Inheritance and Encapsulation Practical Example in Java

Lecture 09

Encapsulation in Java

1. Create an Encapsulated class Book. Its data members are
   - author (String)
   - chapterNames[100] (String[])

2. Create two overloaded constructors, one with no argument and one with two arguments.

3. Create a method compareBooks that compares the author of two Books and returns true if both books have same author and false otherwise. (This method must manipulate two Book objects)

4. Create a method compareChapterNames that compares the chapter names of two Books and returns the book with larger chapters. Display the author of the book with greater chapters in main.

Encapsulation in Java

Create a runner class that declares two objects of type Book.

One object should be declared using no argument constructor and then the parameters should be set through the set() methods.

The second object should be declared with argument constructor. Finally the CompareBooks() and compareChapterNames method should be called and the result should be displayed in the runner class.

Encapsulation in Java

1. **Create an Encapsulated class Book. Its data members are**
   - **author (String)**
   - **chapterNames[100] (String[])**

```
// Encapsulated Book class
public class Book {
    // Private data members (Encapsulation)
    private String author;
    private String[] chapterNames = new String[100];
    private int chapterCount;  // Keeps track of actual number of chapters
```

Encapsulation in Java

Create an Encapsulated class Book. Its data members are

- **author (String)**
- **chapterNames[100] (String[])**

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

   public void setChapterNames(String[] chapterNames) {
      this.chapterNames = chapterNames;
      this.chapterCount = chapterNames.length;
   }

   // Getter methods
   public String getAuthor() {
      return author;
   }

   public int getChapterCount() {
      return chapterCount;
   }
```

Encapsulation in Java

Create two overloaded constructors, one with no argument and one with two arguments.

```
// Default constructor (no arguments)
    public Book() {
        this.author = "";
        this.chapterCount = 0;  // No chapters initially
    }

    // Overloaded constructor with two arguments
    public Book(String author, String[] chapterNames) {
        this.author = author;
        this.chapterNames = chapterNames;
        this.chapterCount = chapterNames.length;
    }
```

Encapsulation in Java

Create a method compareBooks that compares the author of two Books and returns true if both books have same author and false otherwise. (This method must manipulate two Book objects)

```
// Method to compare authors of two books

public boolean compareBooks(Book otherBook) {

    return this.author.equals(otherBook.getAuthor());

}
```

Encapsulation in Java

Create a method compareChapterNames that compares the chapter names of two Books and returns the book with larger chapters. Display the author of the book with greater chapters in main.

```
// Method to compare the number of chapters between two books
  public Book compareChapterNames(Book otherBook) {
      if (this.chapterCount > otherBook.getChapterCount()) {
          return this;  // This book has more chapters
      } else {
          return otherBook;  // Other book has more chapters
      }
  }
}
```

Encapsulation in Java

```
// Runner class to test the Book class
public class Main {

  public static void main(String[] args) {
    // Creating the first Book object using no-argument constructor

Book book1 = new Book();
    book1.setAuthor("Author A");
    String[] chapters1 = {"Introduction", "Chapter 1", "Chapter 2"};
    book1.setChapterNames(chapters1);

    // Creating the second Book object using argument constructor
    String[] chapters2 = {"Intro", "Chapter 1", "Chapter 2", "Chapter 3"};
     Book book2 = new Book("Author B", chapters2);
```

Encapsulation in Java

```
// Compare the authors of both books
boolean sameAuthor = book1.compareBooks(book2);
if (sameAuthor) {
    System.out.println("Both books have the same author.");
} else {
    System.out.println("The books have different authors.");
}

// Compare the chapter counts of both books and display the author of the book with more chapters
Book bookWithMoreChapters = book1.compareChapterNames(book2);
        System.out.println("The book with more chapters is written by: " +
bookWithMoreChapters.getAuthor());
    }
}
```

Inheritance in Java

Imagine a publishing company that markets both book and audio-cassette versions of its works. Create a **class publication** that stores the **title** and **price** of a publication.

```
import java.util.Scanner;
// Base class Publication
class Publication {
    private String title;
    private double price;

    // Setters for title and price
    public void setTitle(String title) {
        this.title = title;
    }

    public void setPrice(double price) {
        this.price = price;
    }
```

Inheritance in Java

Imagine a publishing company that markets both book and audio-cassette versions of its works. Create a **class publication** that stores the **title** and **price** of a publication.

```
    // Getters for title and price
    public String getTitle() {
        return title;
    }

    public double getPrice() {
        return price;
    }
    // Display method to show the details of the publication
    public void display() {
        System.out.println("Title: " + title);
        System.out.println("Price: $" + price);
    }
}
```

Inheritance in Java

book, which adds a <u>page count</u>

```
// Derived class Book from Publication
class Book extends Publication {
    private int pageCount;

    // Setter for pageCount
    public void setPageCount(int pageCount) {
        this.pageCount = pageCount;
    }
    // Getter for pageCount
    public int getPageCount() {
        return pageCount;
    }
```

Inheritance in Java

```
// Override display method to include page count
    @Override
    public void display() {
      super.display();  // Call the display method of Publication
      System.out.println("Page Count: " + pageCount);
    }
}
```

Inheritance in Java

- tape, which adds a <u>playing time</u> in minutes.

```
// Derived class Tape from Publication
class Tape extends Publication {
    private double playingTime;

    // Setter for playingTime
    public void setPlayingTime(double playingTime) {
        this.playingTime = playingTime;
    }

    // Getter for playingTime
    public double getPlayingTime() {
        return playingTime;
    }
```

Inheritance in Java

```
// Override display method to include playing time
    @Override
    public void display() {

        super.display();  // Call the display method of Publication

        System.out.println("Playing Time: " + playingTime + " minutes");
    }
}
```

Inheritance in Java

Write a main() program to test the book and tape class by creating instances of them, asking the
user to **<u>fill in their data</u>** and then **displaying** the data with display().

Inheritance in Java

```
// Main class to test Book and Tape
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Creating an object of Book
        Book book = new Book();
        System.out.println("Enter details for the book:");
        System.out.print("Enter title: ");
        book.setTitle(sc.nextLine());
        System.out.print("Enter price: ");
        book.setPrice(sc.nextDouble());
        System.out.print("Enter page count: ");
        book.setPageCount(sc.nextInt());

        sc.nextLine();  // Clear the buffer
```

Inheritance in Java

```
// Creating an object of Tape
        Tape tape = new Tape();
        System.out.println("\nEnter details for the tape:");
        System.out.print("Enter title: ");
        tape.setTitle(sc.nextLine());
        System.out.print("Enter price: ");
        tape.setPrice(sc.nextDouble());
        System.out.print("Enter playing time in minutes: ");
        tape.setPlayingTime(sc.nextDouble());

        // Displaying the details of the book and tape
        System.out.println("\nBook Details:");
        book.display();

        System.out.println("\nTape Details:");
        tape.display();

        sc.close();
    }
```