# CS163/164 Final Exam Study Session

# Review

# What is printed?

```
public static void main (String [] args){
  String s = "Summer Break";
  System.out.println(s.indexOf('c'));
  System.out.println(s.indexOf('e'));
  System.out.println(s.charAt(2));
  System.out.println(s.length());
}
```

-1
4
m
12

Using printf print the double variable called **d** with 2 decimal point precision (with a newline character)

```
System.out.printf(“%.2f\n”, d);
```

Create a 2D array of doubles called **dArray** with a width of 4 and a height of 5.

```
double [][] dArray = new double[5][4];
```

# I/O

Create a Scanner that reads from the keyboard called **keys**

```
Scanner keys = new Scanner (System.in);
```

Create a Scanner (called **fReader**) that reads from the file named **in.txt**. Include a try-catch.

```
try {
  Scanner fReader = new Scanner (new File (“in.txt”));
} catch (IOException e) {
  System.out.println(e.getMessage());
}
```
OR
```
try {
  File f = new File (“in.txt”);
  Scanner fReader = new Scanner (f);
} catch (Exception e) {
   System.out.println(“Can’t read from in.txt”);
   System.exit(-1);
}
```
Note 1: You must create a new File object (whether inside or outside the Scanner declaration) this makes it read from a file. If you don’t have it, it would read in.txt as a String (ie fReader.nextLine() would return in.txt not the first line).

Create a Scanner (called **fReader2**) that reads from the file name stored in the String **filename.** Include a try-catch.

```
try {
  Scanner fReader2 = new Scanner (new File (filename));
} catch (FileNotFoundException error) {
  System.out.println(error.getMessage());
}
```
OR

```
try {
  File f = new File (filename);
  Scanner fReader2 = new Scanner (f);
} catch (IOException error) {
  System.out.println("Can't read from " + filename);
}
```

Note 1: You must create a new File object (whether inside or outside the Scanner declaration) this makes it read from a file.

Note 2: You can use Exception, IOException, or FileNotFoundException. You can use some sort of println in your catch using exceptionName.getMessage() or a customized println. If we don't specify on the exam which to use you can assume we'll accept multiple answers.

Create a PrintWriter (called **writer**) that writes to the file name stored in **outfile.** Include a try-catch.

```
try{
  PrintWriter writer = new PrintWriter (new File (outfile));
} catch (Exception e) {
  System.out.println(e.getMessage());
}
```

OR

```
try{
  File f = new File (outfile);
  PrintWriter writer = new PrintWriter (f);
} catch (FileNotFoundException e) {
  System.out.println("Can't write to " + outfile);
}
```

Note 1: You can use Exception, IOException, or FileNotFoundException. You can use some sort of println in your catch using exceptionName.getMessage() or a customized println. If we don't specify on the exam which to use you can assume we'll accept multiple answers.

Read the next word, int, and next line from the predefined Scanner called **read**. Print the word, int, and next line separated by colons (:)

```
String word = read.next();
int num = read.nextInt();
read.nextLine();
String line = read.nextLine();
System.out.println(word + “:” + num + “:” + line);
```

Write the the following variables' content to a file using the predefined PrintWriter called **pw**. all on same line but separated by spaces.

- **d** - type double
- **word** - type String
- **line** - type String
- **i** - type int

```
pw.printf(“%f %s %s %d”, d, word, line, i);
```

OR

```
pw.print(d + “ “ + word + “ “+ line + “ “ + i);
```

# ArrayList

Declare an ArrayList of type **int** called **iList**.

```
ArrayList <Integer> iList = new ArrayList<Integer>();
OR
ArrayList <Integer> iList = new ArrayList<>();
```

Print (with a new line) the size of the predefined ArrayList called **list**.

```
System.out.println(list.size());
```

Create a **String** ArrayList called **strList** and add “code blooded” to the end of **strList**.

```
ArrayList <String> strList = new ArrayList<String>();
strList.add(“code blooded”);
```

Use the **String** ArrayList from the last slide (called **strList**) and add “happy 3rd exam” at the first index and then add “almost there” to the end.

```
strList.add(0, “happy 3rd exam”);
strList.add(“almost there”);
```

Remove the 2nd element of **strList**.

```
strList.remove(1);
```

Print **strList**

```
System.out.println(strList);
```

Remove “abc” from the predefined **String** ArrayList called **s**.

s.remove(“abc”);

# Objects and Classes

Given the following class and instance variable declarations write a constructor that assigns every instance variable.

```
public class Assignment {
  private String name;
  private String dueDate;
  private String className;

  // constructor goes here

}
```

```
public Assignment (String n, String d, String c) {
  name = n;
  dueDate = d;
  className = c;
}
```
(your parameters may have different names)

OR

```
public Assignment (String name, String dueDate, String
className) {
  this.name = name;
  this.dueDate = dueDate;
  this.className = className;
}
```

Create an Assignment object (using your constructor from the last slide) called **hw1** with the name: HW1, due date: 04/24/17, class name: CS163

```
public class Assignment {
  private String name;
  private String dueDate;
  private String className;

  // constructor goes here
}
```

```
Assignment hw1 = new Assignment (“HW1”, “04/24/17”, “CS163”);
```

Create a toString method that prints an Assignment object's name, due date, and class name separated by commas.

```
public class Assignment {
  private String name;
  private String dueDate;
  private String className;

  // constructor goes here
}
```

```
public String toString() {
  return name + “,” + dueDate + “,” + className;
}
```

(could also split up code and create a String variable and return that String variable)

# Tracing

(for each of the following questions, write what is printed).

```
import java.util.Arrays;
public class Trace1 {
  public static void mystery (int [] iArray){
    for (int i = 1; i < iArray.length - 2; i++){
      if (iArray[i] % 2 == 0)
        iArray[i] += 1;
    }
  }
  public static void main (String [] args){
    int [] array = {1, 4, 23, 8, 42, 1, 2};
    mystery(array);
    System.out.println(Arrays.toString(array));
  }
}
```

**[1, 5, 23, 9, 43, 1, 2]**

```
public class Trace1 {
  public String mystery (String s){
    if (s.length() <= 0) return s;
    else
      return s.charAt(0) + mystery(s.substring(1));
  }
  public static void main (String [] args){
    String s = "cold";
    Trace1 t = new Trace1();
    System.out.println(t.mystery(s));
  }
}
```

cold

```
import java.util.ArrayList; import java.util.Scanner;
public class Trace1 {
  private ArrayList<String> words = new ArrayList<>();
  public void readFile (String filename){
    try {
      Scanner read = new Scanner (filename);
      while (read.hasNext())
        words.add(read.next());
    } catch (Exception e){
      System.out.println(e.getMessage());
    }
  }
  public static void main (String [] args){
    Trace1 t = new Trace1();
    t.readFile("in.txt");
    System.out.println(t.words);
  }
}
```

in.txt:
3 Hey!
How's it going?
Fun Stuff…

**[in.txt]**

```
public static void mystery (int i){
  if (i == 0 || i == 1) return;
  else {
    System.out.print(--i);
    mystery(i);
  }
}
public static void main (String [] args){
  mystery(5);
}
```

4321

```
public static void read (String inFile, String [] lines){
    try {
      Scanner read = new Scanner (new File ("inFile"));
      lines = new String[read.nextInt()];
      for (int i = 0; i < lines.length; i++) {
        lines[i] = read.nextLine();
      }
    } catch (IOException e){
      System.out.println("can't read: " + inFile);
      System.exit(-1);
    }
  }
  public static void main (String [] args){
    String [] lines = null;
    read ("in.txt", lines);
    System.out.println(Arrays.toString(lines));
  }
```

inFile:
3 Hey!
How’s it going?
Fun Stuff…

can't read: in.txt

# What is in the output file?

```
public static void writeFile (String filename, ArrayList<String> list){
  try {
    PrintWriter pw = new PrintWriter (new File (filename));
    for (String s : list){
      pw.println(s);
    }
    pw.close();
  } catch (IOException e){
    System.out.println(“ERROR");
    System.exit(-1);
  }
}
public static void main (String [] args){
  ArrayList<String> lines = new ArrayList<String>();
  lines.add("line4"); lines.add("line3"); lines.add("line2"); lines.add("line1");
  writeFile("out.txt", lines);
}
```

line3
line2
line4
line1

# What is in the output file?

```
public static void writeFile (String filename, ArrayList<String> list){
  try {
    PrintWriter pw = new PrintWriter (new File (filename));
    for (String s : list){
      pw.println(s);
    }
 } catch (IOException e){
    System.out.println(“ERROR");
    System.exit(-1);
  }
}
public static void main (String [] args){
  ArrayList<String> lines = new ArrayList<String>();
      lines.add("line4"); lines.add(0,"line3");
      lines.add(1,”line2"); lines.add("line1");
  writeFile("out.txt", lines);
}
```

Nothing…didn’t close my Print Writer

# General Questions

What is a toString method used for? When should I use it?

toString methods are used to specify the printing format for objects and classes you create. For example, if you make a Triangle class (and in turn an object of type Triangle) and you try and print Triangle without a toString method you will print out the akdfjl;kj;@kjflkjaeoifj mess (memory address), when you have a toString method you can tell the compiler to print the object in a certain format (ie. side1: 3, side2: 4: side3: 89).

Use a toString method anytime you want to print an object of class you created.  Notice when we make our projects and classes a lot of times we don't ask you to write a toString method (ie Q6.java didn't have a toString method) because we weren't asking you to print an object of that Class (ie Q6 q6 = new Q6();) we don't end up printing q6 we just use it to call your methods.

What is a constructor used for? When should I use it? If I don’t add one myself is there still a default one?

A constructor is used to create objects. For example, if we had a class called Music and we wanted to make an object of type Music we could have a constructor that takes two Strings (title and artist) and an int (release year). We might want to do this to make sure that every Music object we make has these attributes. However, we could use the default constructor which takes no parameters and we could manually change the instance variables (if they're public), this can be dangerous because it doesn't guarantee every instance variable is assigned a value.