COMP 1601A: Mobile Application Development

Carleton University
Winter 2021 Final Exam

April 23, 2021

There are seven questions on **four pages** (+9 code listing pages) worth a total of 48 points. Answer all questions in a text file following the supplied template. If you find a question is ambiguous, explain your interpretation and answer the question accordingly.

This test is open book. You may use previous tutorials and assignments, your notes, and any generally available online resources. However, you **may not collaborate, work with, or get assistance from anyone**, including but not limited to anyone in this class. *Remember there will be selected post-exam interviews where you may be asked how you came up with your answers.*

Be sure to monitor the Announcements channel on Teams, as clarifications or corrections to the exam will be posted there. Do not use any other channels on Teams (or any other communication methods) during the exam to discuss anything related to the exam, except to talk with the instructor or TAs.

Cite any sources that you use that were not part of or referenced directly referenced in class. For example, you do not need to cite the official Swift documentation, but you do need to cite a random stack overflow answer if you used it to come up with your answer. *No matter the source, the words you use should be your own.*

You have 120 minutes. Good luck!

1. [2] When an Android activity is "destroyed", are the state variables of the activity's class preserved? How do you know? Explain briefly.

2. [2] Give an example in Kotlin of converting a nullable type to a non-nullable type safely. Why is your example safe?

3. [2] Give an example in Swift of converting an optional type to a regular type safely. Why is your example safe?

4. [2] In Assignment 3 it was easy to add a "Cars to Cows" conversion option to Coverter2A that had no corresponding Converter object. You are curious about how hard this would be to do in the SwiftUI version. You ask a friend and they say "It doesn't work because we used an exclamation point in Converter2." What is your friend talking about? Why is this a potential problem? (See the attached listings for Converter2, Conversions.swift and ContentView.swift.)

5. [2] Rotations in Android can cause a loss of state, as we saw in the first Android TapDemo (Lecture 16). What is happening under the hood to cause the loss of state? How can we preserve state across rotations?

6. [4] Android activities:

   (a) [2] If I wanted to add a new activity to an Android app, I have to modify one file and add two files. What existing file has to be modified? What two files have to be added? And how does Android know to use these files?

   (b) [2] If I wanted to run this new activity, would I use an explicit or an implicit intent? What arguments would I give to the intent?

7. [2] Assume we have a program that includes **Converter.kt**. How can we convert `ks` to `m`, a `Double` which contains the number of miles equal to the number of kilometers in `ks`? Be sure to consider the following constraints:

   - Declare m as a constant.
   - We're given `ks` of type String which may **or may not** contain a valid number.
   - If `ks` does not contain a valid number, miles should be equal to 1000.
   - Your code should make proper use of the code in **Converter.kt** (i.e., don't hard code the conversion, use the one provided).

8. [4] Consider the following code from Contentview.swift from iOS TapDemo (Lecture 16):

```
struct ContentView: View {
    @State var count = 0
    var body: some View {
        VStack{
            Button("Tap here!", action: onTapped)
            Text("You have tapped \(count) times.")
        }
    }

    func onTapped() {
        count = count + 1
    }
}
```

   (a) [2] When is the function `onTapped()` called? How do you know?

   (b) [2] When is the on-screen count updated? How do you know?

9. [4] Consider the following code from MainActivity.kt from Android TapDemo (Lecture 16):

```
class MainActivity : AppCompatActivity() {
    private lateinit var myMessage: TextView
    private var count = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        myMessage = findViewById(R.id.myMessage)
    }

    fun myButtonPressed(v: View) {
        count++
        myMessage.text = "You clicked $count times."
    }
}
```

   (a) [2] When is the function myButtonPressed called? How do you know?

   (b) [2] When is the on-screen count updated? How do you know?

10. [6] Consider the following layout declaration from activity_main.xml from Android PicViewer2A (Tutorial 8):

```
<EditText
     android:id="@+id/y"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginBottom="32dp"
     android:ems="5"
     android:inputType="number"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintStart_toStartOf="@+id/ylabel" />
```

(a) [2] What kind of input does this EditText allow to be inputted? Specify the characters that may be entered. How do you know?

(b) [2] How does the above specify how wide the entry field will be? What are the units of this width?

(c) [2] Where will this widget be placed relative to the parent (containing) view? How do you know? Be specific.

11. [6] Answer the following questions about code from **Converter.kt**:

```
class Converter {
    var convFrom = "From Type"
    var convTo = "To Type"
    var convert: (from: Double) -> Double?

    fun formatConversion(from: Double): String {
        val to = this.convert(from)
        if (to != null) {
            return "${from} ${this.convFrom} is ${to} ${this.convTo}."
        } else {
            return "Converting ${from} ${convFrom} to ${this.convTo}
                failed."
        }
    }

    constructor(from: String, to: String, f: (from: Double) -> Double?) {
        this.convFrom = from
        this.convTo = to
        this.convert = f
    }
}
```

(a) [1] What type is `convert`?

(b) [2] Can `convert` be given a value of null? Why?

(c) [1] When will `formatConversion()` return the failed message?

(d) [1] What type is the `f` argument to `constructor()`?

(e) [1] Give an example of calling `constructor()` normally.

12. [6] Answer the following about `theImage` in picviewer-1's **Contentview.swift**:

    (a) [2] Where is `theImage` declared? Where is it used?

    (b) [1] What is the value of `theImage` used for, precisely?

    (c) [3] What is equivalent to `theImage` in PicViewer2A (the Tutorial 8 solutions version)? Is it of the same type as `theImage`? Explain briefly.

13. [6] Answer the following about lines 53–54 of picviewer-1's **Contentview.swift**:

```
.position(moved ? position :
          CGPoint(x: g.size.width / 2, y: g.size.height / 2))
```

    (a) [1] What are the possible values of `moved`?

    (b) [2] What would happen if we just replaced this code with `.position(position)`? Why?

    (c) [3] What code in Android's PicViewer2A **MainActivity.kt** is equivalent to this code? When is this code run? How does this compare to when the above swift code is run? (See the attached code from the Tutorial 8 solutions for PicViewer2A.)

End of exam. Code listings begin on the next page.

Converter2A/.../Converter.kt

```kotlin
package carleton.comp1601.converter_2a

//  Converter.kt

class Converter {
    var convFrom = "From Type"
    var convTo = "To Type"
    var convert: (from: Double) -> Double?

    fun formatConversion(from: Double): String {
        val to = this.convert(from)
        if (to != null) {
            return "${from} ${this.convFrom} is ${to} ${this.convTo}."
        } else {
            return "Converting ${from} ${convFrom} to ${this.convTo} failed."
        }
    }

    constructor(from: String, to: String, f: (from: Double) -> Double?) {
        this.convFrom = from
        this.convTo = to
        this.convert = f
    }
}

fun InToCM(inch: Double): Double? {
    val cm = 2.54 * inch
    return cm
}

val conversions = mapOf(
        "F to C" to Converter("Farenheit", "Celsius",
            {F ->  ((F - 32.0)*(5.0/9.0))}),
        "C to F" to Converter("Celsius","Farenheit",
            {((9.0/5.0)*it + 32.0)}),
        "km to mi" to Converter("kilometers", "miles",
            fun (k: Double): Double? {
                val m = k * 0.6213712
                return m
            }),
        "inch to cm" to Converter("inches", "centimeters", ::InToCM))
```

Converter2/Shared/Conversions.swift

```swift
//  Conversions.swift

struct Converter {
    var convFrom = "From Type"
    var convTo = "To Type"
    var convert: (_ from: Double) -> Double?

    func formatConversion(_ from: Double) -> String {
        if let to = self.convert(from) {
            return "\(from) \(self.convFrom) is \(to) \(self.convTo)."
        } else {
            return "Converting \(from) \(convFrom) to \(self.convTo) failed."
        }
    }

    init(_ from: String, _ to: String, _ f: @escaping (_ from: Double) -> Double?) {
        self.convFrom = from
        self.convTo = to
        self.convert = f
    }
}

func InToCM(_ inch: Double) -> Double {
    let cm = 2.54 * inch
    return cm
}

let conversions: [String: Converter] =
    ["F to C": Converter("Farenheit", "Celsius",
                         {F in return ((F - 32.0)*(5/9))}),
     "C to F": Converter("Celsius", "Farenheit",
                         {C in return ((9/5)*C + 32.0)}),
     "km to mi": Converter("kilometers", "miles",
                           {k in
                            let m = k * 0.6213712
                            return m
                           }),
     "inch to cm": Converter("inches", "centimeters",
                             InToCM)
    ]
```

Converter2/Shared/ContentView.swift

```swift
// ContentView.swift (Converter2)

import SwiftUI

struct ContentView: View {
    @State var convMode: String? = nil
    @State var fromS = ""
    var body: some View {
        VStack{
            ConvMenu(convMode: $convMode,
                     fromString: $fromS)
            Spacer()
            ConvOutput(convMode: $convMode,
                       fromString: $fromS)
            Spacer()
            Spacer()
        }
    }
}

struct ConvMenu: View {
    @Binding var convMode: String?
    @Binding var fromString: String
    var body: some View {
        let availConversions = [String](conversions.keys)
        Menu("Conversions menu") {
            ForEach(availConversions, id: \.self) {
                conversion in
                Button(conversion, action: {
                    convMode = conversion
                    fromString = ""
                })
            }
        }
    }
}

struct ConvOutput: View {
    @Binding var convMode: String?
    @Binding var fromString: String
    var body: some View {
        if let mode = convMode {
            let conv = conversions[mode]!
            TextField("Enter \(conv.convFrom)", text: $fromString)
            Spacer()
            if let from = Double(fromString) {
                Text(conv.formatConversion(from))
            }
        } else {
            Text("Please select a conversion")
            Text("type from the menu above.")
        }
    }
}

```

```
56 struct ContentView_Previews: PreviewProvider {
57     static var previews: some View {
58         ContentView()
59     }
60 }
```

picviewer-1/picviewer-1/ContentView.swift

```
1  //
2  //  ContentView.swift
3  //  picviewer-1
4  //
5  //  Created by Anil Somayaji on 2/15/21.
6  //
7
8  import SwiftUI
9
10 struct ContentView: View {
11     @State private var theImage = "kittens"
12     @State private var moved = false
13     @State private var finalAmount: CGFloat = 1
14     @State private var angle = Angle(degrees: 0.0)
15
16     func resetState() {
17         moved = false
18         finalAmount = 1
19         angle = Angle(degrees: 0.0)
20     }
21
22     var body: some View {
23         VStack {
24             Menu("Animals!") {
25                 Button("Kittens", action: {
26                     theImage = "kittens"
27                     resetState()
28                 })
29                 Button("Sad Dog", action: {
30                     theImage = "sadDog"
31                     resetState()
32                 })
33             }
34             ActiveImage(theImage: $theImage, moved: $moved, finalAmount:
                   $finalAmount, angle: $angle)
35         }
36     }
37 }
38
39 struct ActiveImage: View {
40     @State private var position = CGPoint(x: 0, y: 0)
41     @State private var currentAmount: CGFloat = 0
42
43     @Binding var theImage: String
44     @Binding var moved: Bool
45     @Binding var finalAmount: CGFloat
46     @Binding var angle: Angle
47
48     var body: some View {
49         GeometryReader {g in
50             Image(theImage)
51                 .resizable()
52                 .scaledToFit()
53                 .position(moved ? position :
54                             CGPoint(x: g.size.width / 2, y: g.size.height / 2))
```

```
                .scaleEffect(finalAmount + currentAmount)
                .gesture(dragging)
                .gesture(SimultaneousGesture(magnifying, rotating))
                .rotationEffect(self.angle)
                .onTapGesture(count: 1, perform: tapReset)
        }
    }

    func tapReset() {
        self.finalAmount = 1
        self.moved = false
        self.angle = Angle(degrees: 0.0)
    }

    var rotating: some Gesture {
        RotationGesture()
            .onChanged { angle in
                self.angle = angle
            }
    }

    var magnifying: some Gesture {
        MagnificationGesture().onChanged { amount in
            self.currentAmount = amount - 1
        }
        .onEnded { amount in
            self.finalAmount += self.currentAmount
            self.currentAmount = 0
        }
    }

    var dragging: some Gesture {
        DragGesture()
            .onChanged {s in
                self.moved = true
                self.position = s.location
            }
            .onEnded {s in
                self.moved = true
                self.position = s.location
            }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

PicViewer2A-tut8sol/.../MainActivity.kt

```
1  package carleton.comp1601.picviewer2a
2
3  import androidx.appcompat.app.AppCompatActivity
4  import android.os.Bundle
5  import android.text.Editable
6  import android.text.TextWatcher
7  import android.util.Log
8  import android.view.View
9  import android.widget.EditText
10 import android.widget.ImageView
11
12 val appName = "PicViewer2A"
13 val pics = arrayOf(R.drawable.kittens, R.drawable.roshi, R.drawable.roshi2)
14
15 class MainActivity : AppCompatActivity() {
16     private lateinit var p: ImageView
17     private var pic = R.drawable.kittens
18     private var picIndex = 0
19     private lateinit var rotationInput: EditText
20     private lateinit var scaleInput: EditText
21     private lateinit var pXInput: EditText
22     private lateinit var pYInput: EditText
23
24     val r = PicWatcher("rotation", 0F, ::updateImage)
25     {r ->
26         if (r > 720F) return@PicWatcher 720F
27         if (r < 0F) return@PicWatcher 0F
28         r
29     }
30     val s = PicWatcher("scale", 1F, ::updateImage)
31     {s ->
32         if (s > 5F) return@PicWatcher 5F
33         if (s < 0) return@PicWatcher 0F
34         s
35     }
36     val X = PicWatcher("X", 0F, ::updateImage, ::clamp_xy)
37     val Y = PicWatcher("Y", 0F, ::updateImage, ::clamp_xy)
38
39     fun clamp_xy(v: Float): Float {
40         if (v > 1000F) return 1000F
41         if (v < -1000F) return -1000F
42         return v
43     }
44
45     override fun onCreate(savedInstanceState: Bundle?) {
46         super.onCreate(savedInstanceState)
47         setContentView(R.layout.activity_main)
48         if (savedInstanceState != null) {
49             // do something
50         }
51
52         p = findViewById(R.id.mypic)
53
54         rotationInput = findViewById(R.id.rotation)
55         rotationInput.addTextChangedListener(r)
```

```
56
57          scaleInput = findViewById(R.id.scale)
58          scaleInput.addTextChangedListener(s)
59
60          pXInput = findViewById(R.id.x)
61          pXInput.addTextChangedListener(X)
62
63          pYInput = findViewById(R.id.y)
64          pYInput.addTextChangedListener(Y)
65
66          updateImage()
67          Log.d(appName, "Activity Created")
68      }
69
70      override fun onSaveInstanceState(outState: Bundle) {
71          super.onSaveInstanceState(outState)
72          Log.d(appName, "State saved")
73      }
74
75      fun updateImage() {
76          p.setImageResource(pic)
77          p.setX(X.value)
78          p.setY(Y.value)
79          p.setRotation(r.value)
80          p.setScaleX(s.value)
81          p.setScaleY(s.value)
82      }
83
84      fun nextImage(v: View) {
85          picIndex = (picIndex + 1) % pics.size
86          pic = pics[picIndex]
87          updateImage()
88      }
89  }
90
91  class PicWatcher: TextWatcher {
92      var value: Float
93      var default_value: Float
94      var name: String
95      var update: () -> Unit
96      var clamp: (Float) -> Float
97
98      override fun afterTextChanged(s: Editable) {
99          val new_value = s.toString().toFloatOrNull()
100
101         if (new_value != null) {
102             value = clamp(new_value)
103             Log.d(appName, "Set ${name} to ${value}.")
104         } else {
105             value = default_value
106             Log.d(appName, "Set ${name} to default value ${value}.")
107         }
108
109         update()
110     }
111
112     override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int
```

```
            ) {
113     }
114
115     override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after:
            Int) {
116     }
117
118     constructor(n: String, v: Float, u: () -> Unit, c: (Float) -> Float) {
119         value = v
120         default_value = v
121         name = n
122         update = u
123         clamp = c
124     }
125 }
```