**Assignment Project: Zoo Management System**

**Deliverables:**

1. Project Report (Template will be provided, you can add additional sections)
2. Code files (.cpp/.h only)
3. Output screenshots
4. Recorded Presentation (YouTube link)

*Please note that there is contributions table and academic integrity statement included in the Project Report that you are required to fill in.*

**Instructions for Students:**

Project Report:

Write a project report describing the design and implementation details of the Zoo Management System. Include an introduction, problem statement, project objectives, and a brief overview of the system's functionality. Describe the classes, data members, member functions, and their relationships in detail. Explain how inheritance, polymorphism, encapsulation, virtual functions, and exception handling are used in the project. Discuss any challenges faced during the implementation and how they were overcome. Include sample screenshots or code snippets to illustrate key features of the system. Conclude the report with a summary and reflection on the learning outcomes achieved through the project.

Data Validations:

Implement data validations to ensure the correctness and integrity of user input. Validate input related to animal attributes such as age, weight, and other relevant characteristics. Handle invalid inputs gracefully, displaying appropriate error messages to the user. Implement range checks, data type checks, or any other necessary validations based on the requirements.

Comments and Documentation:

Add meaningful comments throughout the code to explain the purpose and functionality of different sections, classes, and functions. Document the purpose, input parameters, return values, and any exceptions thrown for each member function. Use descriptive variable and function names to enhance code readability. Provide a header comment at the beginning of each source code file, including your name, date, project description, and any other relevant information.

Exception Handling:

Implement appropriate exception classes to handle exceptional scenarios in the Zoo Management System. Catch exceptions using try-catch blocks and handle them gracefully. Include meaningful error messages to provide feedback to the user in case of exceptions. Handle exceptions related to invalid input, out-of-range values, or any other exceptional situations.

User Interface:

Design a user-friendly and intuitive menu-driven interface for zookeepers to interact with the system. Clearly display available options and provide prompts for user input. Ensure the interface is responsive and provides appropriate feedback after each user action. Format the output in a readable manner, using proper spacing and indentation.

Testing and Validation:

Test the Zoo Management System thoroughly to ensure its correctness and robustness. Test different functionalities of the system, including adding animals, removing animals, feeding animals, etc. Validate that the system handles exceptional scenarios and invalid inputs correctly. Perform unit testing on individual classes and functions to verify their functionality. Document the testing process and results in the project report.

Code Organization and Modularity:

Organize the code into appropriate files and folders, following good programming practices. Utilize header files (.h) and source files (.cpp) to separate class declarations and implementations. Ensure that each class has its own header and source file. Implement modular functions to handle specific tasks and promote code reusability.

**Project Description: Zoo Management System**

Description:

You have been assigned to develop a Zoo Management System using object-oriented programming in C++. The system should allow zookeepers to manage various aspects of the zoo, including adding and removing animals, feeding and caring for animals, and keeping track of their health and behavior. The project will primarily focus on topics such as inheritance, polymorphism, encapsulation, virtual functions, and exception handling.

Requirements:

1. Inheritance and Polymorphism:

- Create a base class named "Animal" to represent a generic animal in the zoo.

- Derive specific animal classes (e.g., "Lion," "Elephant," "Giraffe") from the base class to represent different types of animals. Create at least 10 specific animal classes, including 4 levels of inheritance hierarchy (e.g., Animal🡪 Bird 🡪 Eagle 🡪 BaldEagle). Please note that it is not required to have 04 levels of inheritance hierarchy for each type of animal. However, you should have at least one such hierarchy.

- Implement appropriate data members and member functions in the derived classes to represent unique characteristics of each animal.

- Utilize inheritance to inherit common attributes and behaviors from the base class.

- Implement virtual functions in the base class and override them in the derived classes to achieve polymorphic behavior.

Use your rationals to add member functions that make sense to you.

2. Encapsulation:

- Encapsulate the data members of the animal classes by making them private or protected.

- Provide public member functions to access and modify the animal's attributes, such as age, weight, and name.

- Implement getter and setter methods to enforce data encapsulation and maintain data integrity.

3. Virtual Functions:

- Define virtual functions in the base class, such as "eat" and "makeSound," to represent common behaviors of animals.

- Override these virtual functions in the derived classes to provide specific implementations based on the type of animal.

- Demonstrate the dynamic binding of virtual functions at runtime based on the actual type of the object.

4. Exception Handling:

- Implement exception handling mechanisms to handle various exceptional scenarios, such as invalid input or inappropriate actions.

- Handle exceptions related to animal feeding, health monitoring, or any other relevant operations.

- Utilize try-catch blocks to catch and handle exceptions gracefully, providing meaningful error messages to the user.

5. Zoo Management:

- Implement a class named "Zoo" to manage the collection of animals in the zoo.

- Include appropriate data members and member functions to add and remove animals from the zoo, monitor their health, and record their behaviors.

- Utilize container classes, such as vectors or arrays, to store and manage the collection of animals.

- Implement functions to search for animals based on various criteria, such as species or age.

6. User Interaction:

- Implement a user-friendly menu-driven interface to allow zookeepers to interact with the system.

- Provide options for adding new animals, removing animals, feeding animals, monitoring their health, and performing other relevant operations.

- Display appropriate messages and error handling for invalid input or exceptional scenarios.

7. Additional Features:

- Implement additional features as per your creativity and understanding, such as animal enrichment activities, animal grouping, or generating daily reports on animal health and behavior.

Note: The above requirements serve as a starting point for developing the Zoo Management System. You are encouraged to enhance the project with your own ideas and additional functionalities that showcase your understanding of the concepts. Remember to plan and organize the project tasks systematically to ensure an efficient implementation.

Submission:

Submit your project code, project report, and any relevant documentation as per the submission guidelines provided in the classroom. Make sure to include all necessary files and ensure they are well-organized. Clearly mention any additional features or functionalities you have implemented beyond the given requirements.

Note: Pay attention to coding standards, readability, and code structure. Follow best practices in programming and adhere to any specific guidelines provided to you in the class.