**Exercise 1:**

**Objective:** Practice using the **public** access specifier.

1. Create a class named **PublicClass**.
2. Declare a public variable called **publicVariable** of type **int** with an initial value of 42.
3. Implement a public method called **publicMethod** that prints "Public method called."
4. In the **MainClass**, create an instance of **PublicClass** and use the public variable and method.

**Solution 1:**

// PublicClass.java

public class PublicClass {

public int publicVariable = 42;

public void publicMethod() {

System.out.println("Public method called.");

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

PublicClass publicObj = new PublicClass();

System.out.println("Public Variable: " + publicObj.publicVariable);

publicObj.publicMethod();

}

}

**Exercise 2:**

**Objective:** Practice using the **protected** access specifier.

1. Create a class named **ProtectedClass**.
2. Declare a protected variable called **protectedVariable** of type **String** with an initial value of "Hello."
3. Implement a protected method called **protectedMethod** that prints "Protected method called."
4. Create a subclass named **DerivedClass** that extends **ProtectedClass**. In **DerivedClass**, use the protected variable and method.
5. In the **MainClass**, create an instance of **DerivedClass** and use its members.

**Solution 2:**

// ProtectedClass.java

public class ProtectedClass {

protected String protectedVariable = "Hello";

protected void protectedMethod() {

System.out.println("Protected method called.");

}

}

// DerivedClass.java

public class DerivedClass extends ProtectedClass {

// Additional functionality in the derived class, if needed.

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

DerivedClass derivedObj = new DerivedClass();

System.out.println("Protected Variable: " + derivedObj.protectedVariable);

derivedObj.protectedMethod();

}

}

**Exercise 3:**

**Objective:** Practice using the default (package-private) access specifier.

1. Create a class named **DefaultClass**.
2. Declare a package-private variable called **defaultVariable** of type **double** with an initial value of 3.14.
3. Implement a package-private method called **defaultMethod** that prints "Default method called."
4. In the **MainClass**, create an instance of **DefaultClass** and use the default variable and method.

**Solution 3:**

// DefaultClass.java

class DefaultClass {

double defaultVariable = 3.14;

void defaultMethod() {

System.out.println("Default method called.");

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

DefaultClass defaultObj = new DefaultClass();

System.out.println("Default Variable: " + defaultObj.defaultVariable);

defaultObj.defaultMethod();

}

}

**Exercise 4:**

**Objective:** Practice using the **private** access specifier.

1. Create a class named **PrivateClass**.
2. Declare a private variable called **privateVariable** of type **boolean** with an initial value of **true**.
3. Implement a private method called **privateMethod** that prints "Private method called."
4. In the **MainClass**, try to create an instance of **PrivateClass** and observe the compilation error.

**Solution 4:**

// PrivateClass.java

public class PrivateClass {

private boolean privateVariable = true;

private void privateMethod() {

System.out.println("Private method called.");

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

// Uncommenting the line below will result in a compilation error.

// PrivateClass privateObj = new PrivateClass();

}

}

**Exercise 5:**

**Objective:** Simulate a real-world scenario using access specifiers.

Consider a scenario where you have a **Person** class with private fields for **name** and **age**. Implement getter and setter methods for these private fields. Additionally, create a subclass **Employee** that extends **Person**. The **Employee** class should have an additional private field for **employeeId**. Implement getter and setter methods for the **employeeId** field.

1. Create a class named **Person** with private fields **name** (String) and **age** (int).
2. Implement getter and setter methods for the **name** and **age** fields in the **Person** class.
3. Create a subclass named **Employee** that extends **Person**.
4. Add a private field **employeeId** (String) to the **Employee** class.
5. Implement getter and setter methods for the **employeeId** field in the **Employee** class.
6. In the **MainClass**, create an instance of **Employee**, set values using the setter methods, and retrieve values using the getter methods.

**Solution 5:**

// Person.java

public class Person {

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

// Employee.java

public class Employee extends Person {

private String employeeId;

public String getEmployeeId() {

return employeeId;

}

public void setEmployeeId(String employeeId) {

this.employeeId = employeeId;

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

Employee employee = new Employee();

employee.setName("John Doe");

employee.setAge(30);

employee.setEmployeeId("E12345");

System.out.println("Employee Name: " + employee.getName());

System.out.println("Employee Age: " + employee.getAge());

System.out.println("Employee ID: " + employee.getEmployeeId());

}

}

**Exercise 6:**

**Objective:** Simulate a real-world scenario using the default (package-private) access specifier.

Consider a scenario where you have a package with multiple classes representing a simple library. Implement a **Library** class with a package-private method to check the availability of a book by its title. Create a class named **Book** with private fields for **title** and **author**. The **Library** class should have a array of available books. The package-private method in **Library** should search for a book by title and return true if available; otherwise, return false.

1. Create a package (directory) named **library** for your classes.
2. Inside the **library** package, create a class named **Library** with a package-private method **isBookAvailable(String title)** that takes a book title as a parameter and returns a boolean indicating whether the book is available.
3. Inside the **library** package, create a class named **Book** with private fields **title** (String) and **author** (String).
4. In the **Library** class, maintain a array of available books. Initialize the list in the constructor with a few books.
5. In the **MainClass** (outside the **library** package), create an instance of **Library**, and use the **isBookAvailable()** method to check the availability of a book by title.

**Solution 6:**

// library/Library.java

package library;

class Library {

private Book[] availableBooks;

Library() {

// Initialize the array with some books

availableBooks = new Book[]{

new Book("The Great Gatsby", "F. Scott Fitzgerald"),

new Book("To Kill a Mockingbird", "Harper Lee"),

new Book("1984", "George Orwell")

};

}

boolean isBookAvailable(String title) {

for (Book book : availableBooks) {

if (book.getTitle().equals(title)) {

return true;

}

}

return false;

}

}

// library/Book.java

package library;

class Book {

private String title;

private String author;

Book(String title, String author) {

this.title = title;

this.author = author;

}

String getTitle() {

return title;

}

String getAuthor() {

return author;

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

// Create an instance of Library

library.Library myLibrary = new library.Library();

// Check the availability of a book

String bookTitle = "1984";

boolean isAvailable = myLibrary.isBookAvailable(bookTitle);

if (isAvailable) {

System.out.println("The book '" + bookTitle + "' is available.");

} else {

System.out.println("The book '" + bookTitle + "' is not available.");

}

}

}

**Exercise 8:**

**Objective:** Simulate a real-world scenario using the default (package-private) access specifier with a practical example.

Consider a scenario where you have a package representing a basic customer management system. Implement a **Customer** class with package-private fields for **customerId** (String) and **customerName** (String). Create a **CustomerManager** class with a package-private method to add a new customer and check if a customer exists by their ID. The **CustomerManager** class should use an array or another suitable data structure to store customer information.

1. Create a package (directory) named **customer** for your classes.
2. Inside the **customer** package, create a class named **Customer** with package-private fields **customerId** and **customerName**.
3. Inside the **customer** package, create a class named **CustomerManager** with a package-private method **addCustomer(String customerId, String customerName)** that adds a new customer and a method **isCustomerExists(String customerId)** that checks if a customer exists by their ID.
4. In the **CustomerManager** class, use an array or another suitable data structure to store customer information. Initialize the data structure in the constructor.
5. In the **MainClass** (outside the **customer** package), create an instance of **CustomerManager**, add a few customers using the **addCustomer()** method, and check if a customer exists using the **isCustomerExists()** method.

**Solution 8:**

// customer/Customer.java

package customer;

class Customer {

String customerId;

String customerName;

Customer(String customerId, String customerName) {

this.customerId = customerId;

this.customerName = customerName;

}

}

// customer/CustomerManager.java

package customer;

class CustomerManager {

private Customer[] customers;

CustomerManager() {

// Initialize the array with some customers

customers = new Customer[]{

new Customer("C001", "John Doe"),

new Customer("C002", "Alice Smith"),

new Customer("C003", "Bob Johnson")

};

}

void addCustomer(String customerId, String customerName) {

// TODO: Implement logic to add a new customer to the array or data structure.

}

boolean isCustomerExists(String customerId) {

for (Customer customer : customers) {

if (customer.customerId.equals(customerId)) {

return true;

}

}

return false;

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

// Create an instance of CustomerManager

customer.CustomerManager customerManager = new customer.CustomerManager();

// Add a new customer (uncomment the line below and implement the logic in CustomerManager)

// customerManager.addCustomer("C004", "Eva Johnson");

// Check if a customer exists

String customerIdToCheck = "C002";

boolean customerExists = customerManager.isCustomerExists(customerIdToCheck);

if (customerExists) {

System.out.println("Customer with ID '" + customerIdToCheck + "' exists.");

} else {

System.out.println("Customer with ID '" + customerIdToCheck + "' does not exist.");

}

}

}

**Exercise 9:**

**Objective:** Create a multi-level class hierarchy for an online shopping system using all access specifiers (default, private, protected, and public).

Consider a scenario where you have a package representing an online shopping system. Implement the following classes:

1. **Product (Public):**
   * Create a public class named **Product** with private fields for **productId** (String), **productName** (String), and **price** (double).
   * Implement a parameterized constructor to initialize these fields.
   * Provide public getter methods for each field.
2. **Customer (Public):**
   * Create a public class named **Customer** with private fields for **customerId** (String) and **customerName** (String).
   * Implement a parameterized constructor to initialize these fields.
   * Provide public getter methods for each field.
3. **Order (Default):**
   * Create a class named **Order** in the same package as **Product** and **Customer**.
   * Include private fields for **orderId** (String), **product** (Product), **quantity** (int), and **customer** (Customer).
   * Implement a parameterized constructor to initialize these fields.
   * Provide default (package-private) getter methods for each field.
4. **ShoppingCart (Protected):**
   * Create a class named **ShoppingCart** in the same package as **Product**, **Customer**, and **Order**.
   * Include a protected field for an array of **Order** objects.
   * Implement a protected method **calculateTotalPrice()** that calculates and returns the total price of all orders in the shopping cart.
   * Provide a protected method **printShoppingCart()** that prints details of all orders in the shopping cart.
5. **MainClass (Public):**
   * Create a public class named **MainClass** outside the package.
   * In the **MainClass**, create instances of **Product**, **Customer**, and **ShoppingCart**.
   * Demonstrate the usage of all access specifiers by accessing fields and methods from different classes.

**Solution 9:**

// shopping/Product.java

package shopping;

public class Product {

private String productId;

private String productName;

private double price;

public Product(String productId, String productName, double price) {

this.productId = productId;

this.productName = productName;

this.price = price;

}

public String getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public double getPrice() {

return price;

}

}

// shopping/Customer.java

package shopping;

public class Customer {

private String customerId;

private String customerName;

public Customer(String customerId, String customerName) {

this.customerId = customerId;

this.customerName = customerName;

}

public String getCustomerId() {

return customerId;

}

public String getCustomerName() {

return customerName;

}

}

// shopping/Order.java

package shopping;

class Order {

private String orderId;

private Product product;

private int quantity;

private Customer customer;

Order(String orderId, Product product, int quantity, Customer customer) {

this.orderId = orderId;

this.product = product;

this.quantity = quantity;

this.customer = customer;

}

String getOrderId() {

return orderId;

}

Product getProduct() {

return product;

}

int getQuantity() {

return quantity;

}

Customer getCustomer() {

return customer;

}

}

// shopping/ShoppingCart.java

package shopping;

class ShoppingCart {

protected Order[] orders;

protected double calculateTotalPrice() {

// TODO: Implement logic to calculate total price of all orders in the shopping cart.

return 0.0;

}

protected void printShoppingCart() {

// TODO: Implement logic to print details of all orders in the shopping cart.

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

Product laptop = new Product("P001", "Laptop", 999.99);

Customer johnDoe = new Customer("C001", "John Doe");

ShoppingCart shoppingCart = new ShoppingCart();

// Demonstrate access to public members

System.out.println("Product ID: " + laptop.getProductId());

System.out.println("Customer ID: " + johnDoe.getCustomerId());

// Demonstrate usage of default (package-private) members

Order order1 = new Order("O001", laptop, 2, johnDoe);

System.out.println("Order ID: " + order1.getOrderId());

// Demonstrate access to protected members

double totalPrice = shoppingCart.calculateTotalPrice();

System.out.println("Total Price: " + totalPrice);

// Demonstrate usage of protected members

shoppingCart.printShoppingCart();

}

}

In this exercise, you created a multi-level class hierarchy for an online shopping system using all access specifiers (**public**, **default**, **protected**). The classes **Product** and **Customer** have public access, **Order** has default (package-private) access, and **ShoppingCart** has protected access. The **MainClass** demonstrates the usage of these classes and access specifiers. Note that the **calculateTotalPrice** and **printShoppingCart** methods in the **ShoppingCart** class are marked for further implementation.

**Exercise 10:**

**Objective:** Create a scenario using access specifiers to model a university registration system.

Consider a scenario where you have a package representing a university registration system. Implement the following classes:

1. **Student (Public):**
   * Create a public class named **Student** with private fields for **studentId** (String) and **studentName** (String).
   * Implement a parameterized constructor to initialize these fields.
   * Provide public getter methods for each field.
2. **Course (Default):**
   * Create a class named **Course** in the same package as **Student**.
   * Include private fields for **courseId** (String), **courseName** (String), and **creditHours** (int).
   * Implement a parameterized constructor to initialize these fields.
   * Provide default (package-private) getter methods for each field.
3. **Registration (Protected):**
   * Create a class named **Registration** in the same package as **Student** and **Course**.
   * Include a protected field for an array of **Course** objects.
   * Implement a protected method **addCourse(Course course)** that adds a course to the registration list.
   * Provide a protected method **printRegistration()** that prints details of all courses in the registration.
4. **University (Public):**
   * Create a public class named **University** in the same package as **Student**, **Course**, and **Registration**.
   * Include private fields for an array of **Student** objects and an array of **Registration** objects.
   * Implement a parameterized constructor to initialize these fields.
   * Provide public methods to add a student, add a course to a student's registration, and print the registration details of a student.
5. **MainClass (Public):**
   * Create a public class named **MainClass** outside the package.
   * In the **MainClass**, create instances of **University**, **Student**, **Course**, and **Registration**.
   * Demonstrate the usage of all access specifiers by accessing fields and methods from different classes.

**Solution 10:**

// university/Student.java

package university;

public class Student {

private String studentId;

private String studentName;

public Student(String studentId, String studentName) {

this.studentId = studentId;

this.studentName = studentName;

}

public String getStudentId() {

return studentId;

}

public String getStudentName() {

return studentName;

}

}

// university/Course.java

package university;

class Course {

private String courseId;

private String courseName;

private int creditHours;

Course(String courseId, String courseName, int creditHours) {

this.courseId = courseId;

this.courseName = courseName;

this.creditHours = creditHours;

}

String getCourseId() {

return courseId;

}

String getCourseName() {

return courseName;

}

int getCreditHours() {

return creditHours;

}

}

// university/Registration.java

package university;

import java.util.ArrayList;

import java.util.List;

class Registration {

protected List<Course> courses;

public Registration() {

courses = new ArrayList<>();

}

protected void addCourse(Course course) {

courses.add(course);

System.out.println("Course added to registration: " + course.getCourseName());

}

protected void printRegistration() {

if (courses.isEmpty()) {

System.out.println("No courses registered.");

} else {

System.out.println("Registered Courses:");

for (Course course : courses) {

System.out.println("Course ID: " + course.getCourseId());

System.out.println("Course Name: " + course.getCourseName());

System.out.println("Credit Hours: " + course.getCreditHours());

System.out.println("--------------------------");

}

}

}

}

// university/University.java

package university;

import java.util.ArrayList;

import java.util.List;

public class University {

private List<Student> students;

private List<Registration> registrations;

public University() {

students = new ArrayList<>();

registrations = new ArrayList<>();

}

public void addStudent(Student student) {

students.add(student);

System.out.println("Student added: " + student.getStudentName());

}

public void addCourseToRegistration(Student student, Course course) {

// Find the registration associated with the student

Registration studentRegistration = findStudentRegistration(student);

if (studentRegistration != null) {

studentRegistration.addCourse(course);

} else {

System.out.println("Student not found. Cannot add course.");

}

}

public void printStudentRegistrationDetails(Student student) {

// Find the registration associated with the student

Registration studentRegistration = findStudentRegistration(student);

if (studentRegistration != null) {

System.out.println("Registration details for student: " + student.getStudentName());

studentRegistration.printRegistration();

} else {

System.out.println("Student not found. No registration details available.");

}

}

// Helper method to find the registration associated with a student

private Registration findStudentRegistration(Student student) {

for (Registration registration : registrations) {

if (registration.getStudent().equals(student)) {

return registration;

}

}

return null;

}

}

// MainClass.java

public class MainClass {

public static void main(String[] args) {

Student johnDoe = new university.Student("S001", "John Doe");

Course javaCourse = new university.Course("C001", "Java Programming", 3);

Registration johnsRegistration = new university.Registration();

// Create an instance of University

University myUniversity = new University(new Student[]{johnDoe}, new Registration[]{johnsRegistration});

// Demonstrate access to public members

System.out.println("Student ID: " + johnDoe.getStudentId());

System.out.println("Course ID: " + javaCourse.getCourseId());

// Demonstrate usage of default (package-private) members

javaCourse.getCreditHours();

// Demonstrate access to protected members

johnsRegistration.addCourse(javaCourse);

johnsRegistration.printRegistration();

// Demonstrate usage of public members

myUniversity.addStudent(johnDoe);

myUniversity.addCourseToRegistration(johnDoe, javaCourse);

myUniversity.printStudentRegistrationDetails(johnDoe);

}

}

In this exercise, you created a scenario using access specifiers to model a university registration system. The classes **Student** and **University** have public access, **Course** has default (package-private) access, and **Registration** has protected access. The solution provides a starting point for you to implement the logic to add a student, add a course to a student's registration, and print the registration details of a student in the **University** class, and to add a course and print the registration in the **Registration** class.