Exam 1 Review
part 2

COMP163

“Seeing much, suffering much, and studying much, are the three pillars of learning.”

Benjamin Disraeli

First Exam

- The first exam in COMP163 will be in lecture on Monday, September 23
- The exam will cover everything since the beginning of the semester
- You are allowed one 8½ by 11 inch page of notes
- Most questions will be of the form
  - complete this program
  - what does this program display

Technical Interview Prep Sessions

- Tuesday, September 24 at 4:00-5:30PM in McNair 132
- Target audience is first semester freshman CS majors who are taking COMP163
- Will give hands-on practice at the kind of technical problem solving and coding questions that are given during Google interviews
- Demystify the interview process
- Share what I and other Google interviewers have looked for over the last decade
- Led by Dr. Dave Foulser of 

Exam Topics

- All of the material in Chapters 1 through 4 of the online textbook

- The exam may contain questions from any of the material covered in class since the beginning of the class

Topics not covered in class
will not be on the exam

One Page of Notes

- You are allowed one and only one 8½ by 11 inch page of notes during this exam
- You are not allowed to use more than 187 square inches of paper surface
- You will do better if you make your own page of notes and not copy your friend's notes

Exam Format

- The exam will be on paper during lecture on Monday
- The exam will be similar to the labs, quizzes and homework
- Exams tend to be programming oriented
- The exam will be similar to the practice exam available on Blackboard
- Try the sample exam before looking at the answers

Declaring an Object

- Imagine we have a class **`Widget`**
- We can declare an object of type **`Widget`** just like we declare a variable to be an int or a double

```
Widget  lizard;
int     moose;
Widget  newt, salamander;
```

- lizard, newt and salamander are objects of the type Widget

Reference Variables

- When you declare a primitive data item, such as a double or int, Java reserves some memory to store the data
- When you declare an object, Java does **not** reserve space for the object until it is created
- You can instantiate a new object with the keyword **new**

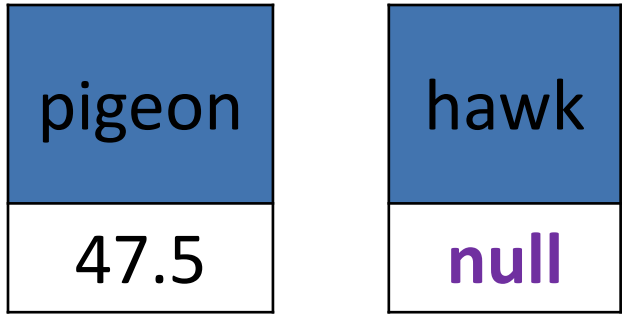
Instantiating Objects

```
Widget tiger = new Widget();
Widget elephant;
elephant = new Widget();
```

- After the word "**new**" is a call to a constructor method that creates the object
- The constructor method may or may not have parameters

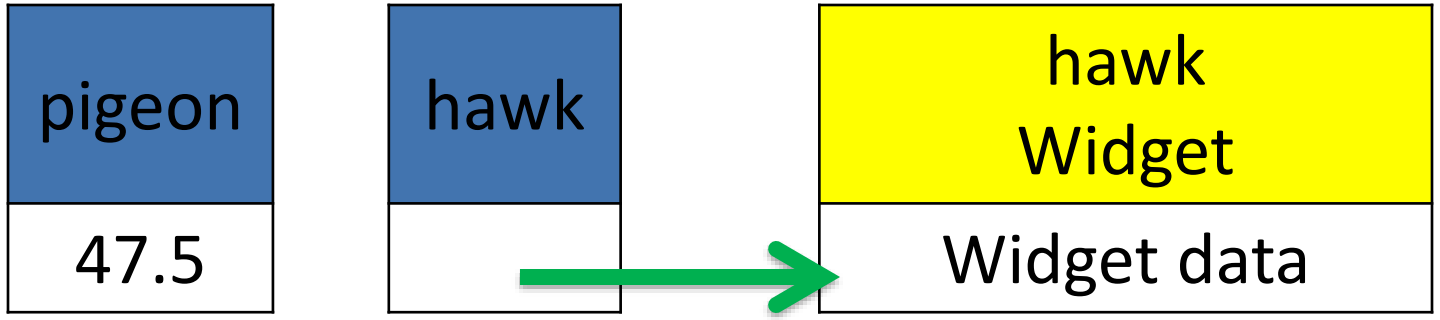
Reference Variables

```
double  pigeon = 47.5;
Widget  hawk;
```



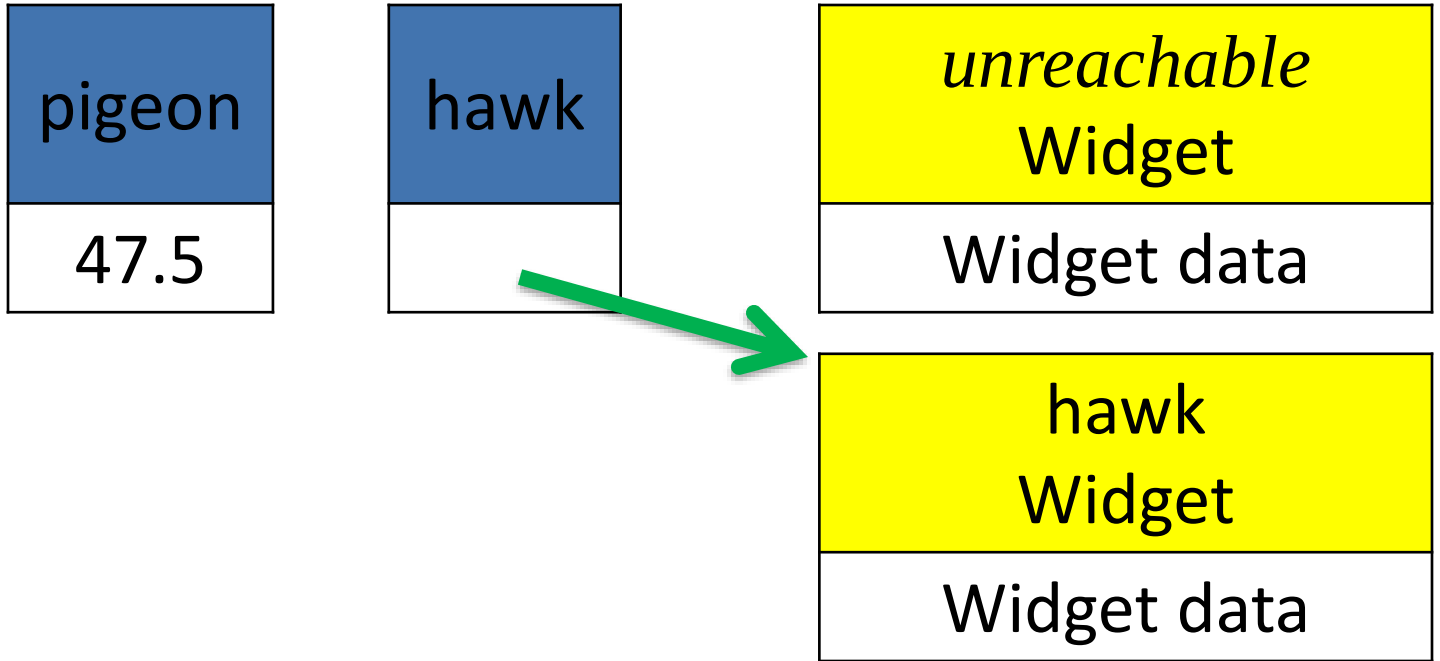
Reference Variables

```
double  pigeon = 47.5;
Widget  hawk;
hawk = new Widget();
```



Reference Variables

```
double  pigeon = 47.5;
Widget  hawk;
hawk = new Widget();
          . . .
hawk = new Widget();// later in program
```



Making a GUI

1. Create a class that extends JFrame and implements java.awt.event.ActionListener
2. Create component objects
3. Specify the layout manager
4. Add the component objects to the container
5. Add an action listener to the components that respond to user input
6. All the program to be visible and stop
7. Create an actionPerformed method

Which creates an object of the class Bird?

A. Bird falcon;

B. falcon = Bird;

C. falcon = Bird();

D. falcon = new Bird();

E. Bird(falcon);

Usual Java GUI

```
public class MyProgram extends javax.swing.JFrame
    implements java.awt.event.ActionListener {
        // declare GUI components here
  public MyProgram() {
        // setup GUI here
  }
  public void actionPerformed(
                java.awt.event.ActionEvent  dog){
        // program here
  }
  public static void main(String[] cat) {
      MyProgram fish = new MyProgram();
  }
}
```

Usual Java GUI in any Order

```
public class MyProgram extends javax.swing.JFrame
    implements java.awt.event.ActionListener {
        // declare GUI components here
 public static void main(String[] cat) {
        MyProgram fish = new MyProgram();
  }
  public MyProgram() {
        // setup GUI here
  }
  public void actionPerformed(
                java.awt.event.ActionEvent  dog){
        // program here
  }
}
```

Creating Components

- GUI components, such as JLabels, JTextFields and JButtons, should be defined in the class but outside of any method

```
javax.swing.JButton squirrel =
     new javax.swing.JButton( "OK" );
javax.swing.JLabel chipmunk =
     new javax.swing.JLabel("Hi there");
```

Specifying the Layout Manager

- You must specify what layout manager your program will use with the **`setLayout`** method
- The parameter of the **`setLayout`** method is any layout manager object

```
java.awt.Container  goat = getContentPane();
javax.swing.BoxLayout sloth =  new
        javax.swing.BoxLayout(goat,javax.swing.BoxLayout.Y_AXIS);


setLayout( sloth );
```

Adding Components to a GUI

- To make them appear in a frame the components must be added to the frame
- The add method links a component to the content pane (variable goat)

```
goat.add( GUIthing1 );
goat.add( GUIthing2 );
```

- The order in which the components are added determines the order in which they will appear in the GUI

Making It Start and Stop

- To make your program terminate if you press the red **X** in the upper right corner, include

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- Call `setVisible(true)` to make the frame appear onscreen

Adding a Listener

- The method addActionListener should be called on each GUI component that can do something

```
javax.swing.JButton squirrel = new
                        javax.swing.JButton("Go");
        …
squirrel.addActionListener( this );
```

Inactive Components

- A button will not do anything unless you specify an actionListener
- JLabels do not need an actionListener
- A JTextField may have an actionListener
  - If you press enter when the cursor is in a JTextField, the actionPerformed method will be called
  - You may make some, but not all, react to input

Responding to a Click

- When a user clicks an object with an ActionListener, Java calls the method

```
public void actionPerformed(java.awt.event.ActionEvent   bear) {
        // do something here
 }
```

- This method should do what the user expects to happen when the button is pressed

You should add an actionListener to

A. All GUI components
B. All JLabels
C. All JButtons
D. All JFrames
E. All Strings

Inputting Numbers in a GUI

- The getText() method returns a String
- If a number is entered in a JTextField, it will have to be converted to an int or double

```
JTextField gekko = new JTextField();
    // in the actionPerformed method
String dragon = gekko.getText();
double lizard = Double.parseDouble( dragon );
```

Displaying Results in an Application

- Display the results in a Java application in an easy to understand format

```
System.out.println("answer is  " + aNumber);
```

GUI Output

- When a program with a GUI wants to display a result, it usually puts the value in a JLabel object

```
JLabel aardvark = new JLabel();
                    // in actionPerformed
double anteater = some equation;
aardvark.setText("The answer is " + anteater);
```

```
JTextField bird = new JTextField();
JTextField cow = new JTextField();
        // Try A
String cat = bird.getText();
double dog = Double.parseDouble(cat);
cat = cow.getText();
double goat = Double.parseDouble(cat);
        // Try B
double dog = Double.parseDouble(cat);
double goat = Double.parseDouble(bull);
String cat = bird.getText();
String bull = cow.getText();
```

Which is Correct?

A. A
B. B
C. both
D. neither

A Comment on Comments

- It is very useful *(and required)* to comment the purpose of all variables
- Comments should explain the purpose of the variable in the program
- Do not restate what is written in Java

```
double trunk;     // volume of trunk in cubic feet
double trunk;     // declare trunk to be a double
```

Work Together to complete GUI to twice the input number

```
import javax.swing.*;
public class SqrtGUI extends JFrame implements
                        java.awt.event.ActionListener {
    JLabel     enterMsg = new JLabel("Enter a number");
    JTextField inStuff = new JTextField();
    JButton      button = new JButton("double");
    JLabel       result = new JLabel();
public SqrtGUI() {
   setSize( 100, 100);
   java.awt.Container pane = getContentPane();
   javax.swing.BoxLayout where = new BoxLayout(pane, BoxLayout.Y_AXIS);
   setLayout( where );
   pane.add(enterMsg);
   pane.add(inStuff);
   pane.add(button);
   pane.add(result);
   button.addActionListener( this );
   inStuff.addActionListener( this );
   setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
   setVisible( true );
}
public void actionPerformed(java.awt.event.ActionEvent thing){
```

/* Write this part */

Possible Solution

```
String    inText     = inStuff.getText();
double  number = Double.parseDouble( inText );
double  answer  = 2.0 * number;
result.setText("twice is " + answer );
```

Another Possible Solution

```
result.setText("twice is " +
        2.0*Double.parseDouble(inStuff.getText()));
```

Trimming the Input

- The parseInt and parseDouble methods do not always like spaces in the string
- It can be helpful to call the String trim() method to eliminate leading and trailing blanks

```
String inText = labelThing.getText();
inText = inText.trim();
```

or

```
String inText = labelThing.getText().trim();
```

```
JTextField bird = new JTextField();
JTextField cow = new JTextField();
        // Try A
String cat = bird.getText();
double dog = Double.parseDouble(cat);
cat = cow.getText();
double goat = Double.parseDouble(cat);
        // Try B
double dog = Double.parseDouble(cat);
double goat = Double.parseDouble(bull);
String cat = bird.getText();
String bull = cow.getText();
```

Which is Correct?

A. A

B. B

C. both

D. neither

Graphics Methods

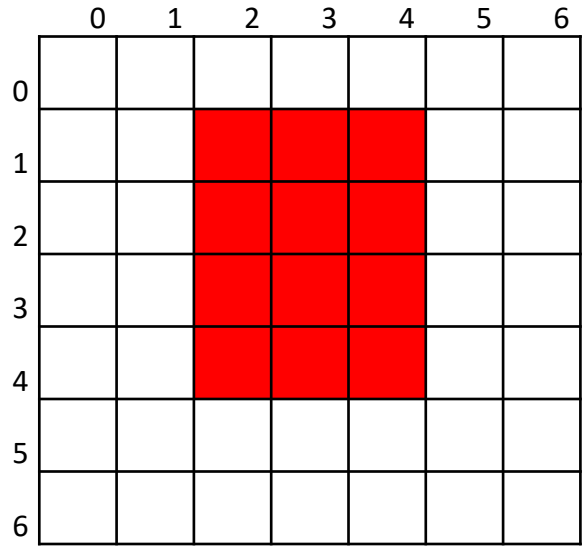
```
bird.drawRect( x, y, width, height);
bird.fillRect( x, y, width, height);
bird.drawOval( x, y, width, height );
bird.fillOval( x, y, width, height );
bird.drawLine( x1, y1, x2, y2 );
bird.drawString( String, x, y)
```

- where **`bird`** is the parameter to paint, an object of the type java.awt.Graphics

Work with the students next to you

Complete this method to draw a small **red** rectangle as shown in the diagram. Each box represents a pixel

```
public void paint( java.awt.Graphics  bird ) {
```

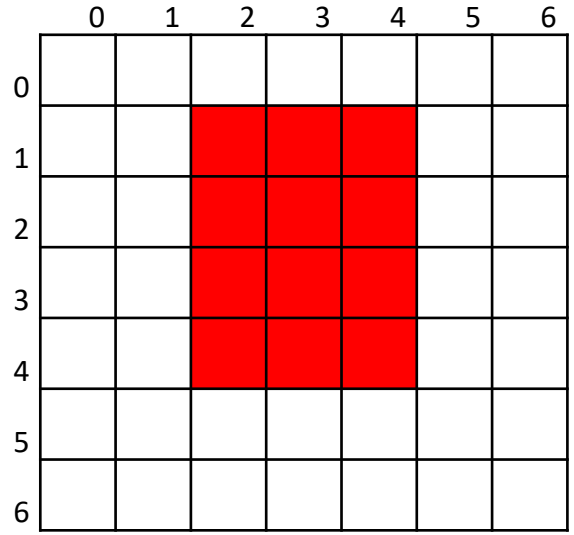


Possible Solution

Complete this method to draw a small **red** rectangle as shown in the diagram. Each box represents a pixel

```
public void paint( java.awt.Graphics  bird ) {


bird.setColor(java.awt.Color.RED);
bird.fillRect( 2, 1, 3, 4);
```



Coloring Your GUI

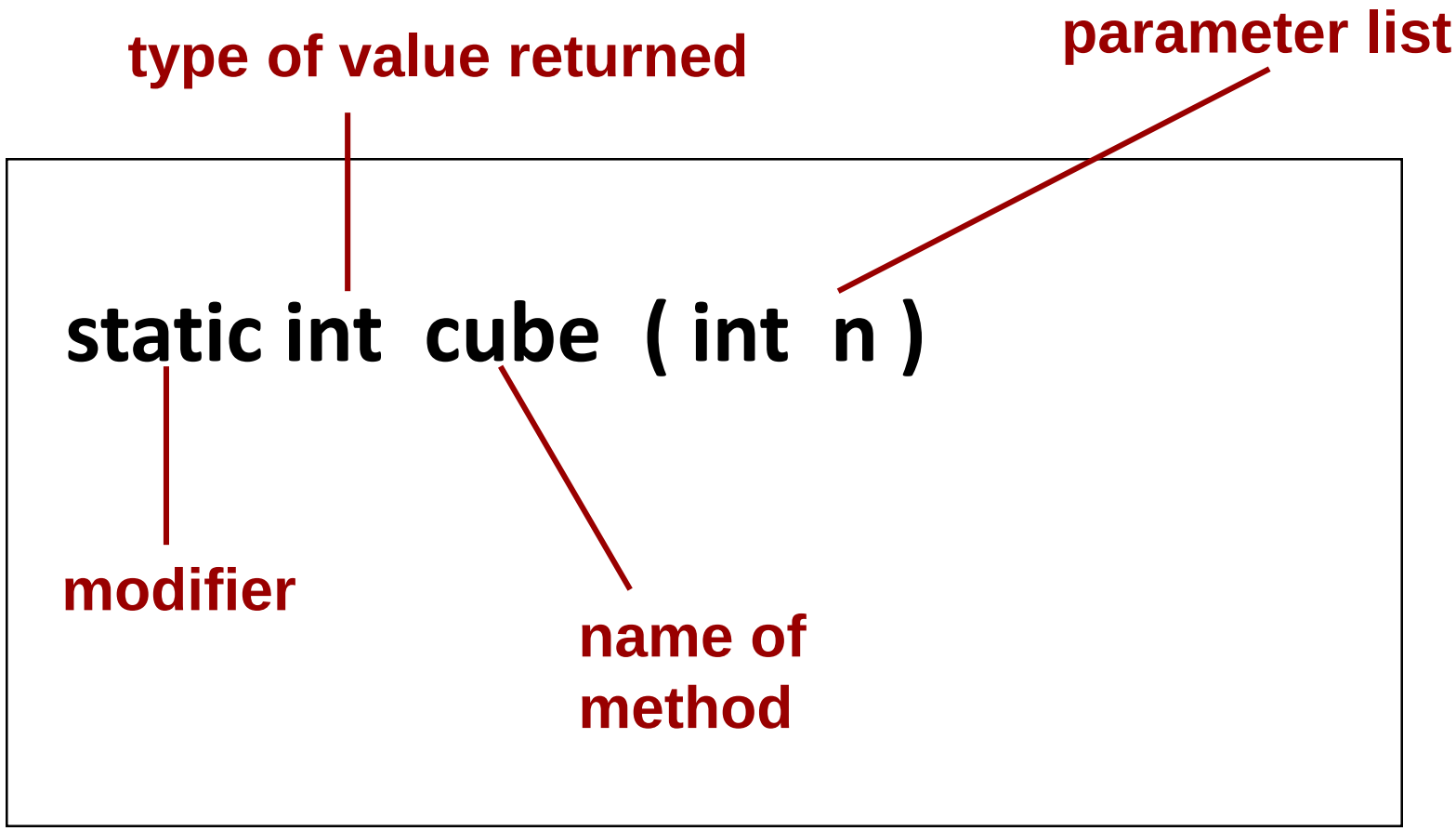
- You can change the color of your GUI components
- The setForeground(java.awt.Color) method sets the text color

```
javax.swing.JLabel cat = new javax.swing.JLabel("GEEN163");
cat.setForeground( java.awt.Color.RED );
```

You can also set the color of the background of the component

```
cat.setBackground( java.awt.Color.BLUE );
```

What is in a heading?



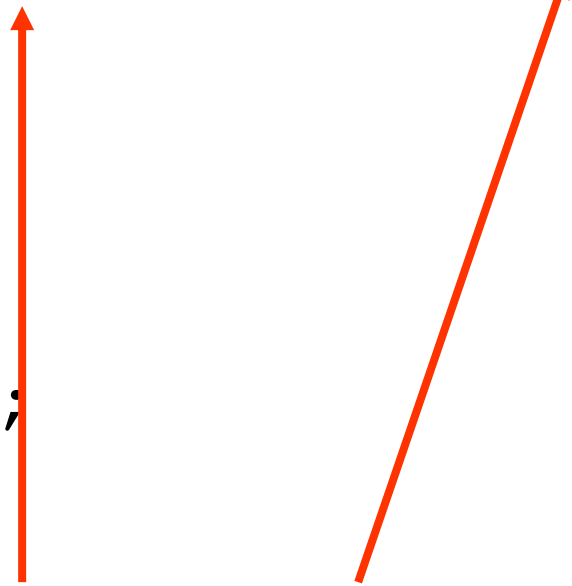
Parameters Must Match Type

- Method

```
double myfunc( int cat, String dog){
 … }
```

- Calling program

```
int  cow = 7;
String bull = "what";
double goat;
     goat = myfunc( cow,    bull );
```



Parameter Passing

- When a main program calls a method, the values of the arguments in the calling statement are copied to the parameters of the method.

```
int goat = 7, cow = 13;

myProg( goat, cow);


void myProg( int sheep, int bull) {
                /* sheep is 7,  bull is 13 */
```

What is displayed by this program?

```
int deer = 2, bear = 7, panther = 4;
panther = racoon( deer, bear);
System.out.print("  panther="+ panther);
        . . .
int racoon( int goat, int pig) {
  System.out.print("goat="+goat);
  return goat + pig;
}
```

A. goat=2 panther = 4
B. goat=2 panther = 9
C. panther = 9 goat = 2
D. panther = 4 goat = 2
E. none of the above

Interesting String Methods

- length
- indexOf
- substring
- charAt
- toUpperCase
- toLowerCase
- trim

Increment in Expression

- You can increment a variable in an expression

```
int cat = 3, dog = 5, bird = 7;
cat = dog + bird++;
```

- **cat** has the value 12, **bird** has the value 8.
- Post increment (with the ++ after the variable) logically occurs **after** the rest of the statement
- You cannot use ++ on an expression

```
dog = (cat + mouse)++; // error
```

Pre-Increment and Decrement

- You can also increment or decrement a variable by putting ++ or – before the variable

```
++frog;      --lizard;
```

- Pre-increment and decrement can also be used in an expression.

```
int  cat = 3, dog = 5, bird = 7;
cat = dog + ++bird;
```

- **cat** has the value 13, **bird** has the value 8.
- Pre increment (with the ++ **before** the variable) logically occurs before the rest of the statement

Big Computer Science Concepts

- Inheritance
- Methods
- Constructor methods
- Creating objects
- Types of variables
- Strings are different from numbers

Likely Exam Questions

- Convert math equations to Java
- Show what will be displayed by a program
- Complete parts of a program
- Complete a GUI program

Technical Interview Prep Sessions

- Tuesday, September 24 at 4:00-5:30PM in McNair 132
- Target audience is first semester freshman CS majors who are taking COMP163
- Will give hands-on practice at the kind of technical problem solving and coding questions that are given during Google interviews
- Demystify the interview process
- Share what I and other Google interviewers have looked for over the last decade
- Led by Dr. Dave Foulser of 

First Exam

- The first exam in COMP163 will be in lecture on Monday, September 23
- The exam will cover everything since the beginning of the semester
- You are allowed one 8½ by 11 inch page of notes
- Most questions will be of the form
  - complete this program
  - what does this program display