Programing oth

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:93531716

Submission Date

Apr 29, 2025, 9:18 PM GMT+5:45

Download Date

Apr 29, 2025, 9:22 PM GMT+5:45

File Name

Programing oth

File Size

17.7 KB

16 Pages

2,788 Words

15,238 Characters

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **31** Not Cited or Quoted 10%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
5%	Publications
9%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **31** Not Cited or Quoted 10%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
5%	Publications
9%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	www.geeksforgeeks.org	1%
2	Publication	Usharani Bhimavarapu. "Java 22 for Healthcare and Medical Applications", CRC Pr...	1%
3	Submitted works	GLA University on 2025-03-27	1%
4	Submitted works	Bedford College Group on 2025-03-14	<1%
5	Submitted works	University of Ghana on 2024-06-08	<1%
6	Internet	examradar.com	<1%
7	Submitted works	HRUC on 2025-04-23	<1%
8	Internet	www.tutorialrepublic.com	<1%
9	Submitted works	Kingston University on 2015-03-26	<1%
10	Submitted works	University of Central Lancashire on 2025-03-05	<1%

11 Submitted works	
University of Wales, Bangor on 2012-12-23	<1%
12 Submitted works	
West Herts College on 2024-11-11	<1%
13 Submitted works	
London School of Commerce on 2010-10-25	<1%
14 Submitted works	
The Manchester College on 2025-04-24	<1%
15 Submitted works	
University of Wales Institute, Cardiff on 2024-08-23	<1%
16 Submitted works	
Coventry University on 2012-03-26	<1%
17 Submitted works	
Kensington College of Business - Brunei on 2023-10-08	<1%
18 Submitted works	
Trident University International on 2024-02-20	<1%
19 Submitted works	
London School of Commerce on 2010-06-21	<1%

1.a. What are the four main principles of Object-Oriented Programming (OOP)? Provide a brief explanation of each.

Ans. The four main principles of object-oriented programming are as follows:

Encapsulation: The process of hiding the implementation and sharing only the necessary through the public interface is called Encapsulation. Encapsulation is also known as the bundle of attribute or properties and methods into a single unit called a class. The class contains everything needed for a part of creation. For example, if you own the car, you know how to run the car, but you don’t know how it works, that’s encapsulation where the implementation is hidden.

Figure 1: Keeping class data private and using getter/setter methods to control access

Inheritance: The process of inheriting behaviors and properties from another class (sub class or base class or supper class) is called Inheritance. It has benefits like Code Reusability, Hierarchy Establishment, Maintainability where avoids writing the same code multiple times, Allows the creation of aa hierarchical structure of classes, flexing the relationship between different classes and objects. For example, you have created a bus and car to set the similar item like wheels, handles etc. are inherited at the time the process of inheriting the behaviors and properties from another class is known as inheritance.

Figure 2:Parent class teacher

6 Figure 3: MathTeacher class to reuse and extend the functionality of the Teacher class.

1 Polymorphism: Polymorphism allows objects to behave differently based on specific
7 class type. Polymorphism has features like multiple behaviors like the same method can behave differently depending on the object call this method, In Method Overriding, Method overloading, A child class can define or redefine a method of its parent class
1 or sub class and we can define multiple methods in the same class with different parameters respectively.

Figure 4:Different notification types to respond differently to the same method call

Abstraction: The process of simplifying the complex reality and hiding the complex implementation is known as Abstraction. It mainly focuses on “what” an object does rather than “how”. For example, we use keyboards, mouse of the laptop but how it runs inside we don’t know what is happening inside Is known as the abstraction of data.

Figure 5:Hiding implementation details while allowing interaction through a general interface.

1.b. How does the concept of a class differ from that of an object in Java? Provide examples from the course to illustrate your answer.

17 Ans. In Java, the concept of class differs from that of an object form the foundation of object-oriented design. The class is also known as the blueprint of the class. It has
3 both properties (fields or attributes) and behavior(methods) that the object created from it will have. For instance, a class called a Bike that includes attributes like Brand, Color, Price and speed and behavior(methods) such as startEngine (), applyBrakes(), applyHorn ().

14 An object is an actual instance of class. It's also known as real world entity
16 representation created by the class . For example, when a class creates something specific like a Red Bullet that moves at the speed of 70 km/h, that the specific red bullet is the object of the class. In a course module, where a Car is demonstrated as class, and it has attributes of model, brand and methods or behavior such as start () and accelerate(). The car is an object and the start () , accelerate () is a method.

Figure 6: A class or Blueprint

Figure 7: An object of a class

2.a. Describe how you implemented inheritance in one of the examples provided in the course. How does inheritance promote code reuse?

Ans. In the course, object-oriented programming, inheritance is implemented by using the real word models, such as "person" and "student" class. I implemented inheritance by creating the class Person and with common attributes like name, age and along with the displayInfo () method or behavior. Extends keyword in java help to inherit the properties. After creating Person class I have created Student class which extends

9 Person. The student class inherits all the properties and behaviors of Person class. And added some attributes to Student class like studentID and courseName. Implementation allowed the student class to reuse existing code form the Person class. After watching the linked learning helped me to understand how to create relationship between classes like parent and child class make easier to reuse code and decrease code redundancy.

Figure 8: Parent class Person

Figure 9: Base Class Student

Figure 10:Demonstration of Inheritance in Java using Course example

2 2.b. How did the course demonstrate the use of interfaces in Java? Provide an example of an interface you created during the course and explain its purpose.

2 Ans: The course demonstrates the use of interfaces in java are a way to define a set of method that a class must implement, helping with the same structure for different parts of program.

One of the examples of an interface created during the course was “Event” was the interface and the “passwordChangeEvent” was the class and we use implements keywords for providing the functionality defined in the main interface class. missedPayement and accountTransfer class was also created. Event has the method named: timestamp () and process (). The main purpose of the interface was to ensure consistency across all types of events, keep the code modular and easy to extend and understand. Enabling the polymorphism to keep that all the events can be handled by using the reference of same interface.

Figure 11:Event Interface

Figure 12: Password Change Event that implements Event

Figure 13:AccountTransferEvent that implements Event

Figure 14:Main class demonstrates creating objects of different event types

Figure 15:Output Of Interface implements

7 3.a. Explain how encapsulation is achieved in Java. Why is encapsulation important in software development?

12 Ans: In LinkedIn learning OOP course, Encapsulation is a fundamental principle of oop (object-oriented programming) that was clearly shown throughout the course.

10 Encapsulation is the method to bundle the data and methods into a single unit. class and restricting access to the inner working of that class. Making variable private and only allowing access through getter setter methods.

For example, on one of the courses, we have created a bankAccount class where the balance variable is set as private. This means balance couldn't be access by or directly accessed or modifieds outside of the class . If we want any changes outside of the class then we have to give getter or setter methods .

Encapsulation I not just a technique to hide the code, it's an essential part of the code to develop secure, maintainable and neat and clean software. It ensures object to reduces bug, hiding the code complexity and making it easier to make change without affecting any parts of the system. In software development encapsulation keeps everything well organized and managed. Allowing each class to manage its own data and behavior which leads better security, reduced risk of error ,easier maintenance and improved reusability . In summary, the course helps us to encapsulation promote clean and reliable code in the real world.

Figure 16:Demo Of Encapsulation

2 3.b. In the course, you worked with access modifiers like `private`, `protected`, and `public’. Describe a scenario where each would be appropriately used.

2 Ans: During Linked learning course, we have used or worked with access modifiers like Private, Protected and public which help us to manage access to class members(variables), promoting safe, clean and understandable code. Reducing complexity. One of the exercises we have used modifiers especially when working with interfaces and their implementations. The scenario where each modifier was demonstrated as follows:

1.private: The process of protecting internal data or used to hide internal details of class and couldn’t be accessed directly outside of the class is private access modifier. It provides unauthorized access and changes to variables from outside. For example, in the passwordChangeEvent class implement the Event interface, where we have used access modifier as private to secure internal data like event id and createdTimestamp.

Figure 17:Example of private access modifier

2.protected: The process of modifier allows access to members with the same package or from subclasses is protected access modifier. It is also known as inheritable but controlled. It’s helpful for designing the class hierarchies, like base event class that share properties to other class with specific events. For example, Hypothetically If we have abstract class BaseEvent then multiple events inherited from like MissedPayementEvent, AccountTransferEvent, a filed like eventType be protected so subclasses can use it. This makes a code that can be reuse and customized event related behavior in child classes.

Figure 18: Example of protected access modifier

3.public: The modifier used to make method, fields or variables must be accessed from anywhere such as without defining extra getter setter methods or methods that define core behavior or are part of an interface. It exposes only necessary behavior to the outside world. For example, the Event interface defines two methods as a public and uses @override to define its behavior for these methods . These methods are public, so they can be accessed across all the event types.

Figure 19: Example of public access modifier

4.a. How did the course explain the concept of abstraction? Give an example of how you applied abstraction in one of your projects.

Ans: In Likened Learning, abstraction is introduced as the pillars of OOP (object-

1 oriented programming language. The process of hiding the complex details of methods and showing only the required features to client interface is called Abstraction. This allows the programmer to understand what an object or system can do to their task. It is also known as hiding the implementation.

Abstraction is mostly used in interfaces and abstract classes. A class is defined as

1 blueprint for other classes including methods where abstract classes have abstract methods without implementation that must be implemented by subclasses.

Figure 20:Example of abstract class

The main reason why I applied abstraction to my project is GymMember Class. This class acts as a blueprint of many classes such as RegularMember and PremiumMember, the abstract class allows me to add the same functionality to Both classes. The benefits of abstraction are simplified interaction, flexibility for further changes, and Code reusability.

4.b. Provide a real-world example where polymorphism would be useful in a Java application. How was polymorphism implemented in one of the course exercises?

4 Ans: As we all know, Polymorphism is also pillars of OOP (object-oriented programming. It allows objects of different types to be treated the same as objects of common super class. It also refers to the ability of an entity to make multiple methods. It enhances the reusability of code and flexibility too.

In a real-world java application, polymorphism is mostly useful especially in scenarios like managing various types of names, address, contacts. For example, suppose we must store the details of the company such as name, contact, email then polymorphism would allow us to these objects in a very proper way even after they have different types if objects are represented differently. Overall, polymorphism is the concept to simplifies working with different objects in Java.

15 Figure 21:Example of polymorphism 1

Figure 22:Example of polymorphism 2

In the course, polymorphism was implemented in the Contact and PhoneNumber class. Contact is demonstrated as constructor overloading (method overloading) as polymorphism. The class can handle multiple combinations of attributes. Method Overriding is also used in the exercise. The method overriding toString () in both Contact and PhoneNumber. It handles multiple phone number format using the overloading method in the phoneNumber class.

5.A. Discuss how exception handling is implemented in Java. What are the benefits of using exception handling in OOP?

Ans: In java, the process of getting the runtime error or the process of detection of error and ensuring the program distributed is known as exceptional handling. It provides well organized exceptional handling though five keywords: throw, throws, try, catch, and finally. Well mostly try to catch keywords is used for exceptional handling.

3 1.try block: The code inside the try block check if the error is enclosed or not.

8 2.catch block: If an error occurs in try block the catch block catch the error and display the appropriate error message.

3 3.finally block: The finally block is used to release the files, database connections and other network sockets. The finally block automatically executed whether try catch or 5 exception is thrown or not.

4.throw statement: Explicitly throw the exception in the code.

5 5.throws keyword: The method signatures declare that a method might throw one or more exceptions.

The benefits of the exception handling in Object Oriented Programming (OOP) are as follows:

1.Enhanced program robustness.

2. separation of error handling code for main logic

3.Clear error information

4.object-oriented design

5.Program Maintainability

6.Assits Debugging and Problem Resolution

Figure 23:Exception Example

5.B. Provide an example of a custom-except class you might create based on a scenario from the course. How would you implement this exception in your code?

Ans: In the real-world environment applications, error may occur in the program or not. To handle the different kinds of error more effectively we create custom exception classes. It is easier to understand, more organized, and gives helpful output to the user for further processing.

For example, based on the previous course work, the Gym Management system project we have worked on with the exceptional handling. There is a situation to prevent duplicate member id, In the Gui the user should input the unique values to procced or he/she accidentally input the same value then the system will show the exception.

To show the exception when user gives invalid input, we have created a class called duplicateMemberIDException. This exception is only for handling the duplicate id. I have used built-in exception function for the custom exception.

Figure 24:Example of custom exception

Figure 25: Class to check exception

Figure 26:Main method to run the exception

6. How would you approach designing a Java application for a library management system using OOP principles? Outline the classes, objects, and interactions you would consider. (Identify key components, explain interactions and relationships among classes, consider additional features, provide sample code snippets).

Ans: Java is a classic and real-world application of object-oriented programming languages. Designing a java application for a library management system using OOP principles like Encapsulation, Abstraction, Polymorphism, and inheritance. The main aim is to build software to manage books, manage members, and borrow transactions.

1. Classes and Objects

libraryBook class:

It represents books in the library.

Key attributes: bookcode, bookTitle, bookAuthor, isIssued.

Purpose: Track the books information and availability.

libraryUser Class:

show general user

Key attributes: userId, userName

Purpose: User can borrow and return books

libraryAdmin Class (inherits from libraryUser):

Add books to the library

Inherits normal features but have extra permissions.

LibrarySystem Class:

Admin of the Library

Key Attributes: List of books and users

Purpose: Add users, add books, manage book borrow and returning.

Interactions and Relationships

A user can borrow and return books from the library

The library class manages books and user records

Only the librarian manages books inventory.

Manages logs of transactions.

Figure 27:libraryBook Class

Figure 28: libraryUser Class

Figure 29:librarySystem Class

7. Upon completing the LinkedIn Learning course on Java Object-Oriented Programming, you should have earned a course completion badge. Attach a screenshot or digital copy of your course completion badge that displays your name and the date of completion. In addition, briefly describe a key concept or project from

the course that you found most insightful. How did this course help you enhance your understanding of Object-Oriented Programming? Provide specific examples or code snippets to illustrate your point.

Figure 30: Course Completion Certificate

Ans:

Well, I have completed my java object-oriented course from LinkedIn learning, earned a course completion badge that shows my achievement above.

One of the most insightful concepts I get from the course is about the use of polymorphism and inheritance to minimize the code or simply code reuse and extend functionality. During linked, clear example of parent and child classes can work together to create a maintainable system.

Key concept learned: Inheritance and Method Overriding

Figure 31: Contact class

Figure 32: Phone number class

Figure 33: Main class to demonstrate the inheritance and polymorphism

In the above example, I have understood how the phoneNumber class could override the toString () method of the contact class to define its own behavior. This helped me to understand polymorphism, where the same method works depending on object type.

The course improved my ability to structure code logically using OOP pillars or principles, to apply real-world scenario, understand the importance of abstraction and encapsulation.

Although, the OOP learning course have enhance and built my confidence as well as theoretical understanding and practical understanding on application of OOP in java. I feel more confident to create more maintainable java applications.