**INFO 6350 Fall 2024**

**Assignment 4**

Use Swift playground and/or the command line for macOS (open XCode, create a new XCode project, macOS, command line tool), and practice the following exercises:

**Swift Structures:**

1. Define a Swift structure named **Animal** with four properties: **name** (String), **type** (String), **age** (Int), and **location** (String). Example: (“Leo”, “Lion”, 5, “Lion Den”)
2. Add a method called **showDetails()** to the Animal structure that returns a description of the animal, including its type and location.
3. Create three instances of the **Animal** structure with different values for **name**, **type**, **age**, and **location**. Provide details for each instance.
4. Choose one instance of the **Animal** structure and call the **showDetails()** method. Display the output in the console.

**Swift Class:**

1. Define a class **Cage** with two methods: **spaceLeft()** and **animalsInside().**
2. Create a subclass **LionCage** inheriting from **Cage**, with properties **totalSpace** and **lionsCount**. Override the methods to calculate the space left and the number of lions inside.
3. Define another subclass **BirdCage** inheriting from **Cage**, with properties **birdCount** and **maxBirds**. Override the methods to calculate space left and the number of birds inside.
4. Create an instance of **LionCage** with **totalSpace** and **lionsCount** and an instance of **BirdCage** with **birdCount** and **maxBirds**.
5. Print the space left and the number of animals in the **LionCage** using a method called **showCageInfo**().
6. Similarly, print the space left and the number of birds in the **BirdCage** using **showCageInfo**().

**Swift Protocols:**

1. Define a protocol **Move** with a method **moveToCage()**. Create a **ZooTruck** and **ZooTrain** struct that conform to it and implement the **moveToCage()** method.
2. Create a protocol **ZooArea** that has:
   1. A method to return the name of the area (e.g., "Lion Den", "Parrot House").
   2. A method that calculates and returns the total space of the area.
   3. Define two classes that conform to this protocol, such as LionDen and ParrotHouse.

**Swift Extensions:**

1. Extend the Int type to include a method named **ticketPrice()** that calculates and returns the ticket price in dollars by multiplying the integer by 10. Use this method to print the ticket prices for Adult, Children, and Senior Citizen categories.
2. Extend String to add a method **animalType()** that converts the string into an animal's type (e.g., "elephant", "giraffe"). Print the corresponding animal types.
3. Extend Date to add a method **visitDate()** that returns a formatted string like "Visit on: MMM dd, yyyy". Print formatted visit dates.

**Swift Error Handling and Optionals in Ticket Price Calculation:**

1. Create an enumeration **AgeError** that defines two possible errors: **invalidInput** and **negativeAge**.
2. Write a method **calculateTicketPrice(for:)** that accepts an optional integer age. Use optional binding (guard let or if let) to unwrap the age.
3. If the age is invalid or negative, throw the appropriate error from the **AgeError** enumeration.
4. Use a **switch** statement to return the appropriate ticket price based on the customer's age:
   1. Ages 0-12: $10 (Children)
   2. Ages 13-59: $20 (Adults)
   3. Ages 60+: $15 (Senior Citizens)
5. Implement a **do-catch** block to call the **calculateTicketPrice()** method, handle errors, and print the appropriate ticket price. Handle these cases:
   1. **invalidInput**: Print "Please enter a valid age."
   2. **negativeAge**: Print "Age cannot be negative."
6. Create test cases to simulate valid and invalid inputs (e.g., "25", "-5", "abc").