Computer Science 205
Review Sheet for Exam #2
Friday, October 28th

Exam #2 will cover all class notes, handouts, and labs completed since the last exam. The labs that will be covered are Labs #5 – #9.

The main topics to be covered are : inheritance; polymorphism (dynamic binding); GUIs; interfaces and listeners; recursion.

Listed below are the important terms and concepts covered on the exam.

Object-Oriented Programming Relationships

- Be able to name and define the three types of relationships that can exist between objects. How do you add inheritance or containment to a class? How do you know whether to use inheritance and whether to use containment?
- Definition of base class and derived class. Be able to draw an inheritance hierarchy diagram. What are the properties of the `Object` superclass, and what methods are defined in it?
- Definition of overriding. Be able to set up the constructor for a derived class and use the `super` keyword.
- Differences between `private`, `protected`, `public`, and no access modifiers. Be comfortable with questions on worksheet.
- Definition of polymorphism. When does it occur? When does a `ClassCastException` occur? What types of polymorphism errors are caught by the compiler? Be able to identify a run-time error and a compiler error involving polymorphism.
- Differences between abstract and final classes. How are both set up?
- Role of `instanceof` operator
- A class named `Throwable` is the superclass for all exception classes, and is where the `getMessage()` method is defined.

Graphical User Interfaces

- What are interfaces and where are they used?
- Definition of listener objects.
- Role of `add` method? What class is it defined in?
- Be able to set up a listener for a single component or multiple components. If we connect two or more buttons to a single listener object, name two ways we can determine which button is involved.
- Role of an inner class? Can an object of an inner class type be instantiated?
- Be able to name the most common layout manager classes. What is the default layout manager for a window? Be able to write a call to `setLayout` that specifies a particular layout.
- What is a panel? What is its default layout manager?

Recursion

- What is recursion and what distinguishes it from iteration?
- Advantages and disadvantages of recursion compared to iteration
- Determining the base case and recursive case of a method
- Tracing through a recursive method (You should show all method calls in a recursive trace on the exam.) The recursive method you will be tracing through on the exam will be similar, but different, to the one in the lab. So, be sure you are comfortable with the integer division operators of / and %.
- Writing a method both iteratively and recursively to find a factorial or a summation. The base case of a summation will always be a value of n less than the start value which always returns a value of 0.
- Be able to give an example of infinite recursion.
- Writing recursive functions using arrays. See the palindrome question and binary search question on your worksheets. What is the `Comparable` interface?
- Tail recursion and infinite recursion.
- What is backtracking?

Other

- Steps for writing objects out to an external file or reading objects in from a external file
- What is a binary file? What Java file classes support working with them?
- Role of the `Serializable` interface
- Using a `BitSet`. Why do we use them? Be able to write the code to print them out, find inverse, or count the number of "on" bits
- Using the `Scanner` class on a `String` object; What is the default delimiter? How do you reset the default delimiter?

Computer Science 205 Exam #2

Answer Sheet

Fall 2016

Friday, October 28th

100 points

Name : Yu-Ching

73/100

1. (a) T1
T2
T3 ✓

(b) protected int b = 3, int c = 4 ✓

-1 (c) No because class T2 and T3 can't be accessed so won't print

-1 (d) Yes it would compile and run, it would print out ~~T1, T2,~~ T3 again

address

2. (a)

Fruity
↑
Fruit
↑
Berry
↑
Strawberry ✓

(b) ~~I am a Fruit~~
~~I am a Fruit~~

I'm a Fruit
I'm a Fruit
I'm a Strawberry
I'm a Berry ✓

(c) 2 times ✓

3. (a) public abstract class Shape

(b) ~~this extends~~ public class Triangle extends Shape, Window Adapter

(c) public void windowClosing(WindowEvent evt)

(d) public abstract int getHeight()

-2 (e) public abstract ing getWidth()

import.java statement

-4

4. **Circle.**

(a) [✓ True] False

(b) [✓ True] False

(c) [✓ True] False

(d) [✓ True] False

(e) True [✓ False]

(f) True [✓ False]

(g) [✓ True] False

(h) [✓ True] False

(i) True [✓ False]

(j) True [✓ False]

-4 (k) [✓ True] False

(l) [✓ True] False

(m) [✓ True] False

(n) [✓ True] False

(o) [✓ True] False

5. int n = 23;

puzzle(23/3)+1 = 8

puzzle(8/3)+1 = 3

puzzle(3/3)+1 = 1 ✓

base case

puzzle(23) = 8+3+1

= 12

-4

6. private static int mySum (int n) {

```
~~if (n = = 1)~~                int n;
~~    return 1;~~               int k;
~~else~~                        if (n = = 1)
~~    return (~~                    return 1;
                             else
                             ~~n = k~~ mySum(n-1)
                             return(~~k~~ x ((k x k x k) + 1));
```

no recursion?

-4

7. (a) return -1 when ?

-2

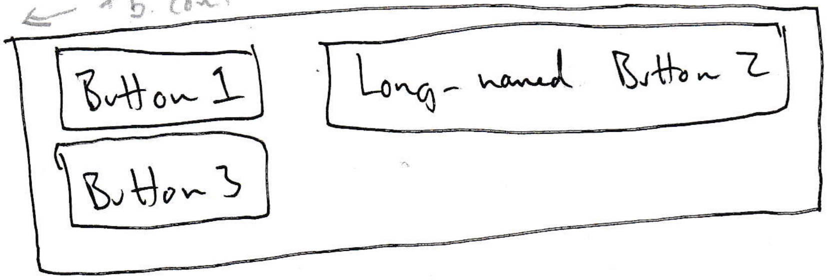
(b) mystery (-3, 1, 0) = mystery (-4, 2, 1)
mystery (-4, 2, 1) = mystery (-2, 3, 2)
mystery (-2, 3, 2) = mystery (-5, 4, 3)
mystery (-5, 4, 3) = mystery (-4, 5, 4)

-4

6

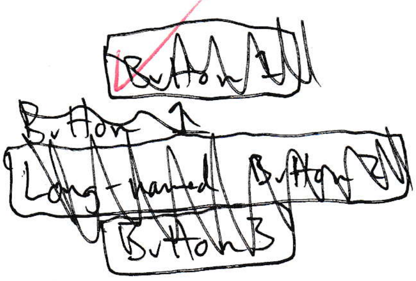
← 7b. cont

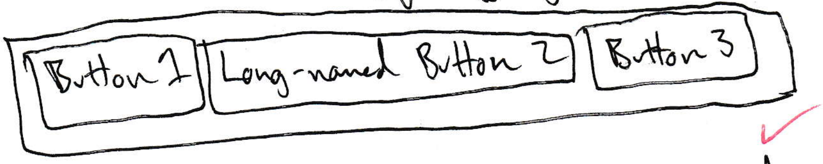
8. (a) i.



✓

ii. ~~BoxLayout~~ ~~GridBagLayout~~ ~~TextField~~





✓ ✓

(b) i. ~~BoxLayout~~ TextField, Button, ~~Listener, Set Text()~~ window, panel

-2

ii. ~~GridBagLayout~~ BoxLayout, GridBag Layout ✓

-12

7b cont.

mystery(-4, 5, 4) = mystery(-2, 6, 5)

mystery(-2, 6, 5) = mystery(-3, 7, 6)

mystery(-3, 7, 6) = mystery(-3, 8, 7)

mystery(-3, 8, 7) = mystery(-2, 9, 8)

mystery(-2, 9, 8) = mystery(-2, 10, 9)

mystery(-2, 10, 9) = 1

boolean buttonPressed = false;

(c)
```
class ArrowListener implements ActionListener {
  public void actionPerformed(ActionEvent evt) {
    ~~buttonPressed = true;~~     ×

    if (buttonPressed = true    ) ×

      arrowIndex = (arrowIndex + pastNumberCount - 1) % pastNumberCount;

    else

      arrowIndex = (arrowIndex + 1) % pastNumberCount;

      phoneNumber.setText( ~~past Arrays.toString~~ (pastNumbers));
                                                       [arrowIndex

  }
}
```

-7

-7