NOTE: This is intended to be harder than the actual exam. If you feel comfortable with these examples, you should be okay on the exam itself. It is intended that you will work with others, posting questions to piazza when needed. We will not be providing direct answers to these questions but will however answer questions you may have. You are more than welcome to post/share your answers with whomever you like, however you like.

This is NOT an individual assignment! It’s not an assignment at all!

That said this is by no means comprehensive meaning that items not listed here may still be on the final exam which will cover the contents of the entire course! I CAN’T SAY THIS ENOUGH: The best way to prepare is to RE DO all of the clicker questions and worksheets and review all the PSAs to make sure you can explain exactly what’s going on and perform similar tasks. And make sure you practice writing code on paper.

Also keep in mind that this final review as of yet is not final! What this means is that we maybe adding to it over the next couple of days. As stated above, this is by no means intended to be comprehensive but instead act as a guide to help you study.
Overview

Loops
- Write for loops/for-each loops/while loops
- Write code that iterates over an array to do something with the elements in the array (e.g., find patterns, count elements, etc).
- Write nested loops to work with 2D arrays (again, find patterns, calculate properties of the data, etc)
- Reason about the behavior of nested loops by tracing the code.
- Write code that iterates over ArrayLists and ArrayList<T> (again to do something interesting with the data stored within).
- Identify errors in loop conditions
- Write boolean expressions (using &&, ||, <, >, ==, etc) to correctly control loops

Strings
- == vs .equals() (do not use .match() it doesn’t do what you think it does)
- Use common String methods (substring(), charAt(), toCharArray(), etc)
- Explain how strings are immutable. Trace code that “modifies” strings by constructing new strings. Write code to “modify” a string by constructing a new string.

File IO
- Scanner class
- has* methods vs next* methods

Objects, Classes, Inner Classes and Inheritance
- Describe the difference classes vs objects
- Draw memory modes with objects (including scope)
- Use and draw memory models for inner classes
- Describe the difference between memory models and subclasses
- Describe the difference between overloaded and overridden methods
- Know how to write a subclass
- Know what Access Level Modifiers (ie. private, public, protected) do
- Determine if a particular method can access given members in other classes based on the given members Access Level Modifier (ie. private, public, protected)
- Describe the difference between Static and Non-Static members and methods
- Explain at a high level what polymorphism is
- Determine which overridden method is called at runtime
- Describe what the toString() method does

Casting
- Know what casts do
- Casting primitives (ie int, double, etc) vs casting references to objects
- Know when a cast (or lack thereof) results in a compile time error
• Know when a cast results in a runtime error
• Use casting correctly

Abstract classes and Interface
• Describe when a class must be abstract
• Write an abstract method
• Know how to define an interface
• Know how to use an interface
• Know the difference between an interface and an abstract class
• Know how to extend and abstract class or implement an interface

GUI/Applet stuff
• Know what is a MouseListener, ActionListener, KeyListener
• Know how to define a custom listener by extending Mouse/Action/KeyListener
• Know how to add a listener to a container (ie a JPanel) using the appropriate method

Generics
• Know what the T in ArrayList<T> means
• Know what can be assigned to generic references

Recursion
• Write simple recursive methods
• Trace recursive methods, including drawing stack frames
• Identify the base case(s) (Yes, there can be more than one!)
• Identify the recursive call(s) (Yes, there can be more than one!)
• Know the basics of recursion
  a. need a base case
  b. need to recurse on a smaller problem (eventually working toward the base case)

Threads
• Know the basic concept of what a thread is
• Know the basic thread methods (start(), sleep(), etc)
• How to create your own thread.
• Describe what happens when you put the main GUI thread to sleep

Exceptions
• Know what an exception is
• Writing a try/catch block possibly with multiple catches
• Difference between Checked and Unchecked Exceptions

Linux
• Know basic linux command
• Know the difference between absolute and relative paths
Examples

Generics
For the following questions assume class **ChildClass** extend **ParentClass**. In other words there exist two class for which **ChildClass** is a subclass of **ParentClass**

1. In the statement “ArrayList<T>”, what is T called?
   a. Type Parameter
   b. Type
   c. That T thingy
   d. Parameter

2. Do the following compile? Why or Why Not?
   a. ArrayList<T> al= new ArrayList<Integer>();
   b. ArrayList<T> al= new ArrayList<T>();
   c. ArrayList<Integer> al = new ArrayList<Integer>();
   d. ArrayList<int> al = new ArrayList<int>();
   e. ArrayList<Double> al = new ArrayList<Integer>();
   f. ArrayList<ParentClass> al= new ArrayList<ChildClass>();
   g. ArrayList<ChildClass> al= new ArrayList<ParentClass>();

Loops

1. Using a *for loop*, *for-each loop* and a *while loop*, write three different code snippets which initialize and print the elements of an integer array **backwards** using the following values:
   
   `{1,2,4,8,16,32,64,128,256,512,1024}`

2. Do the same as above but this time use an ArrayList<Integer>

3. Would it have been possible to have an ArrayList<int> instead of ArrayList<Integer]? A simple “yes” or “no” is all we are looking for. You do not need to say why, but if you have an answer why, we’re more than happy to hear it.

4. For every loop type, what did you have change, if anything, when using an int[] versus an ArrayList<Integer> from questions 1 and 2?

5. Write a nested loop (pick any loop type you like) to print the contents of a double array of Strings a *column* at a time separated by an underscore ‘_’ using the following array:

   ```java
   String[][] arr = {
   {"00","01","02","03"},
   {"10","11","12","13"}
   };
   //Add your nested loops here
   ```

   The results should be:

   ```plaintext
   “00_10_01_11_02_12_03_13”
   ```

Objects, Class, and Inheritance

1. What is the difference between an Object and Class?

2. What does it mean for a subclass to *override* a superclass’ method?

3. Write class named Bar with a toString() that returns the class’ name.

4. Write a class Foo that extends the class Bar written in question 2 above overriding the toString() method to return “Foo” and not “Bar”
1. Using the above classes, do the following compile? Why or Why not? If not, is it a compiler or runtime error? If they do compile, what gets printed?
   a. `Snow snow = new Fog();
      snow.visibility();`
   b. `Snow var1 = new Rain();
      var1.temperature();`
   c. `Snow snow = new Rain();`
((Fog)var1).visibility();

d. Snow snow = new Fog();
   snow.burnOffInAfternoon();

e. (new Rain()).temperature();

f. Fog fog = new Snow();
   fog.visibility();

g. Rain rain = new Rain();
   Fog fog = (Fog) rain;
   Snow snow = fog;
   snow.temperature();

h. Snow snow = new Fog();
   snow.burnOffInAfternoon();

i. ((Snow)new Fog).visibility();

j. ((Snow)new Fog).burnOffInAfternoon();

2. Does casting an object to its base class change its runtime type?
   For example:
   Fog fog = new Fog();
   Snow snow = (Snow)fog; //is the underlying object for “fog” changed by this cast?

Threads

1. What is a thread?
2. What class must you extend to define a custom thread?
3. When method do you call to run your custom thread?
Linux
You may need to look some of these up. Google will be your friend!

1. What does the ~ (tilde) mean in a Linux Path?
2. Which of the following are “Absolute paths”? Which are “Relative paths”? (Assume all directories exist)
   a. ../parentDirectory
   b. /home/myHouse/myDirectory
   c. ~/myDirectory
3. Match the following from the Linux commands on the left to the description of the command on the right. (Draw arrows if it makes things easier)

<table>
<thead>
<tr>
<th>Linux Command</th>
<th>Linux Command Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. mv</td>
<td>1. access the manual pages for a given command</td>
</tr>
<tr>
<td>2. ls</td>
<td>2. move or rename a file</td>
</tr>
<tr>
<td>3. grep</td>
<td>(renaming can be thought of moving the file to a new name)</td>
</tr>
<tr>
<td>4. cd</td>
<td>3. change directory</td>
</tr>
<tr>
<td>5. pwd</td>
<td>4. search for a given word inside of a given file</td>
</tr>
<tr>
<td>6. man</td>
<td>5. print the current working directory</td>
</tr>
<tr>
<td></td>
<td>(where you currently are)</td>
</tr>
<tr>
<td></td>
<td>6. list the contents of the current directory</td>
</tr>
</tbody>
</table>

File IO (Using Scanner)

1. Scanner s = new Scanner(new File("test.txt");
   
   if(__________)
   
   s.nextInt();

   What should be in the if statement above to test that there is actually an int to read from the file?

2. What is the difference between the next() and hasNext() methods?
3. What is the difference between the next() and the nextLine() methods?
4. What is the difference between the hasNext() and hasNextLine() methods?
5. If you wanted to read a double from a file, which Scanner method would you call? Which method would you use to test if there was a double to read?

Exceptions
All Exception questions dealing with code will refer to the code below:

Scanner scan = new Scanner(new File("someFile.txt");
return scan.nextInt();
1. Frank is trying to create a Scanner object to read an int from a file and return the read int. Every time he tries to compile his program, he gets an error stating: 

   Unhandled exception type FileNotFoundException

   What does Frank need to add to his code to get this to compile and print the error 

   System.err.println("File Not Found"); when the file cannot be found?

2. Sally attempts to use Frank’s fixed code (from problem 1). Her file, however, contains a double and not an int. Every time she runs the program it crashes with an InputMismatchException. She tells Frank about the problem, asking him to add the error message System.err.println("Non-int value in file"); Frank updates his code by adding a catch block to the original try/catch he added in question 1.
   Adding to your solution from question 1, write the code Frank has so far.

3. Bob then attempts to run Frank’s new program using an empty file. This time, Bob gets a NoSuchElementException. He informs Frank about this problem asking him to add the error message System.err.println("No input left in file"); Frank once again solves this problem by adding yet another catch block to his program.
   Adding to your solution for 2, write the last update Frank makes to his program.

4. What is the difference between Checked and Unchecked Exceptions in Java? In other words, why was it a compiler error for question 1 versus runtime errors for questions 2 and 3?

Recursion

```java
/*01*/ public class Recursive
/*02*/ {
/*03*/   public static void main( String [] args )
/*04*/   {
/*05*/     down_up(8);
/*06*/   }
/*07*/   public static void down_up( int n )
/*08*/   {
/*09*/     System.out.println(" Level: " + n );
/*10*/     if( n > 1 )
/*11*/       down_up( n-2 );
/*12*/     System.out.println(" Level: " + n );
/*13*/   }
/*14*/ }
```
1. Identify which line the base case is on
2. Identify which line the recursive call is on
3. When run, does this result in a StackOverflowException? If yes, what needs to be fixed?
4. Assuming you fixed any issues that may have existed, what gets printed in the above code?
5. What is the Runtime Stack?
6. What is the maximum number of stack frames that will be generated by the program? Keep in mind the main() method is on the runtime stack too!

Abstract classes and Interface
1. Abstract classes and interfaces both declare methods without bodies that user of the abstract class/interface must implement in their code. What is the main differences between an abstract class and an interface?
2. What two things may you do if you extend an abstract class in order to compile? What if you implement an interface?
3. Janice has been writing a lot of GUI's lately. All of her GUIs have needed to implement a custom MouseListener as an inner class to handle users mouse clicks. She has, however, gotten tired of writing stubs for all the MouseListener methods she didn't use. Janice had an idea to create a class named MouseAdapter which she could extend her inner classes from so she no longer needed method stubs in her custom MouseListeners for her GUIs.
Write the code she would have written for her MouseAdapter class.

    public class MouseAdapter implements MouseListener{
        //you code here
    }

GUI/Applet stuff
Below are the code for a JApplet and the MouseMotionListener Interface which are defined in two different files. Don’t worry too much about understanding every line. This example is using an interface you haven’t seen before.

Code shamelessly borrowed/modified from: http://www.javakode.com/applets/07-backbuffer/

/*01*/ import javax.swing.*;
/*02*/ import java.awt.*;
/*03*/ import java.awt.event.*;
/*04*/ import java.lang.Math;
/*05*/
/*06*/ public class NoBackbuffer1 extends JApplet implements MouseMotionListener {
    /*07*/
    /*08*/    int width, height;
    /*09*/    int mx, my; // the mouse coordinates
    /*10*/    Point[] points;
```java
/*11*/ int N = 300;
/*12*/
/*13*/ public void init() {
/*14*/    width = getSize().width;
/*15*/    height = getSize().height;
/*16*/    setBackground( Color.black );
/*17*/
/*18*/    mx = width/2;
/*19*/    my = height/2;
/*20*/
/*21*/    points = new Point[ N ];
/*22*/    for ( int i = 0; i < N; ++i ) {
/*23*/        int x = (int)(( Math.random() - 0.5 ) * width / 1.5);
/*24*/        int y = (int)(( Math.random() - 0.5 ) * height / 1.5);
/*25*/        points[i] = new Point( x, y );
/*26*/    } 
/*27*/    addMouseMotionListener( this );
/*28*/ }
/*29*/
/*30*/
/*31*/ public void mouseMoved( MouseEvent e ) {
/*32*/    mx = e.getX();
/*33*/    my = e.getY();
/*34*/    showStatus( "Mouse at (" + mx + "," + my + ")" );
/*35*/    repaint();
/*36*/    e.consume();
/*37*/ }
/*38*/
/*39*/
/*40*/ public void update( Graphics g ) {
/*41*/    g.setColor( Color.white );
/*42*/    for ( int j = 1; j < N; ++j ) {
/*43*/        Point A = points[j-1];
/*44*/        Point B = points[j];
/*45*/        g.drawLine( mx+A.x, my+A.y, mx+B.x, my+B.y );
/*46*/    }
/*47*/ }
/*48*/ }
```

The source for the interface MouseMotionListener

```java
/*01*/ public interface MouseMotionListener extends EventListener
/*02*/ {
/*03*/ /**
/*04*/   * This method is called when the mouse is moved over a component
/*05*/   * while a button has been pressed.
/*06*/   */
/*07*/   * @param event the <code>MouseEvent</code> indicating the motion
/*08*/   */
```
1. What would it take to convert the above code into a JFrame? Copy and paste the JApplet (not the MouseMotionListener) code into your favorite editor and create a JFrame based GUI. Make sure your solution compiles and runs!
2. When does the init() method on line 13 get called?
3. What does the update() method on line 40 do? Note we are not asking about the method contents, but about the general purpose of the method.
4. Why did we have to define a method stub on line 38 of the JApplet code?