**Lab 4: Inheritance - Polymorphism**

**Introduction**

This lab manual is designed to help students grasp essential Java programming concepts through hands-on exercises. Students will explore important topics such as constructors, inheritance, polymorphism, and good programming practices. Each exercise focuses on a specific concept, encouraging students to apply their knowledge practically.

**Objectives**

1. Understand and implement the this keyword for constructor overloading.
2. Practice good programming practices, including constructor chaining and avoiding method calls in constructors.
3. Learn the concept of inheritance and method overriding.
4. Differentiate between a reference variable and an object.
5. Understand compile-time vs. runtime polymorphism through method overloading and overriding.

**Exercises**

**Exercise 1: Constructor Overloading**

* **Task:** Create a class Book with two constructors: one that initializes only the title and another that initializes both the title and the author. Implement a method to display the book details.
* **Output:**

Book Title: The Great Gatsby

Author: Unknown

Book Title: 1984

Author: George Orwell

**Exercise 2: Constructor Chaining**

* **Task:** Implement a class Person with a constructor that initializes the name. Use constructor chaining to create another constructor that initializes both name and age. Display the person’s details.
* **Output:**

Name: Alice

Age: 0

Name: Bob

Age: 25

**Exercise 3: Method Overriding**

* **Task:** Create a base class Shape with a method area(). Derive classes Circle and Rectangle from Shape and override the area() method to return the area of the respective shape. Demonstrate polymorphism using an array of Shape references.
* **Output:**

Circle Area: 78.5

Rectangle Area: 30.0

**Exercise 4: Reference vs Object**

* **Task:** Create a class Car with a model property. Instantiate a Car object and use a reference variable to access and display the car's model.
* **Output:**

Car Model: Tesla Model 3

**Exercise 5: Method Overloading**

* **Task:** Implement a class Calculator with overloaded methods for add(). Create methods to add two integers, three integers, and two doubles. Display the results of each method call.
* **Output:**

Sum of 3 and 5: 8

Sum of 3, 4, and 5: 12

Sum of 2.5 and 3.5: 6.0

**Exercise 6: Inheritance and Super Keyword**

* **Task:** Create a class Employee with properties like name and salary. Derive a class Manager from Employee and add an additional property department. Use the super keyword to initialize inherited properties. Display the details of a manager.
* **Output:**

Name: John Doe

Salary: 60000

Department: Sales

**Exercise 7: Abstract Classes**

* **Task:** Create an abstract class Appliance with an abstract method turnOn(). Derive classes WashingMachine and Refrigerator from Appliance and implement the turnOn() method. Demonstrate the functionality.
* **Output:**

Washing Machine is now ON.

Refrigerator is now ON.

**Exercise 8: Interface Implementation**

* **Task:** Define an interface Playable with a method play(). Create classes Song and Video that implement this interface. Call the play() method for each.
* **Output:**

Playing song: Shape of You

Playing video: Tutorial

**Exercise 9: Exception Handling**

* **Task:** Create a method divide(int a, int b) that throws an ArithmeticException when dividing by zero. Implement a try-catch block to handle this exception and display an appropriate message.
* **Output:**

Cannot divide by zero.

**Exercise 10: Static Members**

* **Task:** Implement a class Counter with a static variable count that increments every time an object is created. Create multiple Counter objects and display the total count.
* **Output:**

Total Count of Objects: 5

**Exercise 11: Exploring Polymorphism with Method Overriding**

**Task:** Create a base class named Shape with a method draw(). Then, create two subclasses, Circle and Square, each overriding the draw() method to display a specific message for the shape. Demonstrate polymorphism by creating a Shape reference that can point to objects of both Circle and Square.

**Expected Output:**

When calling the draw() method on the Shape reference, it should output the specific messages for both shapes.

**Exercise 12: Library Management System**

**Task**: Create a simple Library Management System that allows users to add books, view the list of available books, and check out books. Use classes to represent Books and the Library.

**Requirements:**

1. **Book Class**:
   * **Attributes**:
     + title (String)
     + author (String)
     + isCheckedOut (boolean)
   * **Methods**:
     + Constructor to initialize the attributes.
     + toString() method to return a string representation of the book (title and author).
       - *Use Method Overloading*: Create multiple toString() methods that provide different levels of detail (e.g., one for just the title and another for title and author).
2. **Library Class**:
   * **Attributes**:
     + books (ArrayList of Book)
   * **Methods**:
     + addBook(Book book): Adds a book to the library.
       - *Use Constructor Overloading*: Provide a constructor that allows initializing the Library with a list of books.
     + viewBooks(): Displays all books in the library. Indicate if a book is checked out or available.
       - *Use Polymorphism*: If you decide to create different types of books (e.g., EBook, AudioBook), this method can utilize polymorphism to display their details based on their specific classes.
     + checkOutBook(String title): Allows a user to check out a book by title. If the book is not available, display an appropriate message.
3. **Main Class**:
   * Create an instance of the Library.
   * Add at least 5 books to the library using the addBook method.
   * Display the list of books using the viewBooks method.
   * Implement a simple menu that allows users to check out a book by entering the book title.

**Output:**

The program should interactively allow users to perform the following actions:

* Display the list of books with their availability.
* Check out a book by title, updating its status.
* Display an appropriate message if the book is already checked out.

**Example Output:**

Available Books:

1. The Great Gatsby by F. Scott Fitzgerald (Available)

2. 1984 by George Orwell (Available)

3. To Kill a Mockingbird by Harper Lee (Available)

4. The Catcher in the Rye by J.D. Salinger (Available)

5. Moby Dick by Herman Melville (Available)

Enter the title of the book you want to check out: 1984

You have checked out "1984" by George Orwell.

Available Books:

1. The Great Gatsby by F. Scott Fitzgerald (Available)

2. 1984 by George Orwell (Checked Out)

3. To Kill a Mockingbird by Harper Lee (Available)

4. The Catcher in the Rye by J.D. Salinger (Available)

5. Moby Dick by Herman Melville (Available)

**Suggested Features and Their Usage:**

* **Polymorphism**: If you extend the Book class to create different types of books (like EBook or AudioBook), use polymorphism in the viewBooks method to call overridden methods from these subclasses, allowing for specific implementations of how each type of book is displayed.
* **Method Overloading**: In the Book class, overload the toString() method to provide different string representations, enhancing the readability and usability of your class.
* **Constructor Overloading**: Use constructor overloading in the Library class to allow for different ways of creating a library instance, such as initializing it with an empty list of books or with a predefined list.