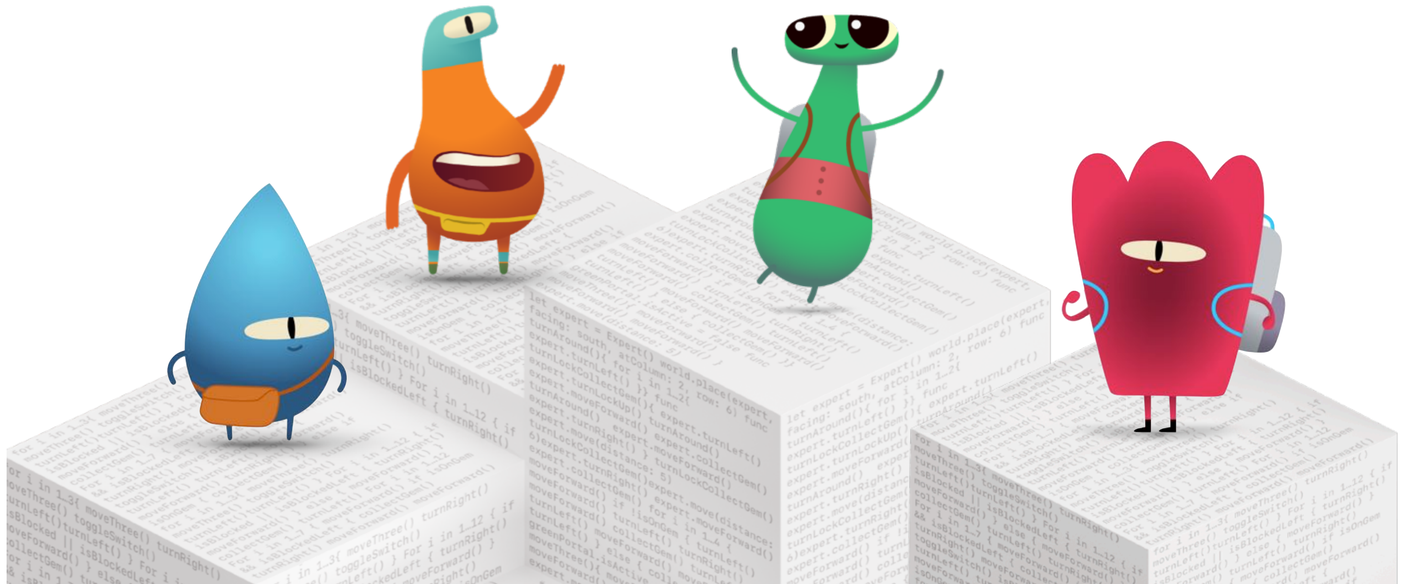
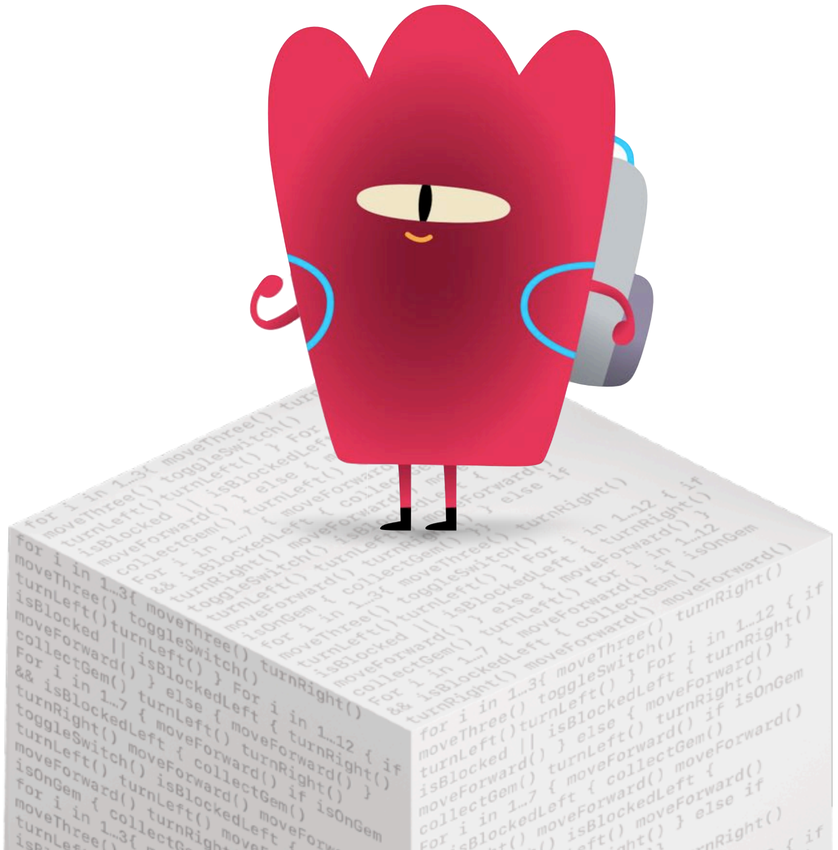
Swift Coding Club

Special Session



Session 3

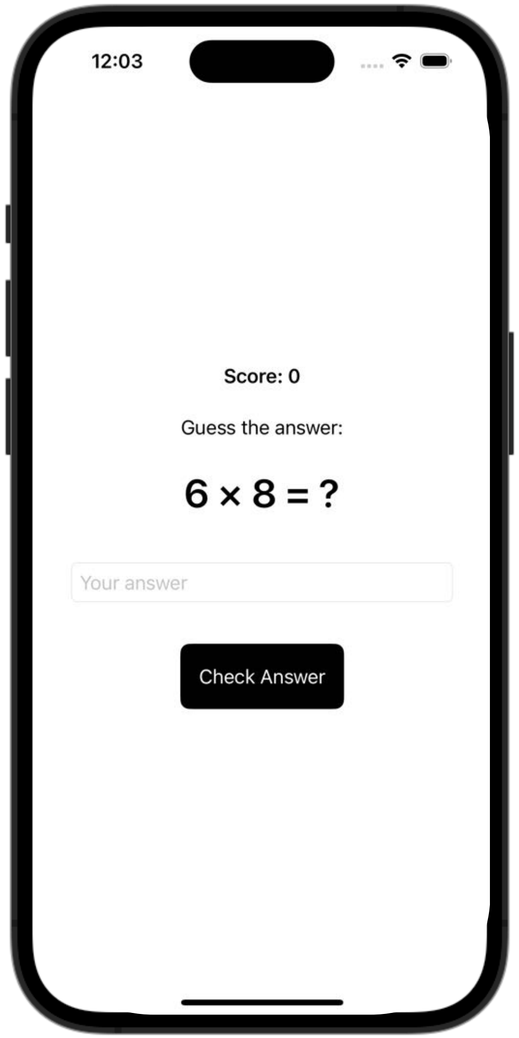
Let’s make some maths game!



Swift Coding Club *Thailand*

Math Game - ***ADVANCE***

Math x game

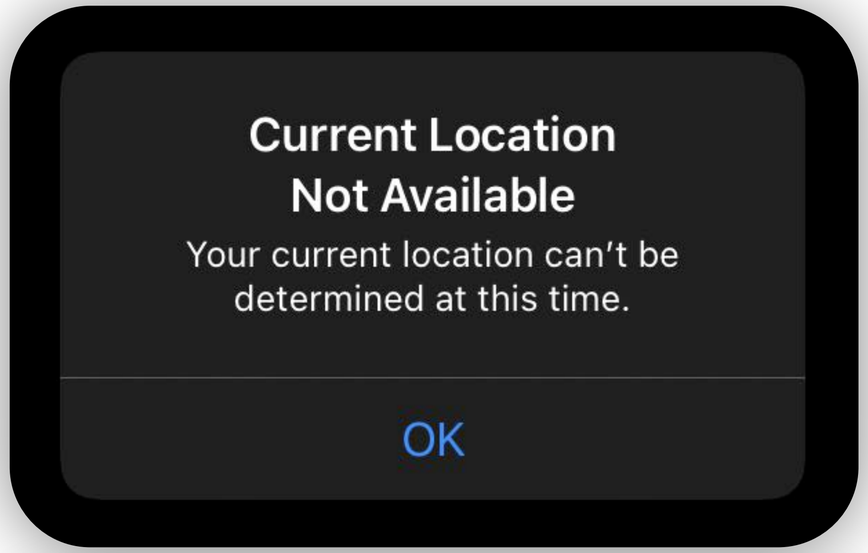


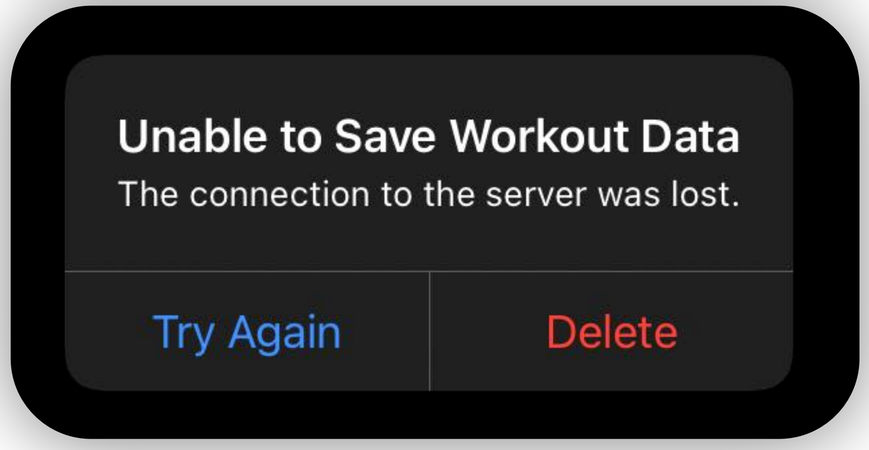
Alert

Alert

Use an alert when you want the user to act in response to the state of the app or system.

Alert


Current Location
Not Available
Your current location can't be
determined at this time.
OK

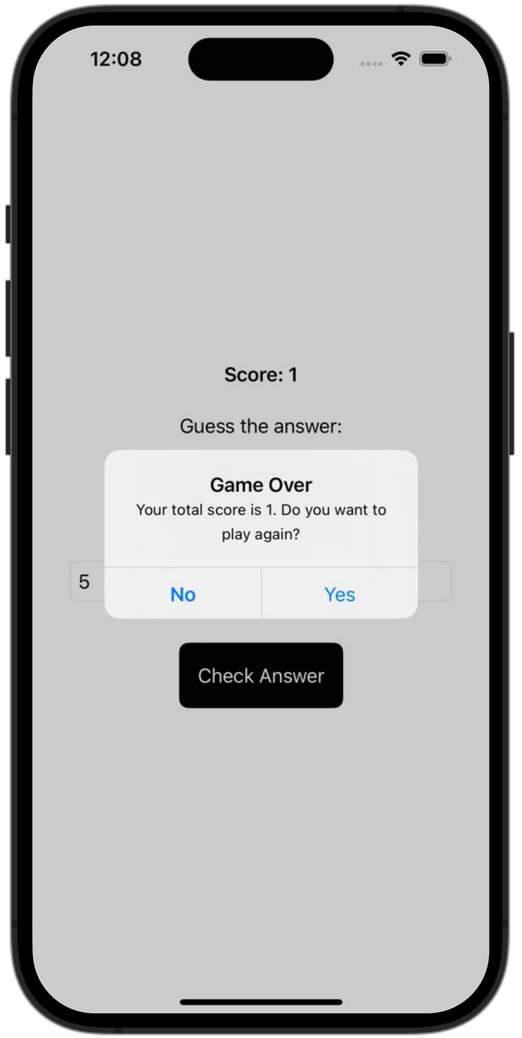

Unable to Save Workout Data
The connection to the server was lost.
Try Again
Delete

Swift Coding Club *Thailand*

Math Game - ***ADVANCE***

Alert

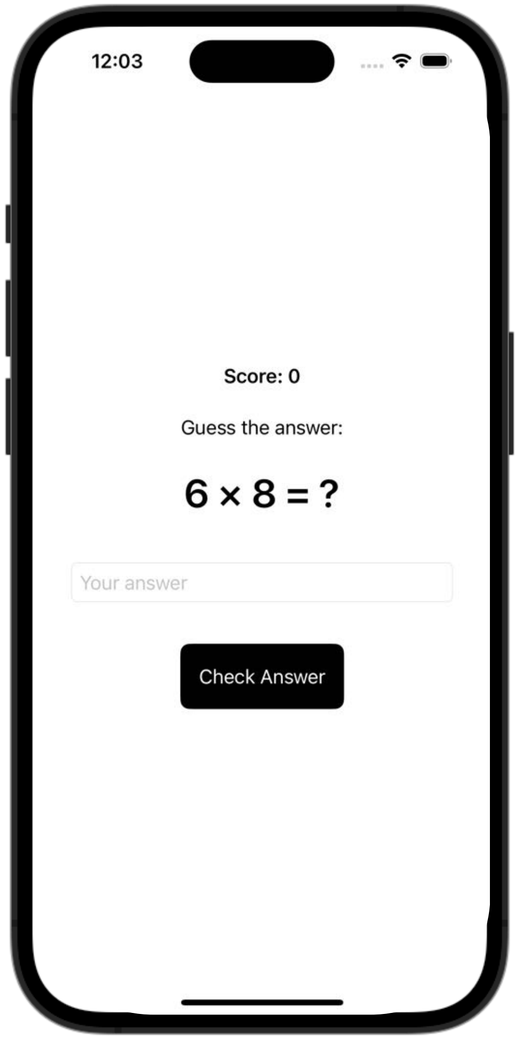
Math x game



Swift Coding Club *Thailand*

Math Game - ***ADVANCE***

Math x game



Swift Coding Club *Thailand*

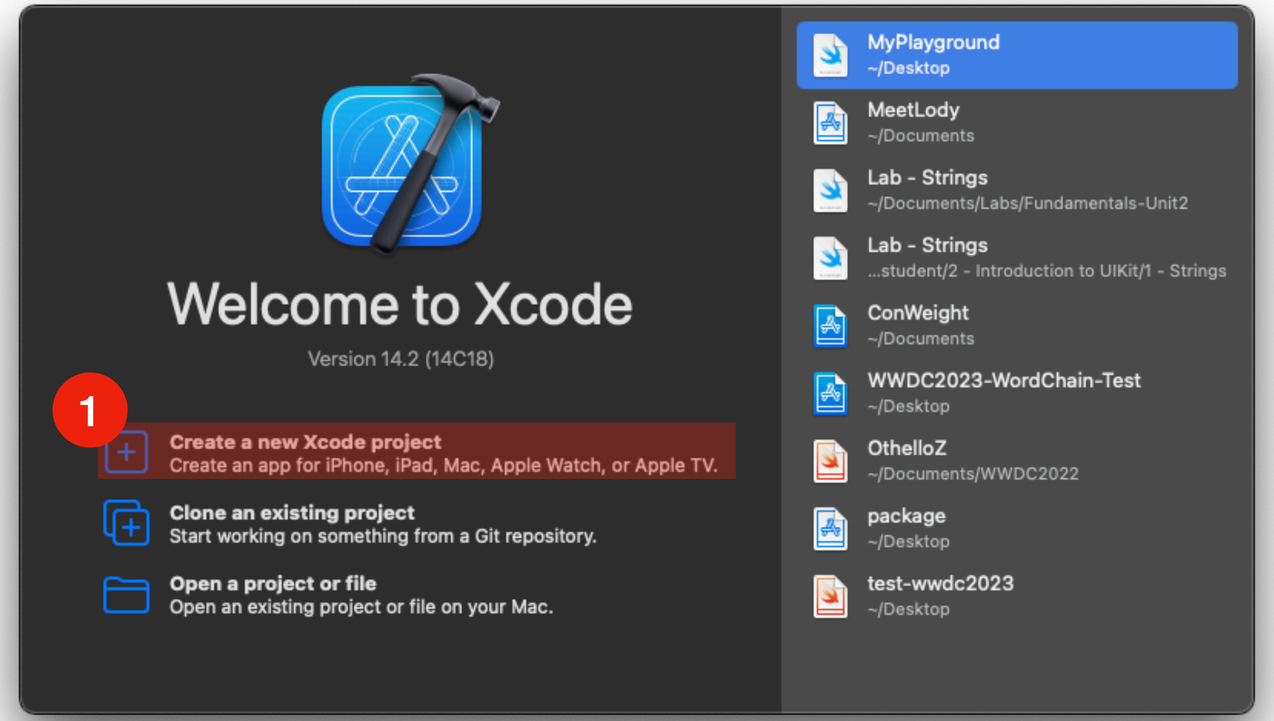
Math Game - ***ADVANCE***

LAB TIME! -

Get Started! 🔥

Math Game - ***ADVANCE***

Open Xcode

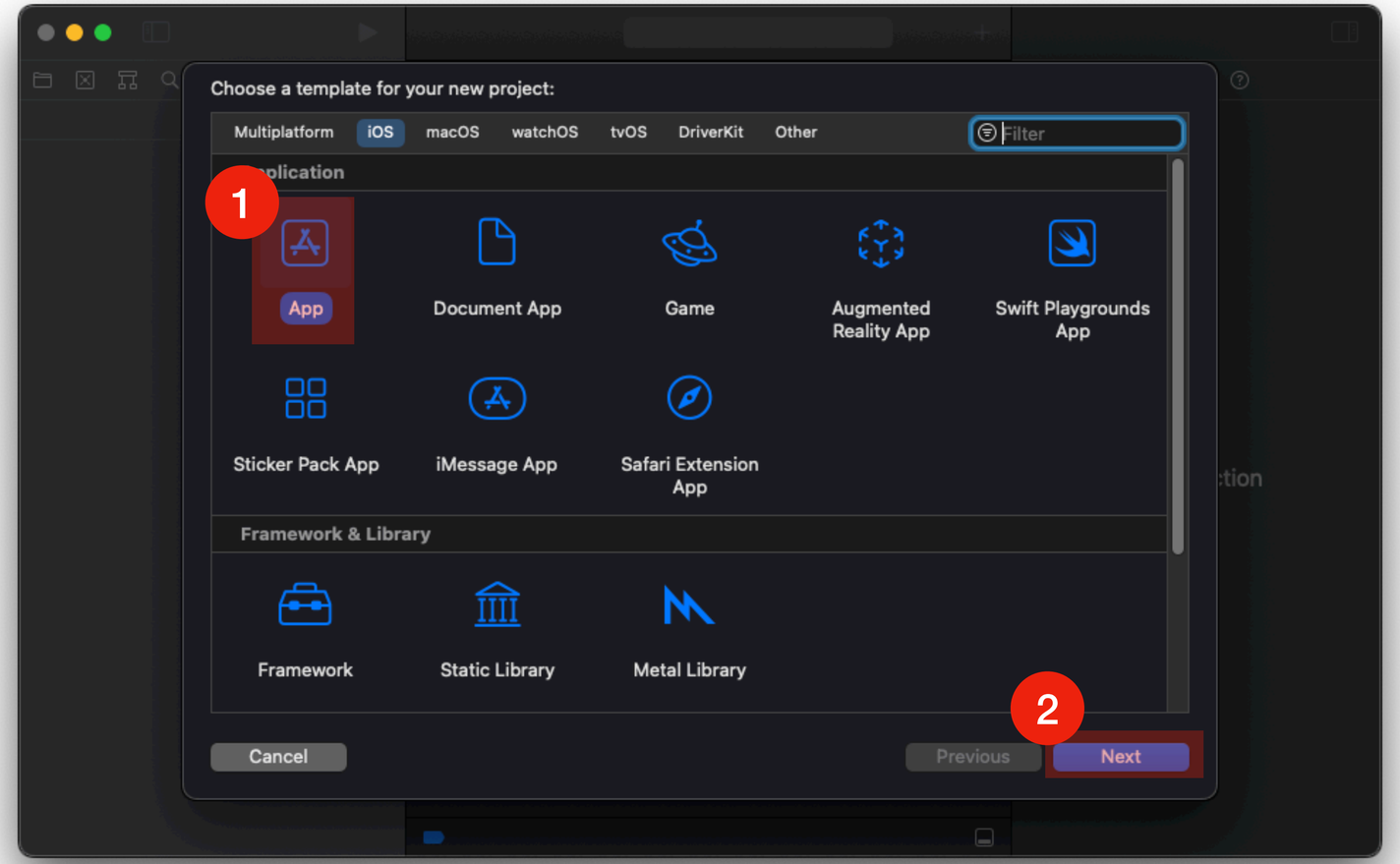


Get ready to dive into SwiftUI like a kid in a ⚾ ball pit! 🔥 Fire up Xcode and click on "Create a New Xcode Project"

🔴 Red dots with numbers refer to the step. You can follow along!

Get Started! 🔥

Math Game - ***ADVANCE***

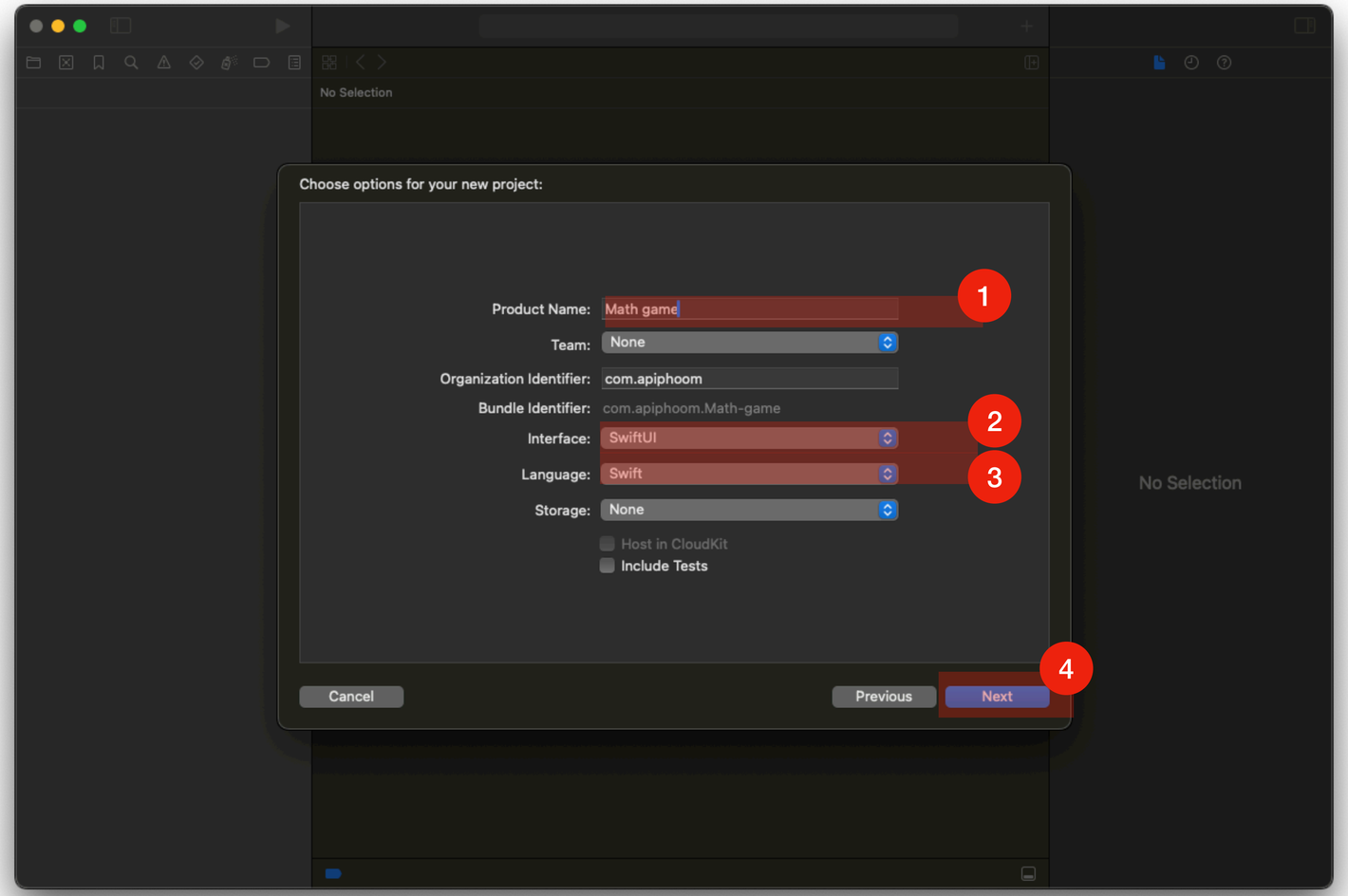


📌 **Choose the iOS App template to get started, then click next**

🔴 Red dots with numbers refer to the step. You can follow along!

Get Started! 🔥

Math Game - ***ADVANCE***

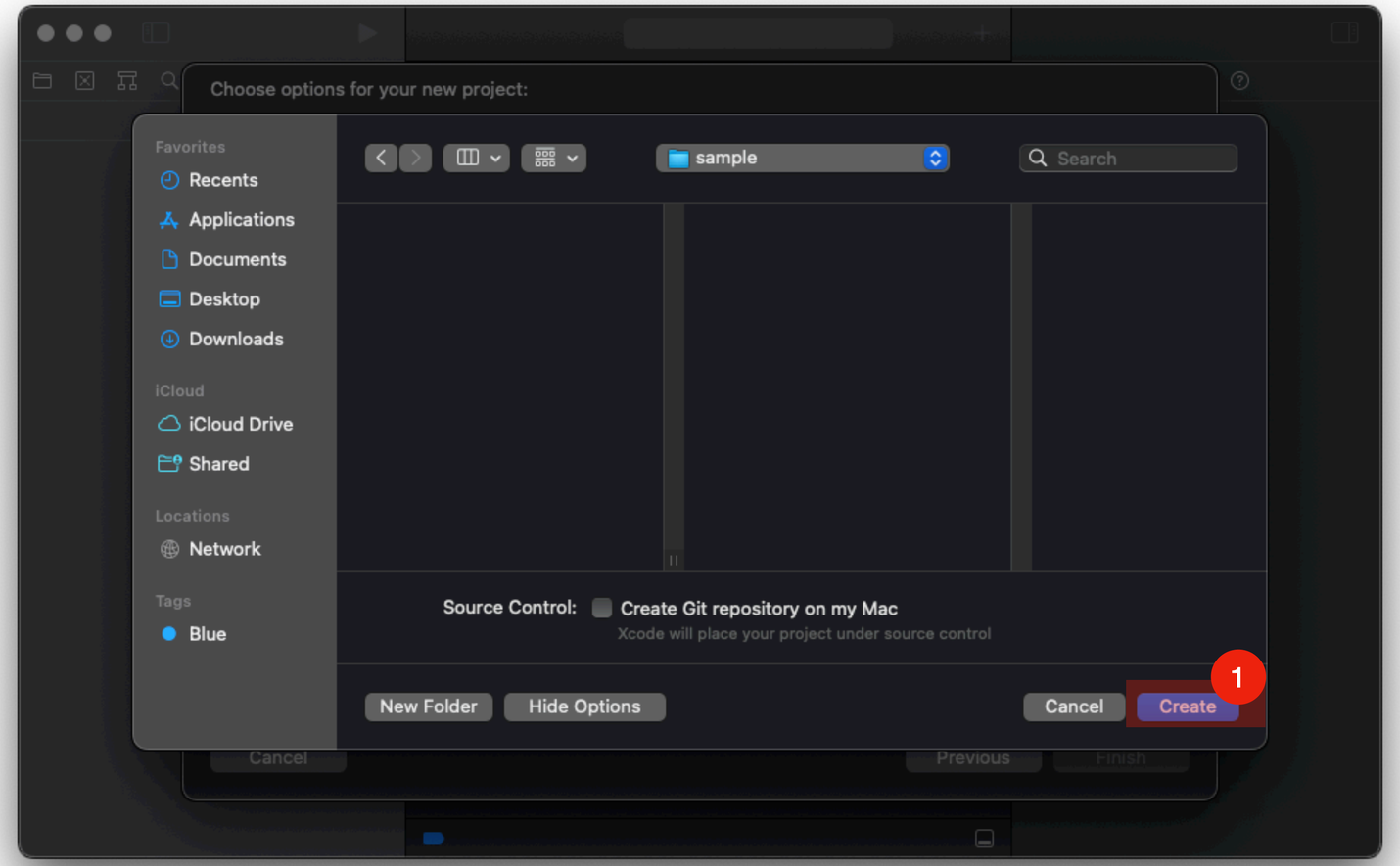


Let's give our project a fun 😆 name like “Math game” and don't forget to set the interface option to SwiftUI. Also, select Swift as the language.

Red dots with numbers refer to the step. You can follow along!

Get Started! 🔥

Math Game - ***ADVANCE***

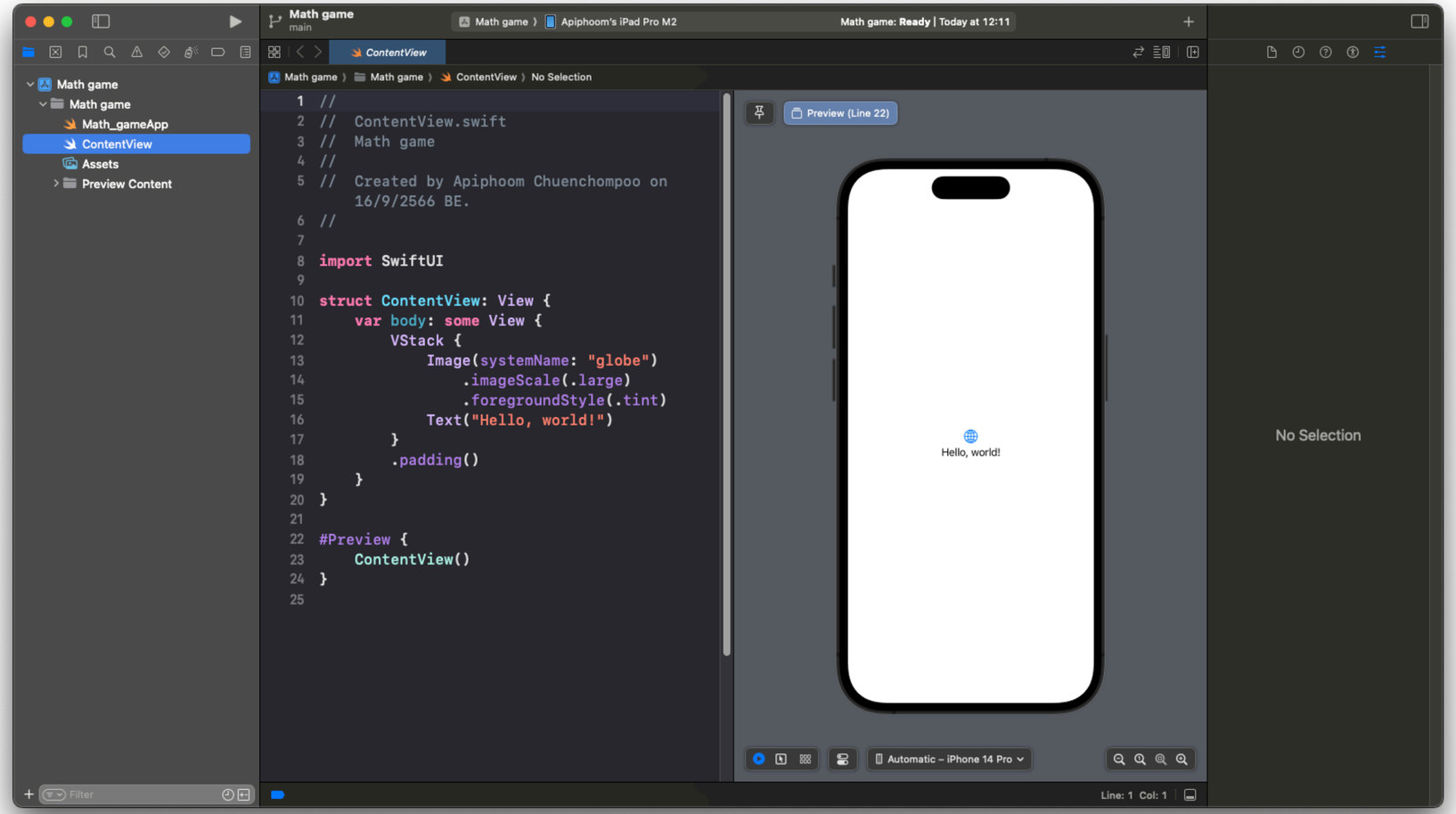


💾 **Save your new project "Recipes to any directory on your Mac that you prefer. Create a Git repository for source control is optional.**

🔴 **Red dots with numbers refer to the step. You can follow along!**

Get Started! 🔥

Math Game - ***ADVANCE***

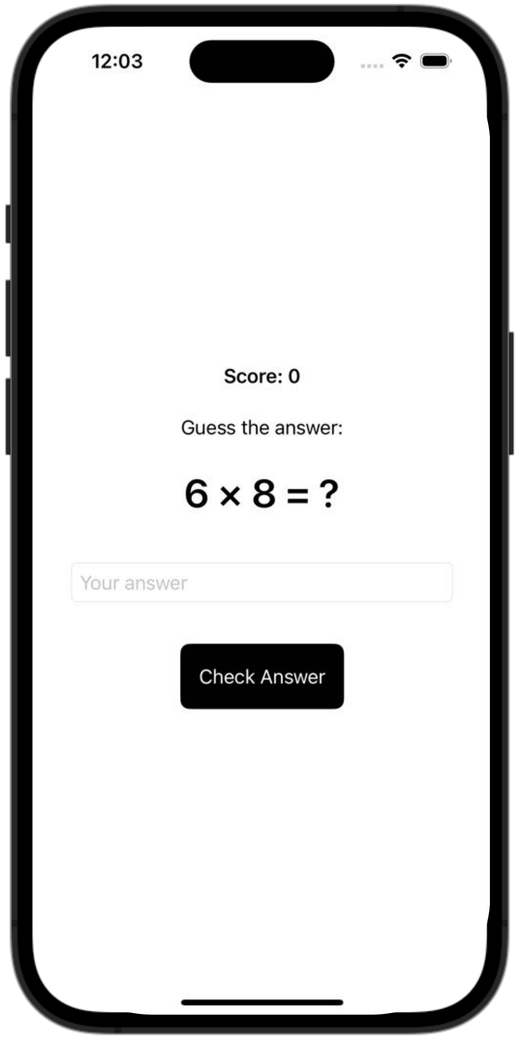


After that, you will see this page, congratulations! 🎉

Swift Coding Club *Thailand*

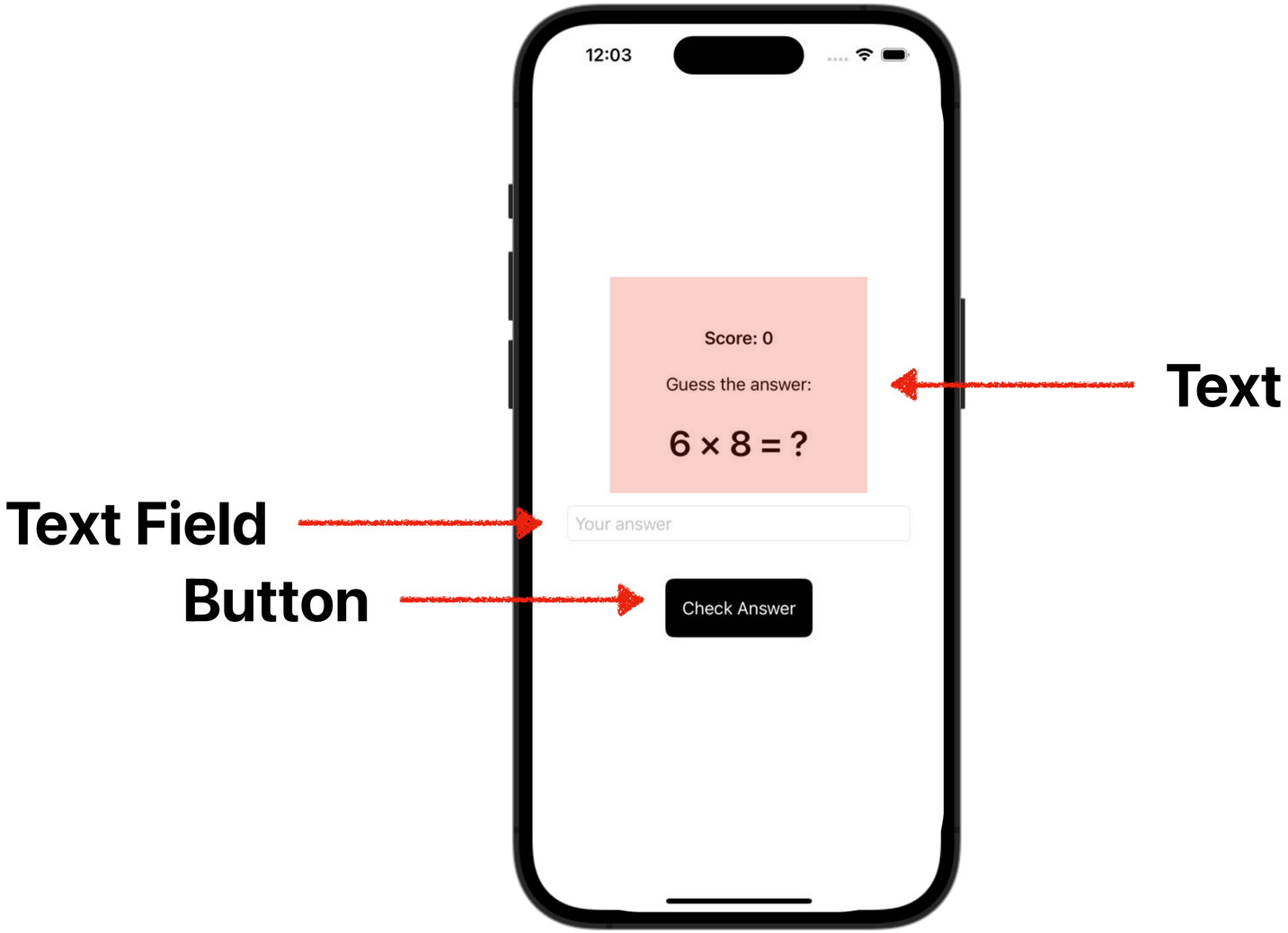
Math Game - ***ADVANCE***

Math x game



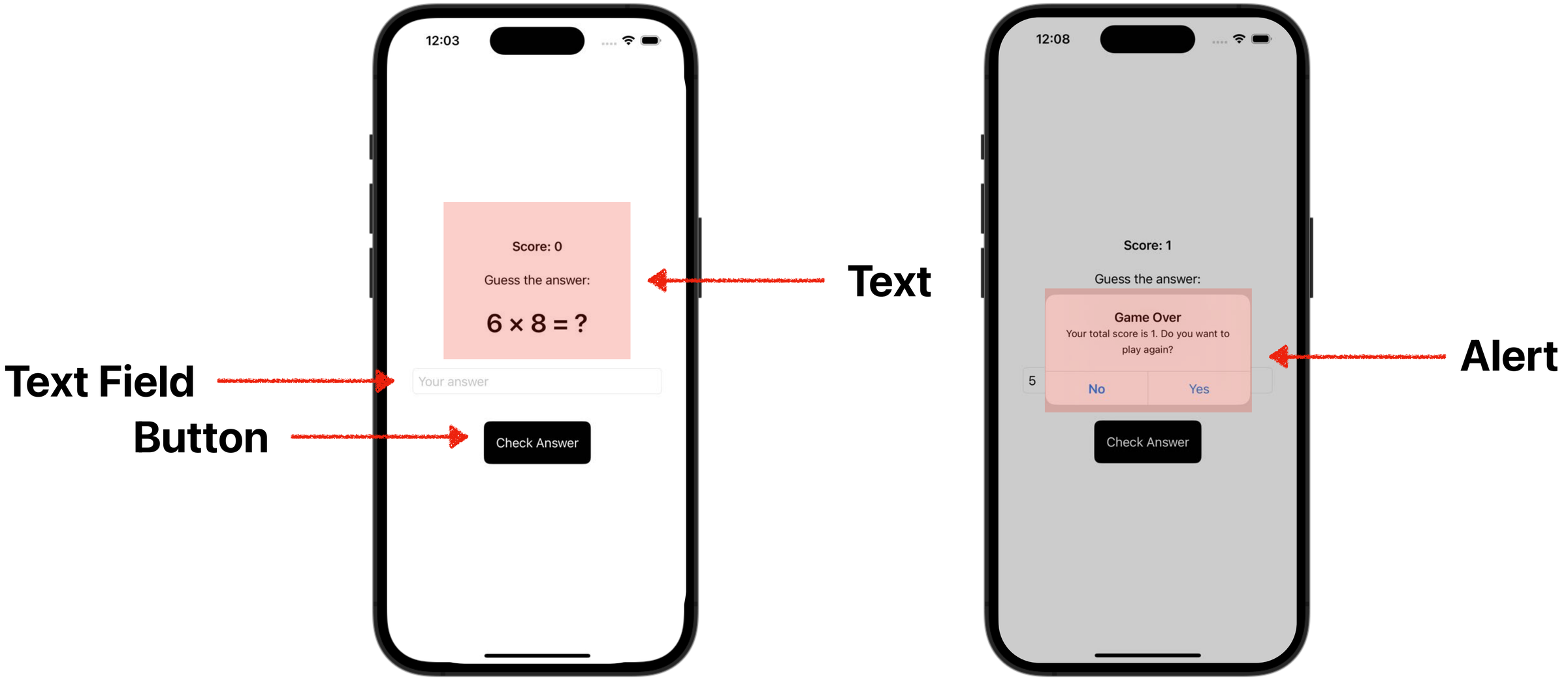
Swift Coding Club *Thailand*

Math Game - ***ADVANCE***



Swift Coding Club *Thailand*

Math Game - ***ADVANCE***



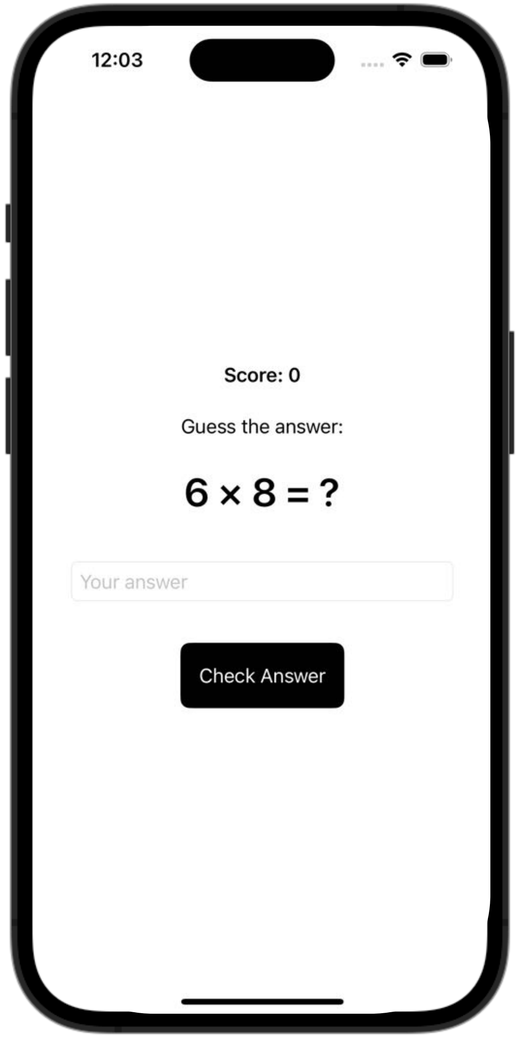
Math x game 🎮✖

Math Game - ***ADVANCE***

Game View

Design a **SwiftUI** game view displaying math challenges and tracking user scores

🎮 Game View



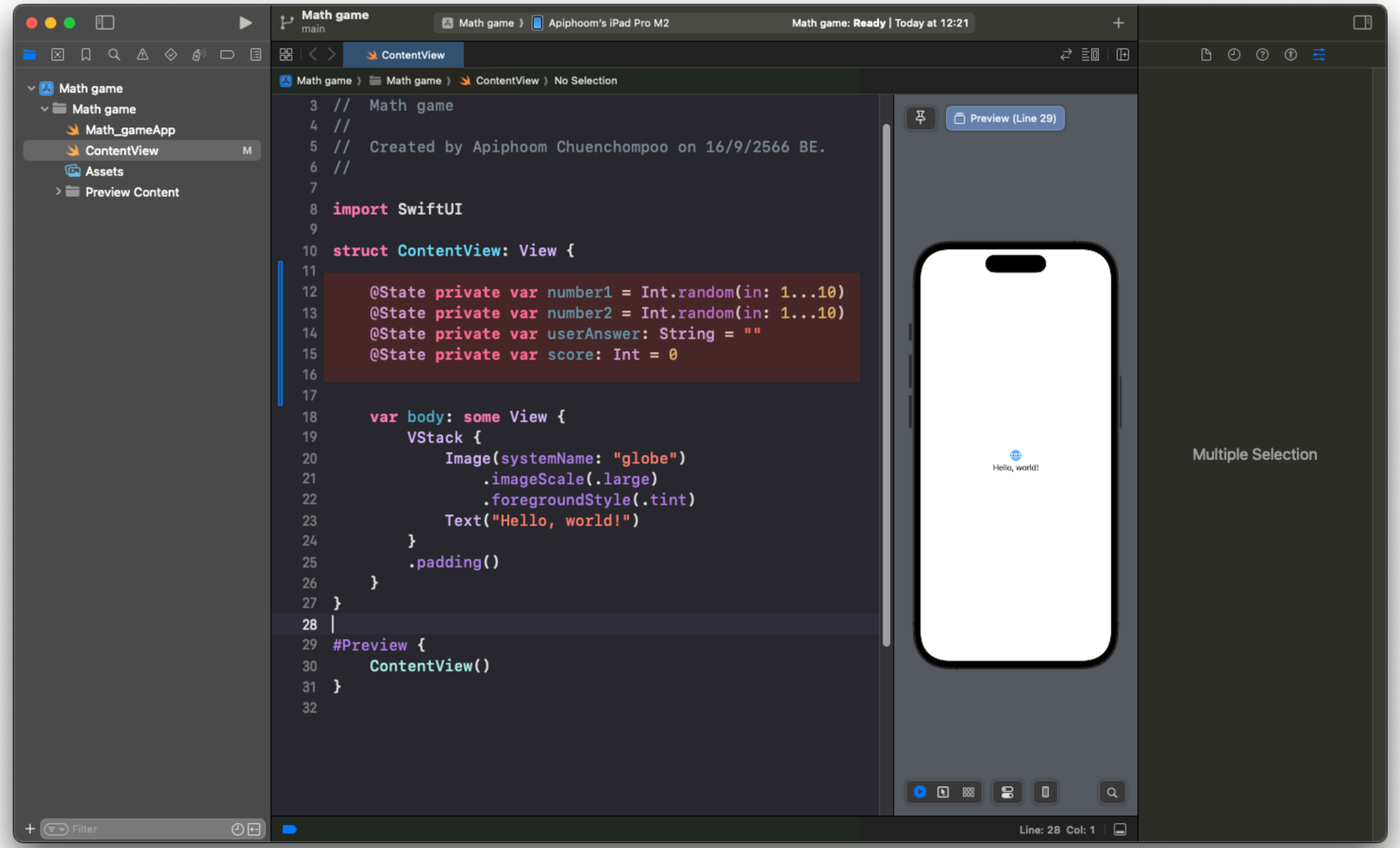
Math x game 🎮✖

Math Game - ***ADVANCE***

Step 1

State Variables

- These are state variables that store dynamic data the view uses



Math x game 🎮✖

Math Game - ***ADVANCE***

```swift
@State private var number1 = Int.random(in: 1...10)
@State private var number2 = Int.random(in: 1...10)
@State private var userAnswer: String = ""
@State private var score: Int = 0
```

Step 1 - Explain

- **number1** and **number2**: Random integers between 1 and 10. They represent the numbers for the math question.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```swift
@State private var number1 = Int.random(in: 1...10)
@State private var number2 = Int.random(in: 1...10)
@State private var userAnswer: String = ""
@State private var score: Int = 0
```

Step 1 - Explain

- **userAnswer**: String to capture the user's answer.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
@State private var number1 = Int.random(in: 1...10)
@State private var number2 = Int.random(in: 1...10)
@State private var userAnswer: String = ""
@State private var score: Int = 0
```

Step 1 - Explain

- **score**: An integer to keep track of the user's score.

Follow the highlight

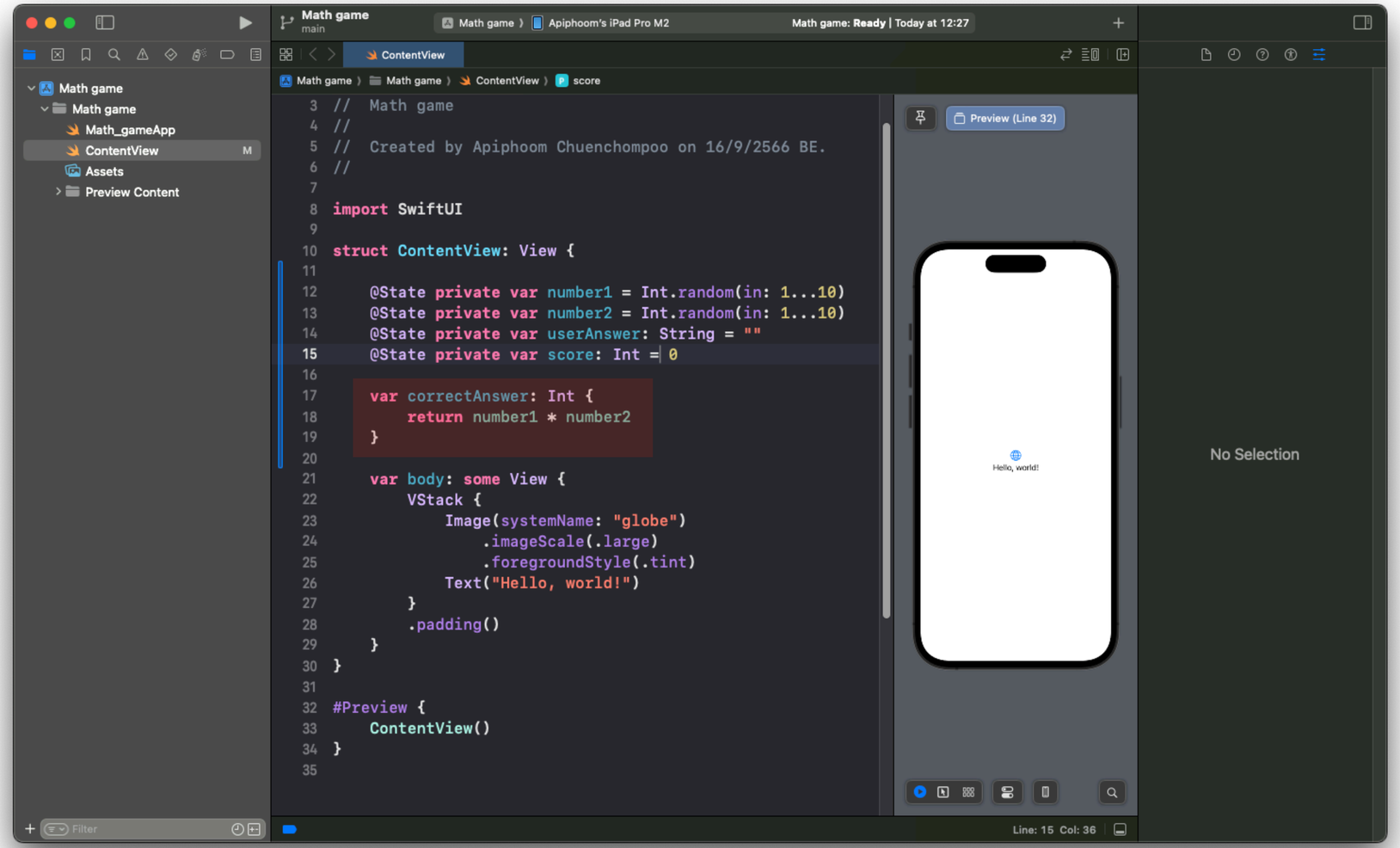
Math x game 🎮✖

Math Game - ***ADVANCE***

Step 2

Computed Property for Correct Answer

- This is a computed property that multiplies **number1** and **number2** to get the correct answer.



Math x game 🎮✖

Math Game - ***ADVANCE***

```
var correctAnswer: Int {
        return number1 * number2
    }
```

Step 2 - Explain

- **correctAnswer** is a computed property. Unlike stored properties that store constant or variable values, computed properties calculate a value every time they're accessed.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
var correctAnswer: Int {
        return number1 * number2
    }
```

Step 2 - Explain

- This is the body of the computed property. Whenever **correctAnswer** is accessed, the code inside this body gets executed.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
var correctAnswer: Int {
        return number1 * number2
    }
```

Step 2 - Explain

- When you access **correctAnswer** in your code, it doesn't fetch a stored value like typical variables. Instead, it calculates the product of **number1** and **number2** on-the-fly and provides that result.

Follow the highlight

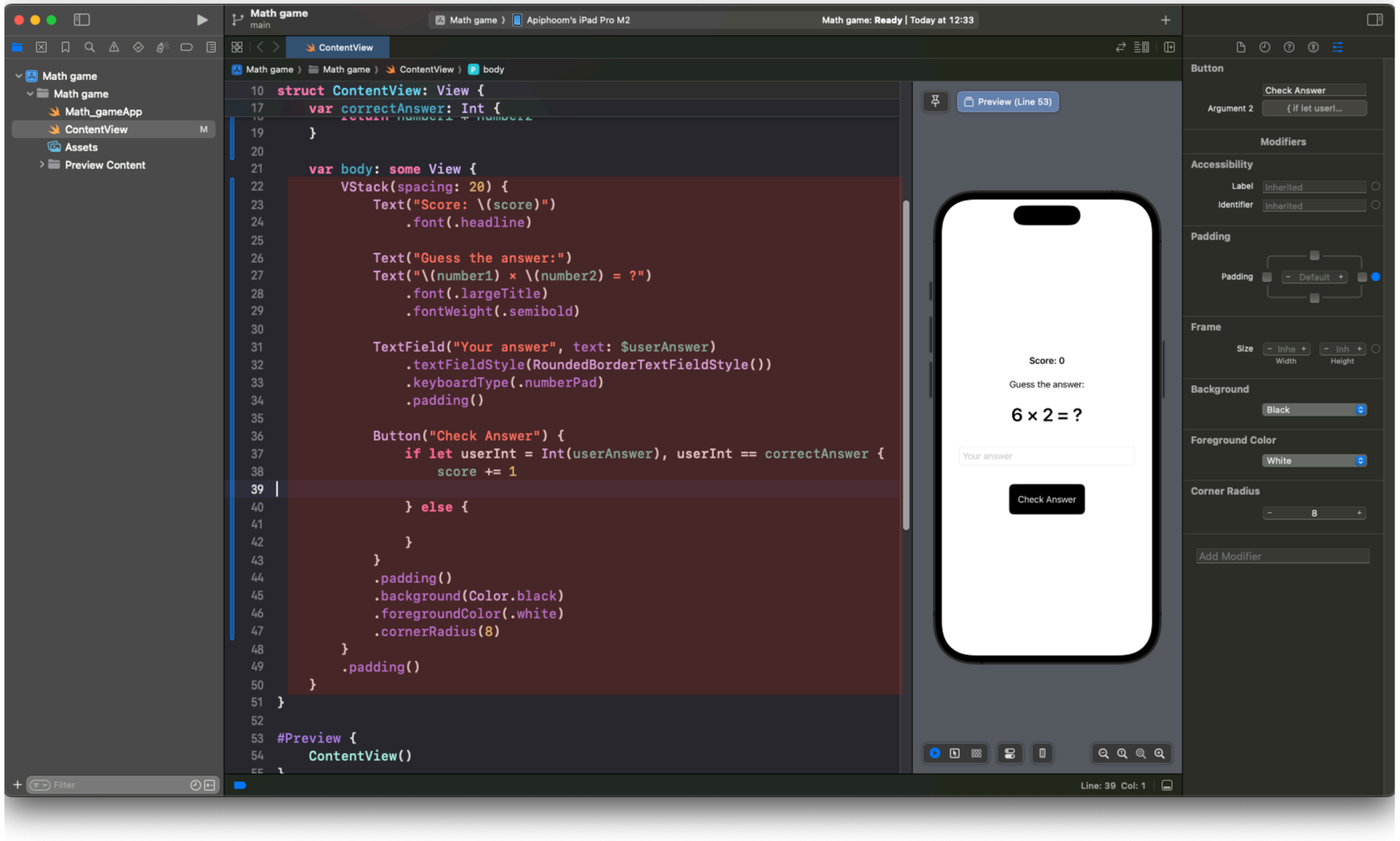
Math x game 🎮✖

Math Game - ***ADVANCE***

Step 3

Body of the ContentView

- Displays the user's score.
- Prompts the user to guess the multiplication result.
- Provides a text field for the user to input their answer.
- Offers a button to check if the answer is correct.




```swift
struct ContentView: View {
    var correctAnswer: Int {
    }

    var body: some View {
        VStack(spacing: 20) {
            Text("Score: \(score)")
                .font(.headline)

            Text("Guess the answer:")
            Text("\(number1) × \(number2) = ?")
                .font(.largeTitle)
                .fontWeight(.semibold)

            TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()

            Button("Check Answer") {
                if let userInt = Int(userAnswer), userInt == correctAnswer {
                    score += 1

                } else {

                }
            }
            .padding()
            .background(Color.black)
            .foregroundColor(.white)
            .cornerRadius(8)
        }
        .padding()
    }
}

#Preview {
    ContentView()
}
```

Math x game 🎮✖

Math Game - ***ADVANCE***

```
VStack(spacing: 20) {

}.padding()
```

Step 3 - VStack - Explain

- **VStack(spacing: 20)** creates a vertical stack of views with a spacing of 20 points between each view. It ensures that each child component is placed below the previous one.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
VStack(spacing: 20) {

}.padding()
```

Step 3 - VStack- Explain

- **.padding()** modifier applied to the **VStack** adds spacing around the entire stack to separate it from the edges of its container or other views.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Text("Score: \(score)")
        .font(.headline)
```

Step 3 - Displays the user's score - Explain

- The **Text** view displays the current score.
- The string "Score: (score)" dynamically includes the value of the **score** variable.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Text("Score: \(score)")
        .font(.headline)
```

Step 3 - Displays the user's score - Explain

- **font(.headline)** changes the font style of the text to "headline", making it stand out more prominently.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Text("Guess the answer:")
```

Step 3 - Prompting the User - Explain

- Another **Text** view simply provides an instruction to the user to guess the answer to the multiplication question.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Text("\(number1) × \(number2) = ?")
                .font(.largeTitle)
                .fontWeight(.semibold)
```

Step 3 - Displaying the Question - Explain

- This **Text** view shows the multiplication problem using the values of **number1** and **number2**.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Text("\(number1) × \(number2) = ?")
                    .font(.largeTitle)
                    .fontWeight(.semibold)
```

Step 3 - Displaying the Question - Explain

- The modifiers **.font(.largeTitle)** and **.fontWeight(.semibold)** ensure that the question is displayed prominently with a larger font and a semi-bold weight.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
  TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()
```

Step 3 - User Input Field - Explain

- The **TextField** is an input field allowing users to type in their answer.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()
```

Step 3 - User Input Field - Explain

- It binds to the **userAnswer** state variable (indicated by **$userAnswer**), meaning that any text inputted by the user will be stored in this variable.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()
```

Step 3 - User Input Field - Explain

- **.textFieldStyle(RoundedBorderTextFieldStyle())** applies a rounded border style to the input field.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()
```

Step 3 - User Input Field - Explain

- **.keyboardType(.numberPad)** ensures that when the input field is focused, only the number pad keyboard appears.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
TextField("Your answer", text: $userAnswer)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .padding()
```

Step 3 - User Input Field - Explain

- **.padding()** adds spacing around the **TextField** for better visual separation.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
 Button("Check Answer") {
            if let userInt = Int(userAnswer), userInt == correctAnswer {
                score += 1
                nextProblem()
            } else {
                showGameOverAlert = true
            }
        }
        .padding()
        .background(Color.black)
        .foregroundColor(.white)
        .cornerRadius(8)
```

Step 3 - User Input Field - Explain

- This is a button labeled "Check Answer".

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Button("Check Answer") {
            if let userInt = Int(userAnswer), userInt == correctAnswer {
                score += 1
                nextProblem()
            } else {
                showGameOverAlert = true
            }
        }
        .padding()
        .background(Color.black)
        .foregroundColor(.white)
        .cornerRadius(8)
```

Step 3 - User Input Field - Explain

- Inside the button's action closure (between the curly braces), there's logic to check if the user's answer matches the correct answer. If it does, it increments the score and loads the next problem. If not, it triggers a game-over alert.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
Button("Check Answer") {
            if let userInt = Int(userAnswer), userInt == correctAnswer {
                score += 1
                nextProblem()
            } else {
                showGameOverAlert = true
            }
        }
        .padding()
        .background(Color.black)
        .foregroundColor(.white)
        .cornerRadius(8)
```

Step 3 - User Input Field - Explain

- The modifiers below the closure **.padding()**, **.background(Color.black)**, **.foregroundColor(.white)**, and **.cornerRadius(8)** style the button by giving it padding, a black background color, white text color, and rounded corners.

Follow the highlight

Math x game 🎮✖

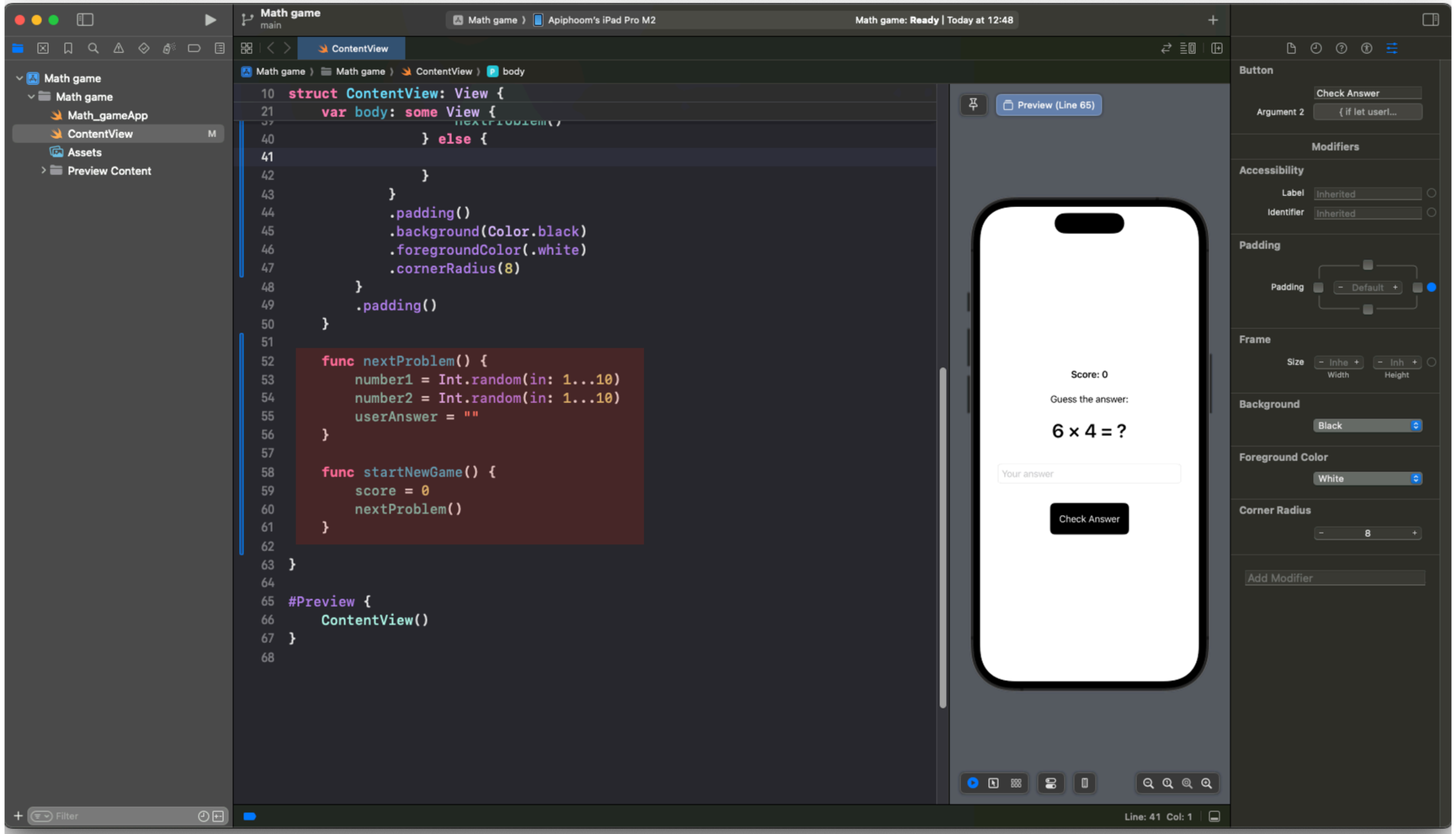
Math Game - ***ADVANCE***

Step 4

Helper Functions

These functions help in controlling the flow of the game

- **nextProblem()**: Loads a new math problem.
- **startNewGame()**: Resets the score and presents a new problem.



Math x game 🎮✖

Math Game - ***ADVANCE***

```
func nextProblem() {
    number1 = Int.random(in: 1...10)
    number2 = Int.random(in: 1...10)
    userAnswer = ""
}
```

Step 4 - nextProblem() Function - Explain

- **number1 = Int.random(in: 1...10)**: This line assigns a random integer value between 1 and 10 (inclusive) to the **number1** variable.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
func nextProblem() {
    number1 = Int.random(in: 1...10)
    number2 = Int.random(in: 1...10)
    userAnswer = ""
}
```

Step 4 - nextProblem() Function - Explain

- **number2 = Int.random(in: 1...10)**: Similarly, this line assigns a random integer value between 1 and 10 (inclusive) to the **number2** variable.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
func nextProblem() {
    number1 = Int.random(in: 1...10)
    number2 = Int.random(in: 1...10)
    userAnswer = ""
}
```

Step 4 - nextProblem() Function - Explain

- **userAnswer = ""**: Resets the **userAnswer** variable by assigning it an empty string. This is important to ensure that the previous answer entered by the user doesn't remain in the input field for the next problem.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
func startNewGame() {
    score = 0
    nextProblem()
}
```

Step 4 - startNewGame() Function - Explain

- **score = 0**: This line resets the **score** variable back to 0. It ensures that when a new game starts, the player's score is cleared and they begin from scratch.

Follow the highlight

Math x game 🎮✖

Math Game - ***ADVANCE***

```
func startNewGame() {
    score = 0
    nextProblem()
}
```

Step 4 - startNewGame() Function - Explain

- **nextProblem()**: This line calls the previously described **nextProblem()** function. As a result, when initiating a new game, not only is the score reset, but a fresh math problem is also presented to the user.

Follow the highlight

Swift Coding Club *Thailand*