OMSCS 6340 - Fall 2016
Assignment 2 (100 points)
Due: 8:00 EDT, September 19, 2016

Objective: The goal of this assignment is to become familiar with random testing and have hands-on experience with testing tools used in industry. In particular, we will use Googles Monkey tool for testing Android apps, and use EMMA to measure the code coverage.

Resources:
1. Android tutorials: http://developer.android.com/training/index.html
2. About Monkey: http://developer.android.com/tools/help/monkey.html
3. EMMA homepage: http://emma.sourceforge.net

Setup: To start this assignment, you need to have the following software installed:

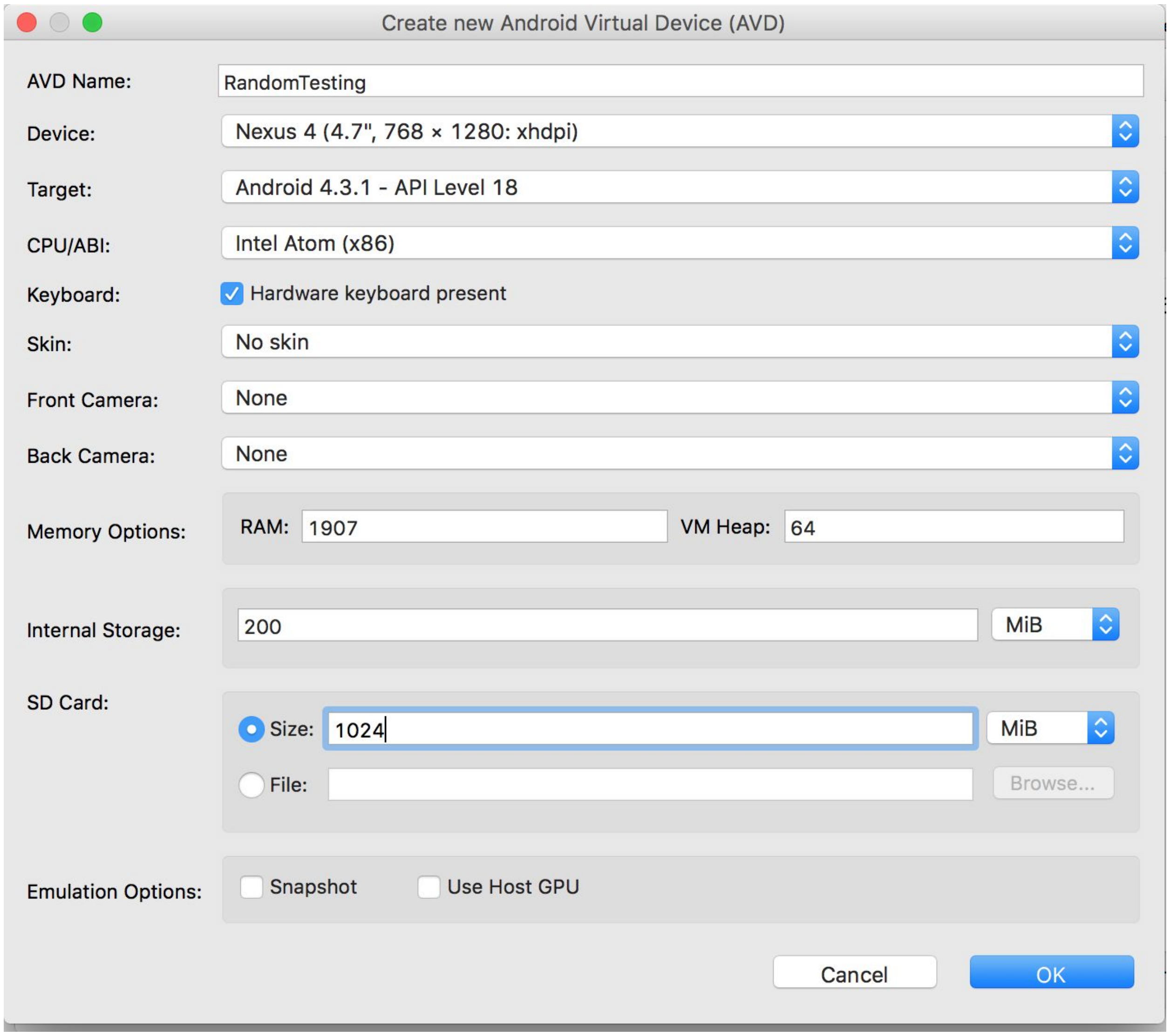
1. JDK 6 or later versions, which are available at http://www.oracle.com/technetwork/java/javase/   downloads/index.html
2. Ant, which is available at http://ant.apache.org. You can also install ant via package managers on
3. Linux or MacOS; you may use the provided virtual machine if you do not have access to your own linux or MacOS system
4. Android SDK tools, available at http://developer.android.com/sdk/installing/index.html?pkg=tools
5. EMMA: http://sourceforge.net/projects/emma/files/emma-release/2.0.5312/emma-2.0.5312.zip/download You can just unzip the file without running any installer.

We are going to use the RandomMusicPlayer app as the testing app. Its source file is provided as RandomMusicPlayer.zip. The app is already modified to enable EMMA's instrumentation. For the simplicity of elaboration, in the rest of the section, we assume the reader uses a UNIX-like environment.

First, we need to create a virtual Android device in an emulator. We are going to create an emulated Nexus 4 phone with Android SDK 4.3.1. Note, you must use the exact device with the exact SDK version; otherwise, very likely your result won't match our key, causing you to lose credits. To install SDK 4.3.1, open the SDK manager by running ⟨SDK DIR⟩/tools/android and choose the corresponding options, where ⟨SDK DIR⟩ is the folder where you installed the Android SDK tools. Although it is optional, we recommend installing Intel HAXM to speed up the emulator (for details, please refer to https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager).

Next, create a Nexus 4 device in the AVD manager by running
`⟨SDK DIR⟩/tools/android avd`

Configure the device as follows:



Create new Android Virtual Device (AVD)

AVD Name:	RandomTesting
Device:	Nexus 4 (4.7", 768 × 1280: xhdpi)
Target:	Android 4.3.1 - API Level 18
CPU/ABI:	Intel Atom (x86)
Keyboard:	✓ Hardware keyboard present
Skin:	No skin
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: 1907 VM Heap: 64
Internal Storage:	200 MiB
SD Card:	● Size: 1024 MiB; ○ File:
Emulation Options:	☐ Snapshot ☐ Use Host GPU

Note, the Intel Atom CPU/ABI option is only viable if you have intel HAXM installed; otherwise, you will have to use the slow ARM option. After creating the device, start the device in AVD manager.

Next, compile and install RandomMusicPlayer instrumented with EMMA. Unzip RandomMusicPlayer.zip, and run the following commands under the RandomMusicPlayer folder:

```
ant clean
ant emma debug
ant installd
```

Note, a file named coverage.em will be generated under RandomMusicPlayer/bin, which we will use later in generating EMMA reports.

Now, we can finally run Monkey on RandomMusicPlayer:

```
⟨SDK DIR⟩/platform-tools/adb shell monkey -p
com.example.android.musicplayer -s 11 -v 1000 > log.txt
```

The above command runs Monkey with 1000 random events and 11 as the seed for generating random numbers. It also redirects Monkeys output to the file log.txt. After Monkey finishes execution, we exit the app and dump the EMMA output to the emulated SD card by clicking BACK button at the bottom of the emulated device. Note, the EMMA output is appended to /sdcard/coverage.ec on the emulated device. Before starting a new Monkey run, you need to delete the file by running

```
adb shell rm /sdcard/coverage.ec
```

To generate the EMMA report, you need to pull the EMMA log file coverage.ec by running

```
adb pull /sdcard/coverage.ec ./
```

Then you can generate the EMMA report by running:

```
java -cp ⟨EMMA DIR⟩/lib/emma.jar emma report -sp
⟨RandomMusicPlayer⟩/src/ -r txt,html -in
⟨RandomMusicPlayer⟩/bin/coverage.em -in ./coverage.ec
```

You can view the EMMA report by opening `./coverage/index.html`

Items to Submit:

1. (50 points) The log.txt file generated by running `⟨SDK DIR⟩/platform-tools/adb shell monkey -p com.example.android.musicplayer -s 11 -v 1000 > log.txt` and the corresponding `coverage.txt` file generated by EMMA.

2. (50 points) Answers to the following problem set in a file named answers.txt. Submit the answers in the following format:

1 ⟨answer choice⟩
2 ⟨answer choice⟩

3 ⟨answer choice⟩
4 ⟨answer choice⟩
5 ⟨answer choice⟩

Problems: There is only one correct option for each question below.
Problem 1. [5 points] Monkey is a tool for __________ of Android apps.
A. White-box testing
B. Black-box testing
C. Checking full functional correctness D. Generating concise test cases
E. Generating non-redundant test cases F. All of the above

Problem 2. [5 points] What kind of code coverage does EMMA support?

A. Function coverage
B. Line coverage
C. Branch coverage
D. Path coverage
E. A and B
F. All of the above

Problem 3. [10 points] Run Monkey with 5000, 10000, 20000, 50000 events (Remember to run ”adb shell rm /sdcard/coverage.ec” before each run). You can conclude from the experiment:

A. With a small budget of events, random testing can achieve high coverage
B. With infinite resources, random testing can always achieve 100% line coverage for any app
C. When the budget of events is large, increasing the budget further does not increase the coverage significantly
D. Random testing is not suitable for stress testing

Problem 4. [10 points] Now, play with the app yourself (start by clicking the icon of the app). Compared to Monkey (to be specific, with 1000 events), can you achieve a higher coverage with fewer actions? Why or why not would this be the case, in general, for any given app?

A. Yes. As a programmer, I understand Android and the source code of the app very well
B. Yes. As a human, I understand the meaning of each UI widget
C. Yes. I operate the phone faster than Monkey
D. No. Monkey is designed by expert engineers from Google while I just started learning testing
E. E. A,B and C

Problem 5. [20 points] White-box testing can achieve higher coverage with fewer events compared to Black-box testing by inspecting the source code the program. For example, white-box testing techniques can find test cases that are equivalent. Read the source code of RandomMusicPlayer and tell which pair of test cases are NOT equivalent in terms of line coverage.

A. "Start the app − > Click Play button − > Click Pause Button" AND "Start the app − > Click Pause button − > Click Play Button"
B. "Start the app − > Click Play button − > Click Eject Button − > Click 'Cancel'" AND "Start the app − > Click Play button − > Click Eject Button − > Click 'Cancel' − > Click Play Button"
C. "Start the app − > Click Skip button − > Click Eject Button − > Click 'Play!' − > Click Stop Button" AND "Start the app − > Click Eject Button − > Click 'Play!' − > Click Stop Button − > Click Skip button"
D. "Start the app − > Click Eject Button − > Click 'Play!' − > Click Skip Button − > Click Stop Button" AND "Start the app − > Click Eject Button − > Click 'Play!' − > Click Stop Button − > Click Skip Button"
E. A and C only
F. All of the above