# Week 2 Java Assignment

## 1. Is calculateTax() a pure function?

No, calculateTax() is not a pure function because it uses the instance variable 'rate'. To make it pure, you can pass 'rate' as a parameter to the method:  
  
code:   
public class TaxUtil {  
 public double calculateTax(double amount, double rate) {  
 return amount \* rate;  
 }  
}

## 2. Output for Static Method

Output:  
A screenshot of a computer

AI-generated content may be incorrect.  
super class show method  
sub class show method

## 3. Output for ThisUse class

A screenshot of a computer

AI-generated content may be incorrect.

\*\*Both this.display() and display() call the subclass method. this.num accesses the subclass variable.

## 4. Singleton Design Pattern

The Singleton Method Design Pattern ensures a class has only one instance and provides a global access point to it. It’s ideal for scenarios requiring centralized control, like managing database connections or configuration settings. This article explores its principles, benefits, drawbacks, and best use cases in software development.

Code:  
public class Singleton {  
 private static Singleton instance;  
 private Singleton() {}  
 public static Singleton getInstance() {  
 if (instance == null) {  
 instance = new Singleton();  
 }  
 return instance;  
 }  
}

## 5. Encapsulation in Java

o achieve encapsulation, you typically:

1. Declare data members as private: This restricts direct access to the data from outside the class.
2. Provide public methods (getters and setters): These methods act as intermediaries, allowing controlled access to the data. Getters retrieve data values, while setters modify them.
3. Implement data validation: Within the setter methods, you can validate data before assigning it to the object’s properties, ensuring data integrity.

Code:  
public class Employee {  
 private int id;  
 private String name;  
 public int getId() { return id; }  
 public void setId(int id) { this.id = id; }  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; }  
}  
  
Explanation: Encapsulation hides class internals using private fields and public accessors.

## 6. CRUD with ArrayList

Code:   
import java.util.\*;  
class Employee {  
 private int id;  
 private String name;  
 private String department;  
 // Constructors, getters and setters  
}  
  
class EmployeeCRUD {  
 private List<Employee> employees = new ArrayList<>();  
 public void addEmployee(Employee e) {  
 employees.add(e);  
 }  
 public void getEmployees() {  
 for (Employee e : employees) {  
 System.out.println(e.getId() + " " + e.getName());  
 }  
 }  
 public void updateEmployee(int id, String name) {  
 for (Employee e : employees) {  
 if (e.getId() == id) {  
 e.setName(name);  
 break;  
 }  
 }  
 }  
 public void deleteEmployee(int id) {  
 employees.removeIf(e -> e.getId() == id);  
 }  
}

## 7. CRUD with JDBC

Code:   
import java.sql.\*;  
public class EmployeeJDBC {  
 Connection con;  
 public EmployeeJDBC() throws Exception {  
 con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "user", "pass");  
 }  
 public void addEmployee(Employee e) throws Exception {  
 PreparedStatement ps = con.prepareStatement("INSERT INTO employee VALUES (?, ?, ?)");  
 ps.setInt(1, e.getId());  
 ps.setString(2, e.getName());  
 ps.setString(3, e.getDepartment());  
 ps.executeUpdate();  
 }  
 public void getEmployees() throws Exception {  
 ResultSet rs = con.createStatement().executeQuery("SELECT \* FROM employee");  
 while (rs.next()) {  
 System.out.println(rs.getInt(1) + " " + rs.getString(2));  
 }  
 }  
 public void updateEmployee(int id, String name) throws Exception {  
 PreparedStatement ps = con.prepareStatement("UPDATE employee SET name=? WHERE id=?");  
 ps.setString(1, name);  
 ps.setInt(2, id);  
 ps.executeUpdate();  
 }  
 public void deleteEmployee(int id) throws Exception {  
 PreparedStatement ps = con.prepareStatement("DELETE FROM employee WHERE id=?");  
 ps.setInt(1, id);  
 ps.executeUpdate();  
 }  
}