CS 359 – Programming Paradigms

PEX 4 – Android App

Preliminary Submission Due: Taps, Lesson M37, Tuesday, December 4

Final Submission Due: Taps, Lesson T39, Tuesday, December 11

Help Policy:

AUTHORIZED RESOURCES: Any, except another cadet's program.

NOTE:

- Never copy another person's work and submit it as your own.
- Do not jointly create a program unless explicitly allowed.
- You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook).
- **DFCS will recommend a course grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.**

Documentation Policy:

- You must document all help received from any source other than your instructor.
- The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
- If no help was received on this assignment, the documentation statement must state "NONE."
- If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why.
- **Vague documentation statements must be corrected before the assignment will be graded and will result in a 5% deduction on the assignment.**

Turn-in Policies:

- On-time turn-in is at the specific time listed above.
- Late penalties accrue at a rate of 25% per 24-hour period past the on-time turn-in date and time. The late penalty is a cap on the maximum grade that may be awarded for late work.
- There is no early turn-in bonus or extra credit for this assignment.

1. OBJECTIVES

- Expand your range of programming language competence
- Gain experience working as part of a team on a medium-sized project
- Frame and resolve an ill-defined problem using your own creativity
- Apply advanced Java programming skills and general programming language skills
- Learn a new API using documentation and examples
- Create an Android app that runs on a handheld device

2. BACKGROUND

This is a very open-ended assignment. The intent is to provide an opportunity for you to demonstrate ability to learn a new technology quickly, to dig into documentation, to glean necessary information from sample code, and to come up with a cool idea or two. Some specifics are below, but beyond those use your imagination. Something that would be of use or interest to you as a cadet would be great, but it is not required.

This is a team programming project with *team size to be exactly two* (yes, this means you *must* work with a partner). Partners will be assigned based in equal parts on student requests, course standing, and instructor discretion. Prior to our next lesson you must send an email to your instructor with your partner preferences. You should send *at least two options*, if not more, for partners you would like to work with and your instructor will make every attempt to accommodate your request. You may also indicate anyone you would not like to be partnered with.

Given the nature of this project and two of the primary objectives (learn new technology and work with a partner), you may not submit an application that you have previously developed, either in full or in part. *You must come up with a new idea and create your application from scratch*.

3. MINIMUM REQUIREMENTS

Given the time constraints many cadets face at the end of a semester, this exercise will be broken into a set of minimum requirements and enhanced requirements. The minimum requirements, if fully satisfied, will earn 80% of the points available.

For 80% credit, your app must incorporate ***all*** of the following:

a) Use widgets drawn from at least three different classes (e.g. Button, TextView, EditText, etc.)
b) Have some form of user interaction. More than half of your widgets must have associated listener methods that are called in response to some sort of user event.
c) Enable and disable widgets as appropriate, both so the user can never put the app in some sort of nonsensical state and so it is always clear what the user can and cannot do.
d) Never crashes, regardless what is done to it. This means exception handling and properly considering possible events in the Android Activity Lifecycle. (For example, what happens to your app if the user receives and answers a phone call while your app is running?)

4. ENHANCED REQUIREMENTS

For 100% credit, your app must also (a) use at least one custom component created by extending an existing component (similar to the examples demonstrated in class) and (b) incorporate ***one*** of the following:

i) Use at least two different screens that pass information between them.
ii) Use multiple threads (see Designing for Responsiveness link later in this document).
iii) Use a system resource such as the GPS, a database, the accelerometer, Bluetooth, access a web resource, etc. Note: Many of these can be emulated, but some require an actual Android device.
iv) Use appropriate Android design practices to allow your application to be localized to multiple regions.
v) Use another Android feature, approved by your instructor.

5. GETTING STARTED

Before you can do any Android development, you must download and install the Android SDK, download and configure an Eclipse plugin, and create an Android emulator. To accomplish this, refer to the instructions on the Android SDK Download Page.

Start with the Android Developers website, in particular the Building Your First App page. This is an excellent tutorial covering most of the major things to be considered when building an app.

You should also consider questions such as:

- How is layout handled?
- What are the class names for the various widgets available?
- How are methods assigned to widgets?
- How is the object associated with a particular widget in a View accessed?
- How do buttons work?
- How is input read from a text field?
- What exceptions are thrown if such input is invalid?
- How are widgets enabled and disabled?

You will probably ask all of the above questions, as well as many others. You should be thinking about them now, and have most of them answered or at least know where answers can be found before you begin coding. Asking the right questions and then digging efficiently to find them is something all good computer scientists do.

6. HELPFUL HINTS

Here is one helpful bit of advice concerning layout. The best way to layout your GUI is to do it in XML, following the examples on the android developer web site. There is a special method your program needs to call when first created that loads your XML file and sets up the layout for your app, including all the widgets with appropriate resource IDs. Android expects this file to be in the /res/layout subfolder of your project ("res" stands for "resources"). The advantage of this design is it separates presentation from code. You can change how your GUI looks by simply editing the XML file, without needing to mess with any Java. Your app will still run exactly the same way.

The Android plugin for Eclipse has a WYSIWYG editor for these layouts, which is an excellent way to initially lay out your components. After the initial layout, you may find it useful to edit the XML directly to fine tune your layout.

While far from an exhaustive list, here are some links that may prove useful:

- What is Android?
- Android Application Fundamentals including Android Activity Lifecycle
- Common Android Tasks
- Buttons and Handling UI Events
- The Android Manifest File
- Android Emulator
- Obtaining User Location
- Designing for Responsiveness, Painless Threading, and Multithreading for Responsiveness
- Finally, the Notepad Tutorial seems especially useful.

7. PRELIMINARY EXERCISE

For the preliminary exercise, you are submitting a shell of what will be your final PEX submission. This shell should give a clear idea of what the final functionality of your app will be, including the layout of all screens and widgets with appropriate behaviors attached. These behaviors do not need to be fully functional, but code stubs should exist indicating what each will do.

The preliminary submission must compile and execute on an Android emulator with no errors.

8. PROGRAMMING EXERCISE

Implement the remaining functionality of your Android application.

9. SUBMISSION REQUIREMENTS

- Include your ***documentation statement*** at the top of the source file containing your ***main activity***.
    - Include separate documentations statements for the preliminary and final submissions.
- IMPORTANT: Do not use the default activity name, **`MainActivity`**, for your main class. Name it something descriptive and unique (e.g., **`TicTacToe`**). This is a tremendous help when grading.
    - Note that the Android manifest file will need to references this class name properly.
- Also include in the header block of your main program file a comment detailing what, if any, features of your program satisfy the Enhanced Requirements listed above. Include specific names of classes and methods that should be examined.
    If you fail to do this, you will not get the points for implementing these requirements.
- Zip your entire Eclipse project folder
- Use the website to submit a **single zip file** containing everything you would like to be graded.

CS 359 – PEX 4 Prelim – Grade Sheet Name:_____

Criteria		Points Earned	Points Available
App compiles and executes on an Android emulator with no errors (all or nothing score in this category … not compiling or crashing is bad)			**10**
Sufficient widgets (and screens, if applicable) are present, including at least one that will be enabled/disabled when final app is complete			**10**
Shell gives user clear understanding of intended functionality of final app			**10**
	Subtotal:		**30**
Adjustments	**All code meets specified standards:**		**– 3**
	Vague/Missing Documentation:		**– 2**
	Submission Requirements Not Followed:		**– 2**
	Late Penalties:		**25/50/75%**
	Total w/adjustments:		

Comments from Instructor:

CS 359 – PEX 4 – Grade Sheet Name:_____

Criteria		Points Earned	Points Available
App compiles and executes on an Android emulator with no errors (all or nothing score in this category … not compiling or crashing is bad)			**10**
All widgets (and screens, if applicable) are present and fully functional, including enabling/disabling of at least once widget			**10**
App is challenging, interesting, and creative and *majority of code is original* (i.e., not copied and pasted from an online example or tutorial)			**10**
App adheres to generally accepted object-oriented design principles and generally accepted Android design principles			**10**
Custom Component Enhanced Requirement			**10**
Optional Enhanced Requirement			**10**
	Subtotal:		**60**
Adjustments	**All code meets specified standards:**		**– 6**
	Vague/Missing Documentation:		**– 3**
	Submission Requirements Not Followed:		**– 3**
	Late Penalties:		**25/50/75%**
	Total w/adjustments:		

Comments from Instructor: