <property name="bookRepository" ref="bookRepository" />
   </bean>
```

```
</beans>
```

```java
package com.example.library;

public class BookService {
   private BookRepository bookRepository;


   public void setBookRepository(BookRepository bookRepository) {
      this.bookRepository = bookRepository;
   }

   public void performSomeService() {
      if (bookRepository != null) {
         System.out.println("BookRepository has been injected successfully!");
      } else {
         System.out.println("BookRepository injection failed.");
      }
   }
}
```

```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
   private BookRepository bookRepository;

   public void setBookRepository(BookRepository bookRepository) {
      this.bookRepository = bookRepository;
   }

   public void performService() {
      bookRepository.doSomething();
   }
}
```

Exercise 3: Implementing Logging with Spring AOP



```java
package com.example.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");

        bookService.performSomeService();

        ((ClassPathXmlApplicationContext) context).close();
    }
}
```

```
<terminated> LibraryManagementApplication [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 1:44:35 am – 1:44:37 am) [pid: 16504]
BookRepository has been injected successfully!
void com.example.library.BookService.performSomeService() executed in 15ms
```

CODE:

```xml
<dependencies>
    <!-- Spring AOP Dependency-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>5.3.24</version> <!-- Use the version compatible with your Spring version-->
    </dependency>
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>1.9.9</version> <!-- Use the version compatible with your Spring AOP version-->
    </dependency>
</dependencies>
```

```java
package com.library.aspect;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;
@Aspect
@Component
public class LoggingAspect {
    @Around("execution(* com.example.library..*(..))")
    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        long executionTime = System.currentTimeMillis()- start;
        System.out.println(joinPoint.getSignature() + " executed in " + executionTime + "ms");
        return proceed;
    }
}
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-beans.xsd
                    http://www.springframework.org/schema/aop
                    http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Enable AspectJ auto proxying-->
    <aop:aspectj-autoproxy/>

    <!-- Define beans-->
    <bean id="bookRepository" class="com.example.library.BookRepository" />
    <bean id="bookService" class="com.example.library.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>

    <!-- Register the LoggingAspect bean-->
    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>
</beans>
```

```
package com.example.library;
```

```
import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;


public class LibraryManagementApplication {

  public static void main(String[] args) {

    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");


    // Get the BookService bean

    BookService bookService = (BookService) context.getBean("bookService");


    // Call a method on BookService to test logging

    bookService.performSomeService();


    // Close the context

    ((ClassPathXmlApplicationContext) context).close();

  }

}
```

Exercise 4: Creating and Configuring a Maven Project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.2</version>
    <relativePath/> <!-- lookup parent from repository-->
  </parent>
  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>LibraryManagement</name>
  <description>LibraryManagement</description>
  <url/>
  <licenses>
```

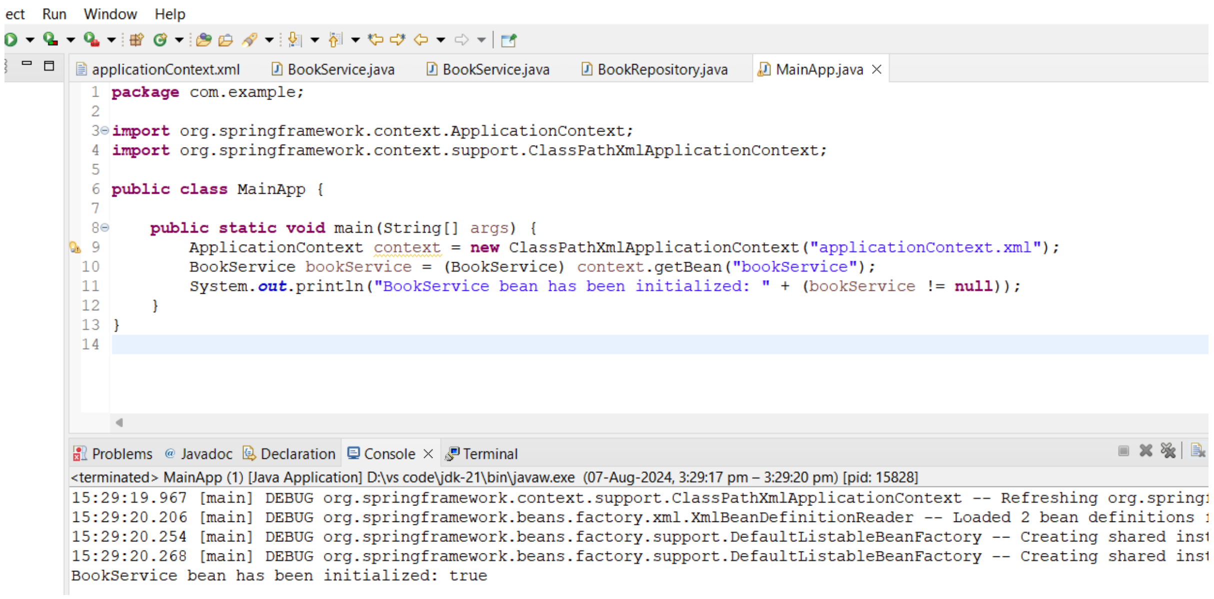
```xml
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <!-- Spring Core Context Dependency-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.25</version>
    </dependency>

    <!-- Spring AOP Dependency-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>5.3.25</version>
    </dependency>

    <!-- Spring WebMVC Dependency (if needed)-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>6.0.7</version>
    </dependency>
  </dependencies>

  <build>
```

```
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

LibraryManagementApplication.java

```
package com.library.librarymanagement;

import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class LibraryManagementApplication {

    public static void main(String[] args) {
        System.out.println("Welcome to Library Management Application!");
    }

}
```

Exercise 5: Configuring the Spring IoC Container



```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        System.out.println("BookService bean has been initialized: " + (bookService != null));
    }
}
```

```
<terminated> MainApp (1) [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 3:29:17 pm – 3:29:20 pm) [pid: 15828]
15:29:19.967 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext -- Refreshing org.spring:
15:29:20.206 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader -- Loaded 2 bean definitions :
15:29:20.254 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared inst
15:29:20.268 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared inst
BookService bean has been initialized: true
```

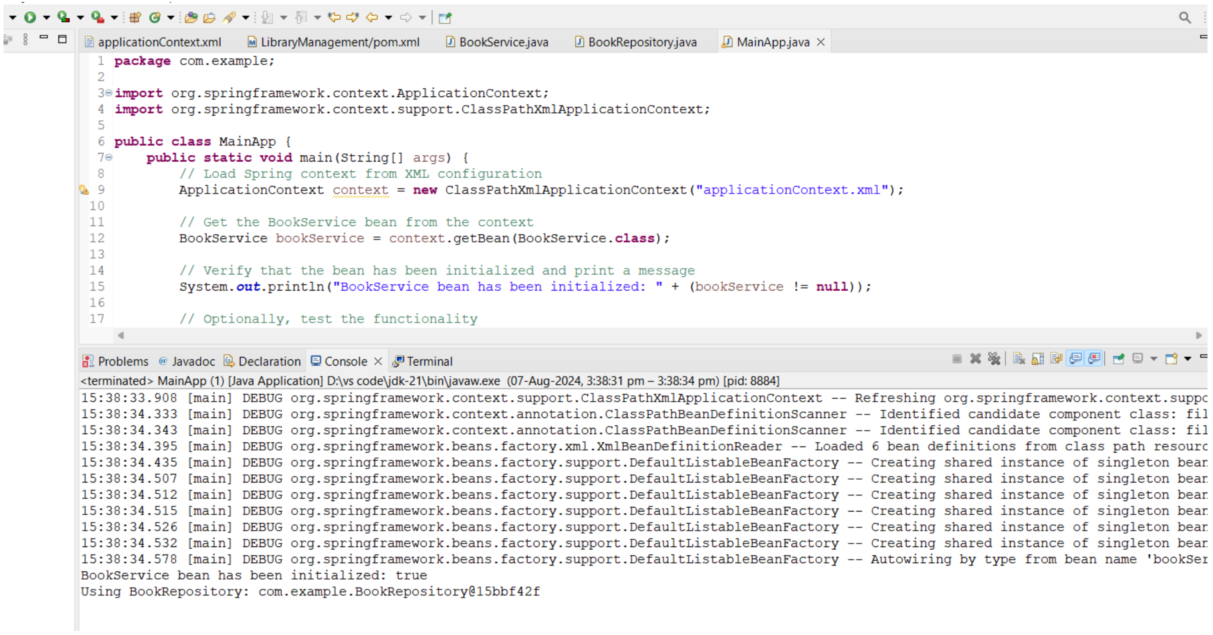
CODE:

```java
// File: src/main/java/com/example/BookService.java
package com.example;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void performService() {
        System.out.println("Using BookRepository: " + bookRepository);
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                     http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Bean definition for BookRepository-->
    <bean id="bookRepository" class="com.example.BookRepository"/>

    <!-- Bean definition for BookService-->
    <bean id="bookService" class="com.example.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>--------------------------------------------------------------------------------------------------------------------
```

```java
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        System.out.println("BookService bean has been initialized: " + (bookService != null));
    }
}
```

Exercise 6: Configuring Beans with Annotations



```java
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean from the context
        BookService bookService = context.getBean(BookService.class);

        // Verify that the bean has been initialized and print a message
        System.out.println("BookService bean has been initialized: " + (bookService != null));

        // Optionally, test the functionality
```

```
<terminated> MainApp (1) [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 3:38:31 pm – 3:38:34 pm) [pid: 8884]
15:38:33.908 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext -- Refreshing org.springframework.context.suppo
15:38:34.333 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner -- Identified candidate component class: fil
15:38:34.343 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner -- Identified candidate component class: fil
15:38:34.395 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader -- Loaded 6 bean definitions from class path resourc
15:38:34.435 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.507 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.512 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.515 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.526 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.532 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating shared instance of singleton bear
15:38:34.578 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Autowiring by type from bean name 'bookSer
BookService bean has been initialized: true
Using BookRepository: com.example.BookRepository@15bbf42f
```

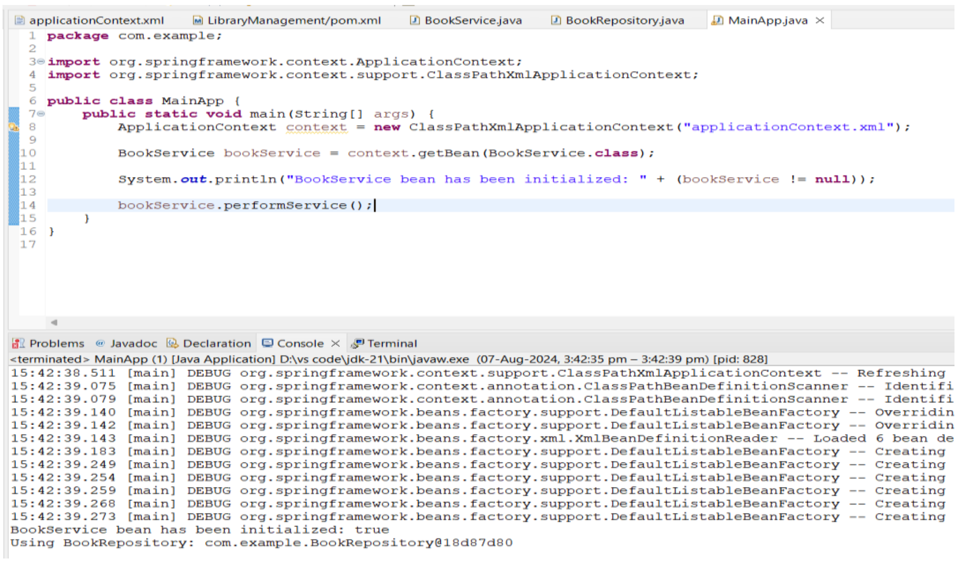
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Enable component scanning for the com.example package-->
    <context:component-scan base-package="com.example"/>

</beans>
```

```java
package com.example;

import org.springframework.stereotype.Service;

@Service
public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void performService() {
        System.out.println("Using BookRepository: " + bookRepository);
    }
}
```

```
package com.example;

import org.springframework.stereotype.Repository;

@Repository
public class BookRepository {
    // Repository methods
}
```

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean from the context
        BookService bookService = context.getBean(BookService.class);

        // Verify that the bean has been initialized and print a message
        System.out.println("BookService bean has been initialized: " + (bookService != null));

        // Optionally, test the functionality
        bookService.performService();
    }
}
```

Exercise 7: Implementing Constructor and Setter Injection



```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean(BookService.class);

        System.out.println("BookService bean has been initialized: " + (bookService != null));

        bookService.performService();
    }
}
```

```
<terminated> MainApp (1) [Java Application] D:\vs code\jdk-21\bin\javaw.exe  (07-Aug-2024, 3:42:35 pm – 3:42:39 pm) [pid: 828]
15:42:38.511 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext -- Refreshing
15:42:39.075 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner -- Identifi
15:42:39.079 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner -- Identifi
15:42:39.140 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Overridin
15:42:39.142 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Overridin
15:42:39.143 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader -- Loaded 6 bean de
15:42:39.183 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
15:42:39.249 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
15:42:39.254 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
15:42:39.259 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
15:42:39.268 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
15:42:39.273 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory -- Creating
BookService bean has been initialized: true
Using BookRepository: com.example.BookRepository@18d87d80
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                          http://www.springframework.org/schema/beans/spring-beans.xsd
                          http://www.springframework.org/schema/context
                          http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Enable component scanning for the com.example package-->
    <context:component-scan base-package="com.example"/>

    <!-- Define BookService with constructor injection-->
    <bean id="bookService" class="com.example.BookService">
        <constructor-arg ref="bookRepository"/>
    </bean>

    <!-- Define BookRepository as a bean-->
    <bean id="bookRepository" class="com.example.BookRepository"/>

</beans>
```

```
package com.example;

import org.springframework.stereotype.Service;

@Service
public class BookService {

    private BookRepository bookRepository;

    // Constructor injection
    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    // Setter injection
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void performService() {
        System.out.println("Using BookRepository: " + bookRepository);
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                  http://www.springframework.org/schema/beans/spring-beans.xsd
                  http://www.springframework.org/schema/context
                  http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Enable component scanning for the com.example package-->
    <context:component-scan base-package="com.example"/>

    <!-- Define BookService with constructor injection-->
    <bean id="bookService" class="com.example.BookService">
        <constructor-arg ref="bookRepository"/>
        <!-- Optionally configure setter injection-->
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

    <!-- Define BookRepository as a bean-->
    <bean id="bookRepository" class="com.example.BookRepository"/>

</beans>
```

Exercise 8: Implementing Basic AOP with Spring

```
package com.library.aspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LoggingAspect {

    @Before("execution(* com.example.service.BookService.*(..))")
    public void logBefore(JoinPoint joinPoint) {
        System.out.println("Before method: " + joinPoint.getSignature().getName());
    }

    @After("execution(* com.example.service.BookService.*(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("After method: " + joinPoint.getSignature().getName());
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <context:component-scan base-package="com.example"/>

    <aop:aspectj-autoproxy/>
    <bean id="bookService" class="com.example.BookService">
        <constructor-arg ref="bookRepository"/>
    </bean>
    <bean id="bookRepository" class="com.example.BookRepository"/>
    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

</beans>
```

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = context.getBean(BookService.class);
        bookService.performService();
    }
}
```

Exercise 9: Creating a Spring Boot Application

Pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
```

```xml
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

```properties
# Server port
server.port=8080

# H2 Database configuration
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# H2 Console configuration
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

```java
package com.example.librarymanagement.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String author;

    // Getters and setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

```
package com.example.librarymanagement.repository;

import com.example.librarymanagement.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {
}
```

```
package com.example.librarymanagement.controller;

import com.example.librarymanagement.entity.Book;
import com.example.librarymanagement.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookRepository bookRepository;

    @GetMapping
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Book> getBookById(@PathVariable Long id) {
        Optional<Book> book = bookRepository.findById(id);
        return book.map(ResponseEntity::ok).orElseGet(()->
ResponseEntity.notFound().build());
    }

    @PostMapping
    public Book createBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable Long id,
@RequestBody Book book) {
        if (!bookRepository.existsById(id)) {
            return ResponseEntity.notFound().build();
        }
        book.setId(id);
        return ResponseEntity.ok(bookRepository.save(book));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        if (!bookRepository.existsById(id)) {
            return ResponseEntity.notFound().build();
        }
        bookRepository.deleteById(id);
        return ResponseEntity.noContent().build();
    }
}
```

PLSQL QUESTIONS

Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.
o Question: Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

```
DECLARE
  CURSOR customer_cursor IS
    SELECT c.CustomerID, l.LoanID, l.InterestRate
    FROM Customers c
    JOIN Loans l ON c.CustomerID = l.CustomerID
    WHERE EXTRACT(YEAR FROM SYSDATE)- EXTRACT(YEAR FROM c.DOB) > 60;

BEGIN
  FOR loan_record IN customer_cursor LOOP
    UPDATE Loans
    SET InterestRate = InterestRate- 1
    WHERE LoanID = loan_record.LoanID;

    DBMS_OUTPUT.PUT_LINE('Applied 1% discount to loan ID: ' || loan_record.LoanID);
  END LOOP;
END;
```

Scenario 2: A customer can be promoted to VIP status based on their balance.
o Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

```
ALTER TABLE Customers ADD (IsVIP CHAR(1));
DECLARE
  CURSOR customer_cursor IS
    SELECT CustomerID, Balance
    FROM Customers;

BEGIN
  FOR customer_record IN customer_cursor LOOP
    IF customer_record.Balance > 10000 THEN
      UPDATE Customers
      SET IsVIP = 'Y'
      WHERE CustomerID = customer_record.CustomerID;

      DBMS_OUTPUT.PUT_LINE('Promoted to VIP status for customer ID: ' ||
customer_record.CustomerID);
    ELSE
      UPDATE Customers
```

```
            SET IsVIP = 'N'
            WHERE CustomerID = customer_record.CustomerID;
        END IF;
    END LOOP;
END;
```

Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.
o Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

```
DECLARE
    CURSOR loan_cursor IS
        SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name
        FROM Loans l
        JOIN Customers c ON l.CustomerID = c.CustomerID
        WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN
    FOR loan_record IN loan_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || loan_record.LoanID ||
                             ' for customer ' || loan_record.Name ||
                             ' is due on ' || loan_record.EndDate);
    END LOOP;
END;
```

Exercise 2: Error Handling

Scenario 1: Handle exceptions during fund transfers between accounts.
o Question: Write a stored procedure SafeTransferFunds that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

```
CREATE OR REPLACE PROCEDURE SafeTransferFunds (
    p_from_account IN NUMBER,
    p_to_account IN NUMBER,
    p_amount IN NUMBER
) AS
BEGIN
    BEGIN
        DECLARE
            v_balance NUMBER;
        BEGIN
            SELECT Balance INTO v_balance
            FROM Accounts
            WHERE AccountID = p_from_account;
            IF v_balance < p_amount THEN
                RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in account ' || p_from_account);
            END IF;
        END;
        UPDATE Accounts
        SET Balance = Balance- p_amount
        WHERE AccountID = p_from_account;

        UPDATE Accounts
        SET Balance = Balance + p_amount
        WHERE AccountID = p_to_account;

        COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    END;
END SafeTransferFunds;
```

Scenario 2: Manage errors when updating employee salaries.
o Question: Write a stored procedure UpdateSalary that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

```
CREATE OR REPLACE PROCEDURE UpdateSalary (
    p_employee_id IN NUMBER,
    p_percentage IN NUMBER
) AS
BEGIN
    BEGIN
        UPDATE Employees
        SET Salary = Salary * (1 + p_percentage / 100)
        WHERE EmployeeID = p_employee_id;

        IF SQL%ROWCOUNT = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Employee ID ' || p_employee_id || ' does not exist');
        END IF;

        COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    END;
END UpdateSalary;
```

Scenario 3: Ensure data integrity when adding a new customer.
o Question: Write a stored procedure AddNewCustomer that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

```
CREATE OR REPLACE PROCEDURE AddNewCustomer (
   p_customer_id IN NUMBER,
   p_name IN VARCHAR2,
   p_dob IN DATE,
   p_balance IN NUMBER
) AS
BEGIN
   BEGIN
      INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
      VALUES (p_customer_id, p_name, p_dob, p_balance, SYSDATE);

      COMMIT;

   EXCEPTION
      WHEN DUP_VAL_ON_INDEX THEN
         DBMS_OUTPUT.PUT_LINE('Error: Customer ID ' || p_customer_id || ' already exists');
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
         ROLLBACK;
   END;
END AddNewCustomer;
```

Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.
o Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
   UPDATE Accounts
   SET Balance = Balance * 1.01
   WHERE AccountType = 'Savings';

   COMMIT;

   DBMS_OUTPUT.PUT_LINE('Monthly interest applied to all savings accounts.');
END ProcessMonthlyInterest;
```

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

o	Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_department IN VARCHAR2,
  p_bonus_percentage IN NUMBER
) AS
BEGIN
  UPDATE Employees
  SET Salary = Salary * (1 + p_bonus_percentage / 100)
  WHERE Department = p_department;

  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Bonus applied to all employees in department: ' || p_department);
END UpdateEmployeeBonus;
```

Scenario 3: Customers should be able to transfer funds between their accounts.
o	Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_from_account IN NUMBER,
  p_to_account IN NUMBER,
  p_amount IN NUMBER
) AS
  v_balance NUMBER;
BEGIN
  SELECT Balance INTO v_balance
  FROM Accounts
  WHERE AccountID = p_from_account;

  IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in account ' || p_from_account);
  END IF;

  BEGIN
    UPDATE Accounts
    SET Balance = Balance- p_amount
    WHERE AccountID = p_from_account;

    UPDATE Accounts
    SET Balance = Balance + p_amount
    WHERE AccountID = p_to_account;

    COMMIT;
```

```
            DBMS_OUTPUT.PUT_LINE('Transfer of ' || p_amount || ' from account ' || p_from_account
|| ' to account ' || p_to_account || ' completed successfully.');
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    END;
END TransferFunds;
```

Exercise 4: Functions

Scenario 1: Calculate the age of customers for eligibility checks.
o Question: Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

```
CREATE OR REPLACE FUNCTION CalculateAge(p_dob DATE)
RETURN NUMBER
IS
    v_age NUMBER;
BEGIN
    SELECT FLOOR(MONTHS_BETWEEN(SYSDATE, p_dob) / 12) INTO v_age FROM dual;
    RETURN v_age;
EXCEPTION
    WHEN OTHERS THEN
        RETURN NULL;
END;
```

Scenario 2: The bank needs to compute the monthly installment for a loan.
o Question: Write a function CalculateMonthlyInstallment that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

```
CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(
    p_loan_amount NUMBER,
    p_annual_interest_rate NUMBER,
    p_loan_duration_years NUMBER
)
RETURN NUMBER
IS
    v_monthly_interest_rate NUMBER;
    v_number_of_months NUMBER;
    v_monthly_installment NUMBER;
BEGIN
    v_monthly_interest_rate := p_annual_interest_rate / 12 / 100;
    v_number_of_months := p_loan_duration_years * 12;

    IF v_monthly_interest_rate > 0 THEN
        v_monthly_installment := (p_loan_amount * v_monthly_interest_rate) /
```

```
                            (1- POWER(1 + v_monthly_interest_rate,-v_number_of_months));
    ELSE
        v_monthly_installment := p_loan_amount / v_number_of_months;
    END IF;

    RETURN v_monthly_installment;
EXCEPTION
    WHEN OTHERS THEN
        RETURN NULL;
END;
```

Scenario 3: Check if a customer has sufficient balance before making a transaction.
o Question: Write a function HasSufficientBalance that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

```
CREATE OR REPLACE FUNCTION HasSufficientBalance(
    p_account_id NUMBER,
    p_amount NUMBER
)
RETURN BOOLEAN
IS
    v_balance NUMBER;
BEGIN
    SELECT Balance INTO v_balance
    FROM Accounts
    WHERE AccountID = p_account_id;

    RETURN v_balance >= p_amount;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN FALSE;
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

Exercise 5: Triggers

Scenario 1: Automatically update the last modified date when a customer's record is updated.
o Question: Write a trigger UpdateCustomerLastModified that updates the LastModified column of the Customers table to the current date whenever a customer's record is updated.

```
CREATE OR REPLACE TRIGGER UpdateCustomerLastModified
BEFORE UPDATE ON Customers
FOR EACH ROW
BEGIN
```

```
   :NEW.LastModified := SYSDATE;
END;
```

Scenario 2: Maintain an audit log for all transactions.
o Question: Write a trigger LogTransaction that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.

```
CREATE TABLE AuditLog (
   AuditID NUMBER PRIMARY KEY,
   TransactionID NUMBER,
   ChangeDate DATE,
   ChangeType VARCHAR2(50)
);

CREATE SEQUENCE AuditLogSeq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE OR REPLACE TRIGGER LogTransaction
AFTER INSERT ON Transactions
FOR EACH ROW
BEGIN
   INSERT INTO AuditLog (AuditID, TransactionID, ChangeDate, ChangeType)
   VALUES (AuditLogSeq.NEXTVAL, :NEW.TransactionID, SYSDATE, 'INSERT');
END;
```

Scenario 3: Enforce business rules on deposits and withdrawals.
o Question: Write a trigger CheckTransactionRules that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

```
CREATE OR REPLACE TRIGGER CheckTransactionRules
BEFORE INSERT ON Transactions
FOR EACH ROW
DECLARE
   v_balance NUMBER;
BEGIN
   IF :NEW.TransactionType = 'Withdrawal' THEN
      SELECT Balance INTO v_balance
      FROM Accounts
      WHERE AccountID = :NEW.AccountID;

      IF v_balance < :NEW.Amount THEN
         RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds for withdrawal');
      END IF;
   END IF;
```

```
  IF :NEW.TransactionType = 'Deposit' THEN
    IF :NEW.Amount <= 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'Deposit amount must be positive');
    END IF;
  END IF;
END;
```

Exercise 6: Cursors

Scenario 1: Generate monthly statements for all customers.
o Question: Write a PL/SQL block using an explicit cursor GenerateMonthlyStatements that retrieves all transactions for the current month and prints a statement for each customer.

```
DECLARE
  CURSOR cur_transactions IS
    SELECT c.CustomerID, c.Name, t.TransactionDate, t.Amount, t.TransactionType
    FROM Customers c
    JOIN Accounts a ON c.CustomerID = a.CustomerID
    JOIN Transactions t ON a.AccountID = t.AccountID
    WHERE t.TransactionDate BETWEEN TRUNC(SYSDATE, 'MM') AND LAST_DAY(SYSDATE);
  v_customerID Customers.CustomerID%TYPE;
  v_name Customers.Name%TYPE;
  v_transactionDate Transactions.TransactionDate%TYPE;
  v_amount Transactions.Amount%TYPE;
  v_transactionType Transactions.TransactionType%TYPE;
BEGIN
  OPEN cur_transactions;
  LOOP
    FETCH cur_transactions INTO v_customerID, v_name, v_transactionDate, v_amount,
v_transactionType;
    EXIT WHEN cur_transactions%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Customer: ' || v_name || ' (' || v_customerID || ')');
    DBMS_OUTPUT.PUT_LINE('Transaction Date: ' || v_transactionDate);
    DBMS_OUTPUT.PUT_LINE('Amount: ' || v_amount || ' Type: ' || v_transactionType);
    DBMS_OUTPUT.PUT_LINE('-----------------------------');
  END LOOP;
  CLOSE cur_transactions;
END;
```

Scenario 2: Apply annual fee to all accounts.
o Question: Write a PL/SQL block using an explicit cursor ApplyAnnualFee that deducts an annual maintenance fee from the balance of all accounts.

```
DECLARE
  CURSOR cur_accounts IS
    SELECT AccountID, Balance
    FROM Accounts;
```

```
   v_accountID Accounts.AccountID%TYPE;
   v_balance Accounts.Balance%TYPE;
   v_annualFee CONSTANT NUMBER := 100;
BEGIN
   OPEN cur_accounts;
   LOOP
      FETCH cur_accounts INTO v_accountID, v_balance;
      EXIT WHEN cur_accounts%NOTFOUND;
      UPDATE Accounts
      SET Balance = Balance - v_annualFee
      WHERE AccountID = v_accountID;
      DBMS_OUTPUT.PUT_LINE('Account ID: ' || v_accountID || ' New Balance: ' || (v_balance - 
v_annualFee));
   END LOOP;
   CLOSE cur_accounts;
END;
```

Scenario 3: Update the interest rate for all loans based on a new policy.
o Question: Write a PL/SQL block using an explicit cursor UpdateLoanInterestRates that fetches all loans and updates their interest rates based on the new policy.

```
DECLARE
   CURSOR cur_loans IS
      SELECT LoanID, InterestRate
      FROM Loans;
   v_loanID Loans.LoanID%TYPE;
   v_interestRate Loans.InterestRate%TYPE;
   v_newInterestRate CONSTANT NUMBER := 5;
BEGIN
   OPEN cur_loans;
   LOOP
      FETCH cur_loans INTO v_loanID, v_interestRate;
      EXIT WHEN cur_loans%NOTFOUND;
      UPDATE Loans
      SET InterestRate = v_newInterestRate
      WHERE LoanID = v_loanID;
      DBMS_OUTPUT.PUT_LINE('Loan ID: ' || v_loanID || ' New Interest Rate: ' || v_newInterestRate);
   END LOOP;
   CLOSE cur_loans;
END;
```

Exercise 7: Packages

Scenario 1: Group all customer-related procedures and functions into a package.
o Question: Create a package CustomerManagement with procedures for adding a new customer, updating customer details, and a function to get customer balance.

```
CREATE OR REPLACE PACKAGE CustomerManagement AS
  PROCEDURE AddCustomer(p_CustomerID NUMBER, p_Name VARCHAR2, p_DOB DATE, p_Balance
NUMBER);
  PROCEDURE UpdateCustomer(p_CustomerID NUMBER, p_Name VARCHAR2, p_DOB DATE,
p_Balance NUMBER);
  FUNCTION GetCustomerBalance(p_CustomerID NUMBER) RETURN NUMBER;
END CustomerManagement;

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS
  PROCEDURE AddCustomer(p_CustomerID NUMBER, p_Name VARCHAR2, p_DOB DATE, p_Balance
NUMBER) IS
  BEGIN
    INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
    VALUES (p_CustomerID, p_Name, p_DOB, p_Balance, SYSDATE);
  EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Customer with this ID already exists.');
  END AddCustomer;

  PROCEDURE UpdateCustomer(p_CustomerID NUMBER, p_Name VARCHAR2, p_DOB DATE,
p_Balance NUMBER) IS
  BEGIN
    UPDATE Customers
    SET Name = p_Name, DOB = p_DOB, Balance = p_Balance, LastModified = SYSDATE
    WHERE CustomerID = p_CustomerID;
    IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Customer not found.');
    END IF;
  END UpdateCustomer;

  FUNCTION GetCustomerBalance(p_CustomerID NUMBER) RETURN NUMBER IS
    v_balance NUMBER;
  BEGIN
    SELECT Balance INTO v_balance
    FROM Customers
    WHERE CustomerID = p_CustomerID;
    RETURN v_balance;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN NULL;
  END GetCustomerBalance;
END CustomerManagement;
```

Scenario 2: Create a package to manage employee data.
o Question: Write a package EmployeeManagement with procedures to hire new employees, update employee details, and a function to calculate annual salary.

```
CREATE OR REPLACE PACKAGE EmployeeManagement AS
```

```
   PROCEDURE HireEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position VARCHAR2,
p_Salary NUMBER, p_Department VARCHAR2, p_HireDate DATE);
   PROCEDURE UpdateEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2);
   FUNCTION CalculateAnnualSalary(p_EmployeeID NUMBER) RETURN NUMBER;
END EmployeeManagement;

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS
   PROCEDURE HireEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position VARCHAR2,
p_Salary NUMBER, p_Department VARCHAR2, p_HireDate DATE) IS
   BEGIN
      INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)
      VALUES (p_EmployeeID, p_Name, p_Position, p_Salary, p_Department, p_HireDate);
   EXCEPTION
      WHEN DUP_VAL_ON_INDEX THEN
         DBMS_OUTPUT.PUT_LINE('Employee with this ID already exists.');
   END HireEmployee;

   PROCEDURE UpdateEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2) IS
   BEGIN
      UPDATE Employees
      SET Name = p_Name, Position = p_Position, Salary = p_Salary, Department = p_Department
      WHERE EmployeeID = p_EmployeeID;
      IF SQL%ROWCOUNT = 0 THEN
         DBMS_OUTPUT.PUT_LINE('Employee not found.');
      END IF;
   END UpdateEmployee;

   FUNCTION CalculateAnnualSalary(p_EmployeeID NUMBER) RETURN NUMBER IS
      v_salary NUMBER;
   BEGIN
      SELECT Salary INTO v_salary
      FROM Employees
      WHERE EmployeeID = p_EmployeeID;
      RETURN v_salary * 12;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RETURN NULL;
   END CalculateAnnualSalary;
END EmployeeManagement;
```

Scenario 3: Group all account-related operations into a package.

o Question: Create a package AccountOperations with procedures for opening a new account, closing an account, and a function to get the total balance of a customer across all accounts.

```
CREATE OR REPLACE PACKAGE AccountOperations AS
```

```
  PROCEDURE OpenAccount(p_AccountID NUMBER, p_CustomerID NUMBER, p_AccountType
VARCHAR2, p_Balance NUMBER);
  PROCEDURE CloseAccount(p_AccountID NUMBER);
  FUNCTION GetTotalBalance(p_CustomerID NUMBER) RETURN NUMBER;
END AccountOperations;

CREATE OR REPLACE PACKAGE BODY AccountOperations AS
  PROCEDURE OpenAccount(p_AccountID NUMBER, p_CustomerID NUMBER, p_AccountType
VARCHAR2, p_Balance NUMBER) IS
  BEGIN
    INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)
    VALUES (p_AccountID, p_CustomerID, p_AccountType, p_Balance, SYSDATE);
  EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Account with this ID already exists.');
  END OpenAccount;

  PROCEDURE CloseAccount(p_AccountID NUMBER) IS
  BEGIN
    DELETE FROM Accounts
    WHERE AccountID = p_AccountID;
    IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Account not found.');
    END IF;
  END CloseAccount;

  FUNCTION GetTotalBalance(p_CustomerID NUMBER) RETURN NUMBER IS
    v_totalBalance NUMBER;
  BEGIN
    SELECT SUM(Balance) INTO v_totalBalance
    FROM Accounts
    WHERE CustomerID = p_CustomerID;
    RETURN v_totalBalance;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN 0;
  END GetTotalBalance;
END AccountOperations;
```