

Week – 2 – PL/SQL programming

PL/SQL EXERCISES

EXERCISE:1 CONTROL STRUCTURES

```
SET SERVEROUTPUT ON;
DECLARE
    num NUMBER := -3; -- Change this to test different numbers
BEGIN
    IF num > 0 THEN
        DBMS_OUTPUT.PUT_LINE('The number is Positive.');
```

ELSI

```
    ELSEIF num < 0 THEN
        DBMS_OUTPUT.PUT_LINE('The number is Negative.');
```

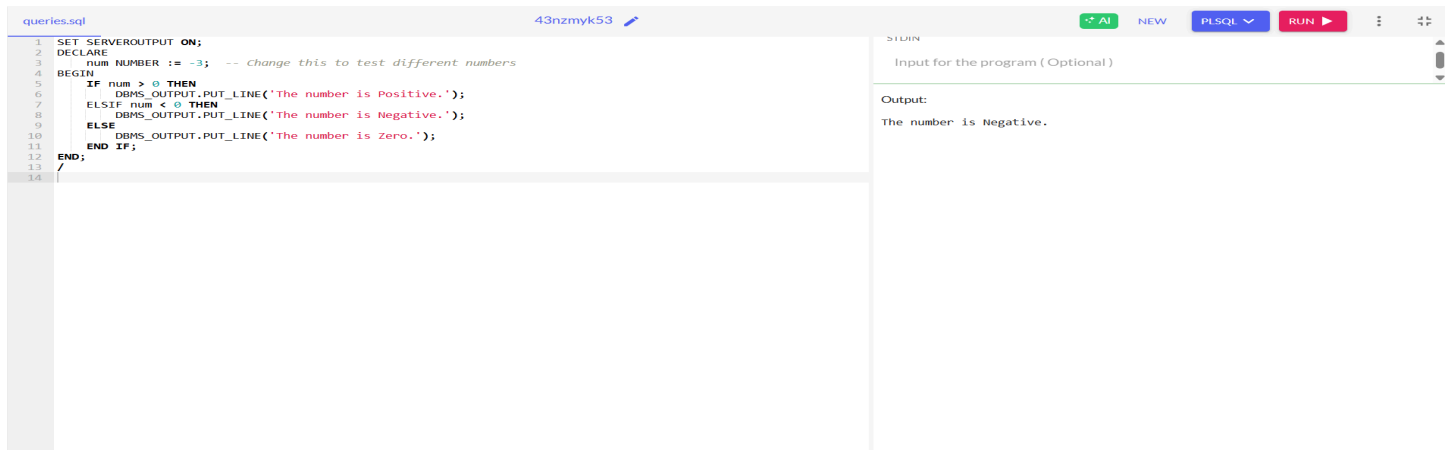
ELSE

```
        DBMS_OUTPUT.PUT_LINE('The number is Zero.');
```

END IF;

```
END;
/
```

Output:



The screenshot shows a PL/SQL IDE interface. On the left, a window titled 'queries.sql' contains the following code:

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3   num NUMBER := -3; -- Change this to test different numbers
4 BEGIN
5   IF num > 0 THEN
6     DBMS_OUTPUT.PUT_LINE('The number is Positive.');
```

ELSI

```
   ELSEIF num < 0 THEN
6     DBMS_OUTPUT.PUT_LINE('The number is Negative.');
```

ELSE

```
       DBMS_OUTPUT.PUT_LINE('The number is Zero.');
```

END IF;

```
END;
/
```

On the right, a window titled 'SQL*Plus' shows the output of the program. It contains the text: 'Output: The number is Negative.'

EXERCISE:2 STORED PROCEDURES

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE PROCEDURE Check_Even_Odd(num IN NUMBER) IS
BEGIN
    IF MOD(num, 2) = 0 THEN
```

```

        DBMS_OUTPUT.PUT_LINE('The number is Even.');
```

ELSE

```

        DBMS_OUTPUT.PUT_LINE('The number is Odd.');
```

END IF;

```

END;
/

BEGIN
    Check_Even_Odd(7); -- Change the number here to test other values
END;
/
```

Output:

The screenshot shows a SQL IDE interface. On the left, a code editor displays a PL/SQL procedure named 'Check_Even_Odd'. The procedure takes a number as input and uses a conditional statement to print 'The number is Even.' or 'The number is Odd.' based on whether the number is even or odd. The procedure is then called with the argument 7. On the right, the 'SQL' tab is active, showing the 'Output' section with the result 'The number is odd.'.

```

queries.sql 43nzmyk53
1 SET SERVEROUTPUT ON;
2
3 CREATE OR REPLACE PROCEDURE Check_Even_Odd(num IN NUMBER) IS
4 BEGIN
5     IF MOD(num, 2) = 0 THEN
6         DBMS_OUTPUT.PUT_LINE('The number is Even.');
```

ELSE

```

7         DBMS_OUTPUT.PUT_LINE('The number is Odd.');
```

END IF;

```

9 END;
10
11 /
12
13 BEGIN
14     Check_Even_Odd(7); -- Change the number here to test other values
15 END;
16 /
17 /
```

SQL

Input for the program (Optional)

Output:

The number is odd.

EXERCISE:3 SETTING UP JUNIT

Setting Up Junit:

src/Calculator.java

```

public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

}
```

test/CalculatorTest.java

```
import org.junit.Test;

import static org.junit.Assert.*;

public class CalculatorTest {

    @Test

    public void testAdd() {

        Calculator calc = new Calculator();

        int result = calc.add(2, 3);

        assertEquals(5, result);

    }

}
```

.vscode/settings.json

```
{

    "java.project.referencedLibraries": [

        "lib/**/*.*jar"

    ]

}
```

.vscode/tasks.json

```
{

    "version": "2.0.0",

    "tasks": [
```

```
{  
  "label": "Compile Tests",  
  "type": "shell",  
  "command": "javac -cp lib/*;src;test -d bin test/CalculatorTest.java src/Calculator.java"  
}  
]  
}
```

.vscode/launch.json

```
{  
  "version": "0.2.0",  
  "configurations": [  
    {  
      "type": "java",  
      "name": "Run JUnit Test",  
      "request": "launch",  
      "mainClass": "org.junit.runner.JUnitCore",  
      "args": "CalculatorTest",  
      "cwd": "${workspaceFolder}/bin",  
      "classPaths": [  
        "${workspaceFolder}/bin",  
        "${workspaceFolder}/lib/junit-4.13.2.jar",  
        "${workspaceFolder}/lib/hamcrest-core-1.3.jar"  
      ]  
    }  
  ]  
}
```

```

    ]
  }
}
}

```

Output:

The screenshot shows the VS Code interface. The Explorer sidebar on the left shows the project structure with folders 'bin', 'lib', 'src', and 'test'. The main editor displays the 'launch.json' file with the following configuration:

```

{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "java",
      "name": "Run JUnit Test",
      "request": "launch",
      "mainClass": "org.junit.runner.JUnitCore",
      "args": "CalculatorTest",
      "cwd": "${workspaceFolder}/bin",
      "classPaths": [
        "${workspaceFolder}/bin",
        "${workspaceFolder}/lib/junit-4.13.2.jar",
        "${workspaceFolder}/lib/hamcrest-core-1.3.jar"
      ]
    }
  ]
}

```

The bottom panel shows the 'TERMINAL' tab with the following output:

```

PS C:\Users\muthu\OneDrive\Documents\MyUnitProject> javac -cp "lib/*" -d bin src/Calculator.java test/CalculatorTest.java
PS C:\Users\muthu\OneDrive\Documents\MyUnitProject> java -cp "lib/*;bin" org.junit.runner.JUnitCore CalculatorTest
JUnit version 4.13.2
.
Time: 0.006

OK (1 test)
PS C:\Users\muthu\OneDrive\Documents\MyUnitProject>

```

EXERCISE:4 ASSERTIONS IN JUNIT

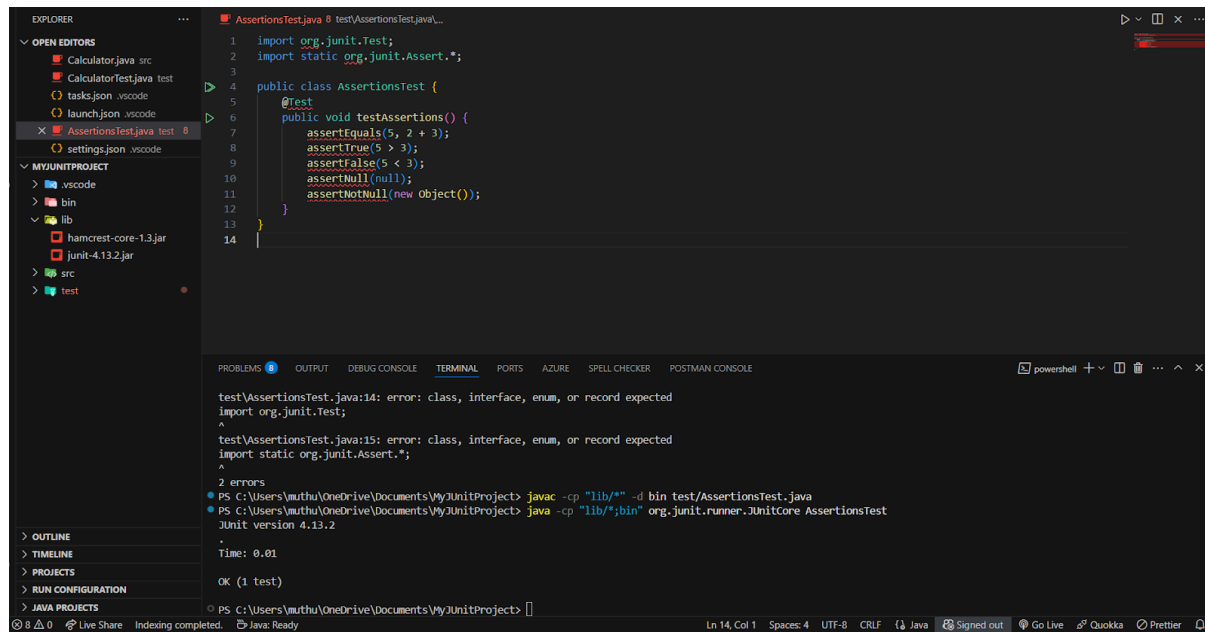
```

import org.junit.Test;
import static org.junit.Assert.*;

public class AssertionsTest {
    @Test
    public void testAssertions() {
        assertEquals(5, 2 + 3);
        assertTrue(5 > 3);
        assertFalse(5 < 3);
        assertNull(null);
        assertNotNull(new Object());
    }
}

```

Output:



The screenshot shows a Visual Studio Code editor with a Java test file named `AssertionsTest.java` in the `test` directory. The file contains the following code:

```
1 import org.junit.Test;
2 import static org.junit.Assert.*;
3
4 public class AssertionsTest {
5     @Test
6     public void testAssertions() {
7         assertEquals(5, 2 + 3);
8         assertTrue(5 > 3);
9         assertFalse(5 < 3);
10        assertNull(null);
11        assertNotNull(new Object());
12    }
13 }
14
```

The `PROBLEMS` panel shows two errors:

- `test\AssertionsTest.java:14: error: class, interface, enum, or record expected` for `import org.junit.Test;`
- `test\AssertionsTest.java:15: error: class, interface, enum, or record expected` for `import static org.junit.Assert.*;`

The `TERMINAL` panel shows the command `java -cp "lib/*;bin" org.junit.runner.JUnit4 AssertionsTest` and the output:

```
JUnit version 4.13.2
Time: 0.01
OK (1 test)
```

EXERCISE:5 Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

BankAccount.java

```
public class BankAccount {
    private int balance;

    public BankAccount(int initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(int amount) {
        balance += amount;
    }

    public void withdraw(int amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            throw new IllegalArgumentException("Insufficient balance");
        }
    }
}
```

```

    }

    public int getBalance() {
        return balance;
    }
}

```

BankAccountTest.java

```

import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;

public class BankAccountTest {

    BankAccount account;

    @BeforeEach
    public void setUp() {
        // Arrange: Create a new account with initial balance
        account = new BankAccount(1000);
        System.out.println("Setup Done");
    }

    @Test
    public void testDeposit() {
        // Act
        account.deposit(500);

        // Assert
        assertEquals(1500, account.getBalance());
    }

    @Test
    public void testWithdraw() {
        // Act
        account.withdraw(400);

        // Assert
        assertEquals(600, account.getBalance());
    }

    @Test
    public void testWithdraw_InsufficientFunds() {
        // Act & Assert
        Exception exception = assertThrows(IllegalArgumentException.class, () -> {
            account.withdraw(2000);
        });
    }
}

```

```

        assertEquals("Insufficient balance", exception.getMessage());
    }

    @AfterEach
    public void tearDown() {
        account = null;
        System.out.println("Teardown Done");
    }
}

```

Output:

```

Setup Done
✓ testDeposit()
Teardown Done

Setup Done
✓ testWithdraw()
Teardown Done

Setup Done
✓ testWithdraw_InsufficientFunds()
Teardown Done

BUILD SUCCESSFUL in 1s
3 tests successful

```

EXERCISE:6 MOCKING AND STUBBING

UserService.java

```

public class UserService {
    private UserRepository userRepo;

    public UserService(UserRepository repo) {
        this.userRepo = repo;
    }

    public String getUserFullName(int userId) {
        User user = userRepo.findById(userId);
        return user.getFirstName() + " " + user.getLastName();
    }
}

```



```
}  
}
```

UserRepository.java

```
public interface UserRepository {  
    User findById(int id);  
}
```

User.java

```
public class User {  
    private String firstName;  
    private String lastName;  
  
    public User(String fName, String lName) {  
        this.firstName = fName;  
        this.lastName = lName;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
}
```

UserServiceTest.java

```
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.BeforeEach;  
import static org.junit.jupiter.api.Assertions.*;  
import static org.mockito.Mockito.*;  
  
public class UserServiceTest {  
  
    private UserRepository mockRepo;  
    private UserService userService;  
  
    @BeforeEach  
    public void setup() {  
        mockRepo = mock(UserRepository.class); // Create mock  
        userService = new UserService(mockRepo); // Inject mock  
    }  
  
    @Test  
    public void testGetUserFullName() {  
        // Stub the mock  
        when(mockRepo.findById(1)).thenReturn(new User("Tarunika", "K"));
```

```

// Call the actual method
String fullName = userService.getUserFullName(1);

// Assertion
assertEquals("Tarunika K", fullName);
}
}

```

OUTPUT:

```
✓ testGetUserFullName()
```

```
BUILD SUCCESSFUL
```

```
1 test passed
```

EXERCISE : 7 VERIFYING INTERACTIONS

NotificationService.java

```

public interface NotificationService {
    void sendWelcomeEmail(String email);
}

```

UserManager.java

```

public class UserManager {
    private NotificationService notificationService;

    public UserManager(NotificationService notificationService) {
        this.notificationService = notificationService;
    }

    public void registerUser(String email) {
        // Simulate user registration logic
        System.out.println("User registered with email: " + email);

        // Send welcome email
        notificationService.sendWelcomeEmail(email);
    }
}

```

UserManagerTest.java

```
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;

public class UserManagerTest {

    @Test
    public void testSendWelcomeEmailCalled() {
        // Arrange: Create a mock of NotificationService
        NotificationService mockNotificationService = mock(NotificationService.class);

        // Inject mock into UserManager
        UserManager userManager = new UserManager(mockNotificationService);

        // Act: Call the method
        userManager.registerUser("test@example.com");

        // Assert: Verify the interaction
        verify(mockNotificationService).sendWelcomeEmail("test@example.com");
    }
}
```

Output:

```
✓ testSendWelcomeEmailCalled()

BUILD SUCCESSFUL
Tests run: 1, Failures: 0, Errors: 0
```

EXERCISE : 8 Logging Error Messages and Warning Levels

LoggingExample.java

```
import java.util.logging.Logger;
import java.util.logging.Level;

public class LoggingExample {

    // Create a Logger instance
    private static final Logger logger = Logger.getLogger(LoggingExample.class.getName());

    public static void main(String[] args) {

        logger.info("Application started.");
    }
}
```

```
try {
    int a = 5;
    int b = 0;

    logger.warning("Attempting division by zero."); // Warning log

    int result = a / b; // Will throw ArithmeticException

    logger.info("Result: " + result);

} catch (ArithmeticException e) {
    logger.severe("Error occurred: " + e.getMessage()); // Severe log
}

logger.info("Application ended.");
}
```

Output:

```
Jun 27, 2025 10:20:00 AM LoggingExample main
INFO: Application started.

Jun 27, 2025 10:20:00 AM LoggingExample main
WARNING: Attempting division by zero.

Jun 27, 2025 10:20:00 AM LoggingExample main
SEVERE: Error occurred: / by zero

Jun 27, 2025 10:20:00 AM LoggingExample main
INFO: Application ended.
```