University of Waterloo

University of Waterloo
Midterm Examination Solutions

Term: **Winter** Year: **2013**

Student Name

UW Student ID Number

Course Abbreviation and Number	**ECE 155**
Course Title	**Engineering Design with Embedded Systems**
Section(s)	**001**
Sections Combined Course(s)	
Section Numbers of Combined Course(s)	
Instructor(s)	**Patrick Lam**

Date of Exam	**Friday, March 1, 2013**
Time Period	Start Time: **1:00 p.m.** End Time: **2:20 p.m.**
Duration of Exam	**80 minutes**
Number of Exam Pages (including this cover sheet)	**14**
Exam Type	**Closed Book**
Additional Materials Allowed	**NO ADDITIONAL MATERIALS ALLOWED**

Marking Scheme (For Examiner Use Only):

Question	Mark	Weight	Question	Mark	Weight
1		16	3a		5
			3b		5
2		10	3c		4
2a		2	3d		5
2b		3	3e		5
2c		5			
			4a		5
5		5	4b		10
Total					**80**

Instructions:

1. No aids permitted. No calculators of any type permitted.
2. Turn off all communication devices.
3. There are 5 questions and some of them have multiple parts.
4. The exam period lasts 80 minutes and there are 80 marks.
5. If you feel like you need to ask a question, know that the most likely answer is "Read the Question". If you find that a question requires clarification, proceed by clearly stating any reasonable assumptions necessary to complete the question. If your assumptions are reasonable, they will be taken into account during grading.
6. After reading and understanding the instructions, sign your name in the space provided below for your signature.

Signature

Question 1. Multiple Choice Questions [16 marks total]

For each of the following multiple choice questions, place the **letter** of the most appropriate solution on the answer line provided.

i. Which part of an embedded system enables the CPU to directly affect the outside world?

(a) sensors
(b) memory
(c) actuators
(d) keyboard

ANSWER: (c)

ii. Which of the following is not a type of polling?

(a) interrupts
(b) tight
(c) occasional
(d) periodic

ANSWER: (a)

iii. Which of the following JUnit assertions pass?

(a) assertEquals("hello", "helo");
(b) assertFalse("Case 1", 5 > 10);
(c) assertTrue("4 > 1", 21*1 > 21);
(d) All will fail
(e) (a) and (b) will pass

ANSWER: (b

iv. Which types of tests detect previously-fixed bugs?

(a) regression tests
(b) integration tests
(c) unit tests
(d) midterm examinations

ANSWER: (a

v. What is NOT true about polling?

(a) Polling can be considered to be passive synchronization.
(b) Tight polling constantly checks the device reading.
(c) Occasional polling checks the device reading whenever convenient.
(d) A polled device imposes its own schedule on the processor.

ANSWER: (d)

vi. Which of the following is **NOT** a characteristic of an embedded system?

(a) May or may not contain an operating system.
(b) System performs tasks without its existence being obvious to the user.
(c) Must contain a processor.
(d) Usually, timing is critical in the design.

ANSWER: (c)

vii. Which class do you use to tell Android to start a new Activity?

(a) TimerTask
(b) Handler
(c) BroadcastReceiver
(d) Intent

ANSWER: (d)

viii. In an Android JUnit test, which method do you need to call to ensure that your actions have occured?

(a) `updates()`
(b) `waitForIdleSync()`
(c) `synchronize()`
(d) `suspend()`

ANSWER: (d)

ix. Which one of the following sets of activities best describes the design loop in the engineering design process?

(a) Problem definition, brainstorming, and refactoring
(b) Synthesis, analysis, and decision making
(c) Synthesis, analysis, and optimization
(d) Design, construction, and testing
(e) Design, analysis, and testing

ANSWER: (b)

x. Which of the following can implement asynchronous execution?

(a) static methods;
(b) watchdogs;
(c) callbacks; or
(d) operator overloading.

ANSWER: (c)

xi. Pick the term (in the ECE155 sense) that constrains your design most, out of these options:

(a) specification
(b) heuristic
(c) guideline
(d) standard

ANSWER: (a)

xii. Which of the following examples is the best analogy for polling as seen in ECE155?

(a) Voting in an election at a polling station.
(b) Obsessively checking your emails to see if you won the lottery.
(c) Sitting in a coffee shop reading the paper, waiting for your phone to ring.
(d) Paying your bills when they arrive.

ANSWER: (b)

xiii. The function of an ADC is to:

(a) Convert a discrete-time digital signal into a continuous-time analog signal.
(b) Convert a continuous-time analog signal into a discrete-time digital signal.
(c) Convert a discrete-time digital signal into a regular signal.
(d) Convert a discrete-time digital signal into an analog-time analog signal.

ANSWER: (b)

xiv. Which of the following is a typical combination of the key components found in an IDE?

(a) Compiler, Graphics Editor, Code Editor
(b) Android SDK, Emulators, Code Editor
(c) SVN, Compiler, Code Editor
(d) Compiler, Code Editor, Debugger

ANSWER: (d

xv. What will generally happen when a watchdog timer is fired?

(a) The timer resets its interval to 0.
(b) The program initiates a break point at the line of code where watchdog timer was fired.
(c) The system asks the user if she or he wants to terminate the offending process.
(d) The program executes a watchdog handler, which usually terminates a long-running task.

ANSWER: (d)

xvi. If an embedded system with large number of activities must respond rapidly to an external event E, which one of the following synchronization mechanisms should be used to notify the program of an occurrence of E?

(a) Periodic polling
(b) Occasional polling
(c) Tight polling (busy-waiting)
(d) Interrupt handling

ANSWER: (d)

Question 2. Terminology [20 marks total]

In this course, you have learned a number of terms related to embedded systems, software systems, and engineering design. Complete each row of Table 1 shown below by writing the term that best corresponds with the definition provided.

Table 1: Terminology

Term	Definition
sensor	A device that converts input from the external environment into a form suitable for a computer system.
interrupt	A mechanism by which a device informs a processor that something has occurred.
unit test	A low-level test which verifies the functionality of a single class at a time.
broadcast receiver	Gets notified of changes to the state of your system.
design review	An independent evaluation of a design alternative.
real-time system	A system in which the time to respond to an external event must be bounded by a known, finite amount of time.
design criteria	Successful solutions to a design problem should (but aren't required to) satisfy these.
low-pass filter	Removes high-frequency noise from its input (used in your labs).
gather	First step in software bricolage (where you put together a system based on others' work).
event handler/listener	A method executed in response to the generation of an Android event.

Part A) Version Control [2 marks]

Alice and Bob both check out a working copy of file `MainActivity.java` from their version control repository. Alice commits a change to line 42 of the file. Bob also makes a change to the file at line 42. Now, what does Bob need to do before he can commit a change to `MainActivity.java`?

Bob first needs to update his working copy. He must then merge the resulting conflicts.

Part B) Simulation [3 marks]

(1 point) List two reasons to use a simulation of a system instead of working on the real thing.

may not exist; expensive to use; dangerous to use; may be disrupted; not enough access

(2 points) Which would you expect to be more accurate near-term weather-forecasting simulation: a stochastic simulation with 32 bits of accuracy, run 10000 times; or a deterministic simulation with 80 bits of accuracy, run once? Why?

Weather is sensitive to initial conditions, so running a lot of times is more important than having a lot of precision.

Part C) Inheritance and Subclassing [5 marks]

Consider the following code:

```
class A {
  void foo() { Log.d("W", "hi"); }
}

class B extends A {
  void foo() { Log.d("W", "iPhone"); }

  void bar() { Log.d("W", "Android"); }
}

class Main {
  public static void main(String[] argv) {
    A a = new A(); // (1)
    B b = new B(); // (2)
    A c = new B(); // (3)
    B d = new A(); // (4)

    b.bar(); // (A)
    a.foo(); // (B)
    c.foo(); // (C)
    d.bar(); // (D)
  }
}
```

(1 point) One of lines (1) through (4) won't compile. Which one, and why?

Line (4) won't compile because an A object is not a B.

(4 points) Write down the output of lines (A) through (D), or "none" for the invalid variable.

A: Android
B: hi
C: iPhone
D: none

Question 3. Android Programming: Timers and XML [24 marks total]

Part A) Problems with Java Timers [5 marks]

This code implements a `Timer` that runs for 10 seconds. The payload is to set the text of an `EditText` to a certain value. However, the code will throw an exception when it is run.

```
protected void onCreate()
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Timer t = new Timer();
    EditText textBox = (EditText)findViewById(R.id.textBox);

    t.schedule(new TimerTask {
        @Override
        public void run() {
            textBox.setText("This is a midterm question!");
        }
    }, 10000);
}
```

(1 point) Why will the code throw an exception?

Android does not permit other threads to cause changes to the UI; the `TimerTask` *runs in a separate thread from the UI thread.*

(4 points) Without using a `Handler`, fix the code so that it works properly. (If you can't remember the syntax but can describe, in precise words, what you need to do, you may get 1.5/4 for this part. If you provide code that uses the wrong names but is otherwise correct, you can get full marks, as long as I can tell what names you meant.)

You may assume that the `EditText` was previously defined in the layout.

```
protected void onCreate()
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Timer t = new Timer();
    EditText textBox = (EditText)findViewById(R.id.textBox);

    Runnable runn = new Runnable() {
        @Override
        public void run() {
            textBox.setText("This is a midterm question!");
        }
    };

    t.schedule(new TimerTask {
        @Override
        public void run() {
            runOnUiThread(runn);
        }
    }, 10000);
```

```
}
```

Part B) Using a Handler instead [5 marks]

(4 points) Now implement the above code using a `Handler` instead of a Java timer.

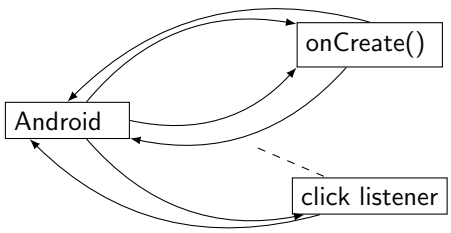
```
protected void onCreate()
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Handler hand = new Handler();
    EditText textBox = (EditText)findViewById(R.id.textBox);

    Runnable runn = new Runnable() {
        @Override
        public void run() {
            textBox.setText("This is a midterm question!");
        }
    };

    hand.postDelayed(runn, 10000);
}
```

(1 point) Which of the two implementations would you expect to fire closer to exactly 10 seconds after it was scheduled?

The `TimerTask` *will fire at close to exactly 10 seconds while the call to the* `Handler` *might be delayed. However, the two implementations will be approximately equivalent in terms of setting the text, since they both have to wait for the UI thread to become available.*

Part C) Event-Driven Programming [4 marks]

Recall this diagram from lecture.



Write down (in text) the relevant actions that need to occur to handle a button click. I'm looking for the actions that your program takes, along with what Android does at certain times. (Say who is doing what.)

0. Android calls onCreate;
1. you register a click event listener in your `onCreate()` *method;*
2. user clicks on the button;

3. Android calls your click listener.

Part D) XML [5 marks]

This XML file is not well-formed: it has 3 errors. Circle two of the XML errors and explain why they are invalid XML.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="ca.uwaterloo.Lab1_plam"
4      android:versionName="1.0" >
5
6      <uses-sdk
7          android:targetSdkVersion="16" >
8
9      <application
10         android:allowBackup="true"
11         <action android:name="ca.patricklam.MAIN" />
12         android:theme="@style/AppTheme" >
13         <activity
14             android:name="ca.uwaterloo.Lab1_plam.MainActivity"
15             android:label="@string/app_name" >
16             <intent-filter>
17                 <action android:name="android.intent.action.MAIN" />
18                 <category android:name="android.intent.category.LAUNCHER" />
19             </activity>
20         </intent-filter>
21     </application>
22 </manifest>
```

Part E) Launching Activities and Getting Responses [5 marks]

(3 points) List (in text) the steps that you need to take to get Android to launch another `Activity`. How does Android know which activity to launch? (2 points) Describe how to get a response back from an `Activity` that you've launched, including anything special that you may have to do while launching the activity.

Use an Intent to start another activity; describe the parameters for the target activity by setting the action, data, etc; call the `startActivity()` *method to initiate intent resolution. To get a response back from the other activity, use the* `onActivityResult()` *callback, matching the request code with the one you provided when starting the activity with* `startActivityForResult()`*. (That activity should* `setResult()` *to pass the result back to you.)*

Question 4. Unit Testing with JUnit [15 marks total]

Part A) Fixing Code Based on a Unit Test [5 marks]

Consider the Finite State Machine implementation and JUnit test provided below.

```
public class FSM {
 private int state = 0;

 public boolean isAccepting() {
    return state == 4;
 }

 public void transition (int input) {
  switch (state) {
  case 0:
    if (input == 1) {
      state = 1;
    }
    break;
  case 1:
    if (input == 3) {
      state = 2;
    } else {
      state = 0;
    }
    break;
  case 2:
    if (input == 3) {
      state = 3;
    } else {
      state = 0;
    }
    break;
  case 3:
    if (input == 7) {
      state = 4;
    } else {
      state = 0;
    }
    break;
  }
 }
}
```

```
import static org.junit.Assert.*;
import org.junit.Test;

public class FSMTest {
  @Test
  public void reachesFinalState() {
    FSM f = new FSM();
    f.transition(1);
    f.transition(3);
    f.transition(7);
    assertTrue(f.isAccepting());
  }

  @Test
  public void initialIsNotFinal() {
    FSM f = new FSM();
    assertFalse(f.isAccepting());
  }
}
```

Test case `reachesFinalState()` currently fails. Assume that both test cases are supposed to pass. Show a set of changes to the `FSM` class which makes both cases pass.

Many possible; probably the easiest is to change `input == 3` *to* `input == 6` *on case 2 and the check in* `isAccepting()` *to* `state == 3`.

Part B) Writing Unit Tests [10 marks]

(5 points) Write a new unit test method which verifies that the sequence of inputs "1 3 3 4" does *not* put the finite state machine into an accepting state.

```
    @Test
        public void doesNotReachFinalState() {
        FSM f = new FSM();
        f.transition(1);
        f.transition(3);
        f.transition(3);
        f.transition(4);
        assertFalse(f.isAccepting());
    }
}
```

(5 points) Write another new unit test method which verifies that the sequence of inputs "1 3 3 7" is accepted by the initial `FSM` class and that no prefix of "1 3 3 7" is accepted by that class (e.g. "1"; "1 3"; "1 3 3").

```
    @Test
        public void doesNotReachFinalState() {
        FSM f = new FSM();
        f.transition(1);
        f.transition(3);
        f.transition(3);
        f.transition(4);
        assertFalse(f.isAccepting());
    }
}
```

Question 5. Inner Classes [5 marks total]

Recall this code from an earlier question.

```
public class UsesInner {
  protected void onCreate()
  {
    // ... removed ...
    Timer t = new Timer();
    EditText textBox = (EditText)findViewById(R.id.textBox);

    t.schedule(new TimerTask {
        public void run() {
            textBox.setText("This is a midterm question!");
        }
    }, 10000);
  }
}
```

This code uses an inner class to define the `TimerTask` object. Rewrite it without using an inner class; that is, create a class `MyTimerTask` and an instance of that class which you pass to `t.schedule()`. Your code should behave just like the original code.

[80 marks grand total]