Midterm
CS9053 (Tue. 3:30pm – 5:50pm)
April 7, 2020
Dr. Dean Christakos

Rules: Part I and Part II Question 1 should follow the terms of an open-book, take home exam: you may consult any textbooks or online sources you choose. Part II Question 2 is like a normal homework assignment—feel free to discuss with your classmates or ask me about it during office hours.

Part I: Written Section

1) In Java, as in most programming languages, * is multiplication, + is addition, - is subtraction, and / is division. What does that % operator do?

2) What is the difference between an overridden and overloaded method?

3) When multiplying an int value and a float value, what will be the type of the result?

4) You have an array of objects called "`objs`" that contains a list of Integer objects. It was declared like this:

```
Object[] objs
```

It doesn't matter how the Integer objects got there or what their values are.

You have a variable `int intVal`. Write some code (this should just take at most 2 lines, but you could do it in 1 line) that gets the integer value of the first object in the `objs` array and assigns that value to `intVal`.

5) Take that object from the array. The result of `(new Object()).toString()` is something like `java.lang.Object@15db9742` - the object type followed by the @ sign and a unique identifier. In the `objs` array from question #4, which is filled with `Integer` objects, let's say we execute the following:

```
objs[5] = new Integer(20);
Object myObj = objs[5];
```

What is the output of `myObj.toString()` ? Explain why.

6) You have two strings: `String s1 = "Alice"` and `String s2 = "Bob"`

What is the result of `(s1 == s2)` ?

What is the result of `s1.compareTo(s2)` ?
Explain your answers – why each gives the result it does and what that means.

7) Remember in an earlier lecture when discussing Abstracts, Interfaces, and Generics we had the class `Fruit`, declared as:

```
public abstract class Fruit { public Fruit(String name) { /*etc*/ } /*
etc */ }
```

We created Apple and Orange objects. Why didn't we ever simply do the following:

```
Fruit f = new Fruit("Apple"); ?
```

8) In an earlier lecture, we had two abstract classes, `Animal` and `Fruit`, from which we made various subclasses. Some of those subclasses implemented the interface `Edible`. Why did we choose to make classes subclasses of `Animal` and `Fruit`, but have `Edible` as an interface? To put the question more generally, why would you design an object as a subclass of another object vs. why would you create an interface and have an object implement an interface? (There's no single right answer to this question, but there are definitely wrong answers)

9) Assume there is a file data.txt that should contain two integers per line (eg, a sample line is "5 10"). This program reads in each line and divides the first number by the second number.

```
Class Main {
 Public static void main(String[] args) {
       FileReader file = new FileReader("data.txt");
       BufferedReader fileInput = new BufferedReader(file);

       String inLine = fileInput.readLine();
       while (inLine!= null) {
             String[] numbers = inLine.split(" ");
             Integer a = Integer.parseInt(numbers[0]);
             Integer b = Integer.parseInt(numbers[1]);
             Integer c = a/b;
             System.out.println("result = " + c);
             inLine = fileInput.readLine();
       }
 }
}
```

   a) This is not going to compile because of the IOExceptions raised by the FileReader and BufferedReader constructors. Modify the code so it will compile.
   b) Add a try/catch block in the while loop to catch any Runtime exceptions. What Runtime exceptions could be raised that you might want to look out for?

10) If we see a method and see which arguments it takes, how do we know if we can pass in a Lambda function? Specifically, **what kind of parameter type does a method have to accept for you to pass in a lambda function?**

11) Look at this code:

```
public class TimerMessage {
 private String msg;

 public TimerMessage(String msg) {
            this.msg = msg;
 }

 public void start() {
       TimePrinter listener = new TimePrinter();
        Timer timer = new Timer(1000, listener);
        timer.restart();
}

public class TimePrinter implements ActionListener {
       public void actionPerformed(ActionEvent event) {
             System.out.println(msg);
       }
}
}

public class TimerMessageRunner {
       public static void main(String[] args) {
             TimerMessage tm = new TimerMessage("Hello!");
             tm.start()
       }
}
```

The `msg` field of `TimerMessage` is `private`. In the `main` method of the `TimerMessageRunner` class, if I executed `System.out.println(tm.msg)`, I would get an error. However, the `TimePrinter` method can access `msg` from the `TimerMessage` class without any problem. Why?

12) The following code opens a window with a "Close" button, but it does nothing. What do you need to add to the code so that when you press the "Close" button, it executes `System.exit(0)`, terminating the application?

```
public class TestFrame extends JFrame {

   private static final int FRAME_WIDTH = 200;
   private static final int FRAME_HEIGHT = 75;
   private JButton closeButton = new JButton("Close");

   public TestFrame()
   {
      setSize(FRAME_WIDTH, FRAME_HEIGHT);
      this.setLayout(new FlowLayout);
      closeButton.addActionListener((e) -> System.exit(0));
      this.add(closeButton);
   }
```

```
    public static void main(String[] args) {
          TestFrame testFrame = new TestFrame();
          testFrame.setVisible(true);
    }
}
```

Part II: Coding Section

1) **Inheritance**

   Demonstrate your ability to program using inheritance in the Java programming language.

   Demonstrate your knowledge of proper equals implementations in the Java programming language.

   Demonstrate your knowledge of variable arguments and enum types in the Java programming language.

Instructions

- There are three tasks to complete
  - Create an object hierarchy which mimics the classification for players of winter sports. The assignment here is to design a logical ontology and class hierarchy for the sports.
    - The hierarchy should be contained within package edu.nyu.cs9053.midterm.hierarchy
    - There should be a base class called WinterSportPlayer
    - There should be the following subtypes:
      - Luger
      - IceSkater
      - Skier
      - SpeedSkater
      - Curler
      - Sledder
      - Bobsledder
      - CrossCountrySkier
      - MogulSkier
      - FigureSkater

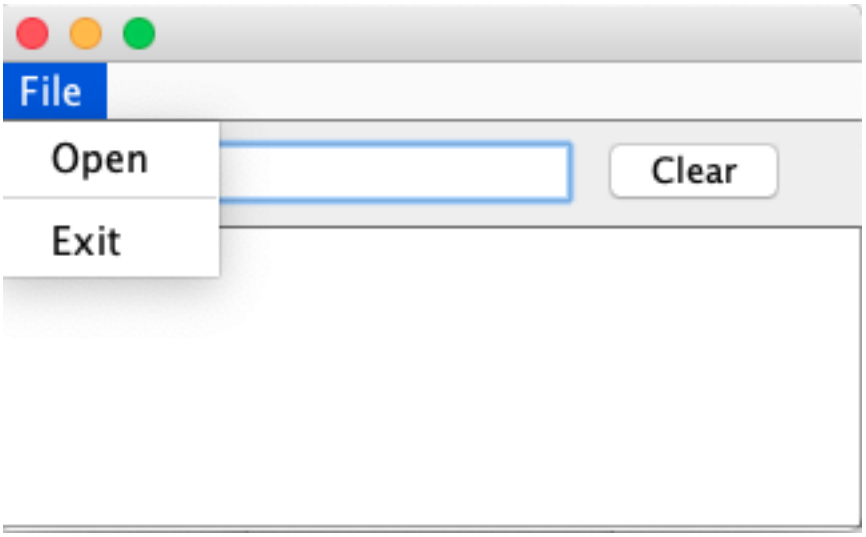
        - Each of the subtypes should have the following methods:
            - getName returns a name for the player as a String. This value is per object and not defined by the class it should be used for equality.
            - getAge returns the age of the player as an int. This value is per object and not defined by the class it should be used for equality.
        - If appropriate, the subtypes should have the following methods:
            - getSkateSize returns the size of the player's skate as an int.
            - getSledColor returns a String representing the color of the sled the player users.
            - getSkiLength returns the length in centimeters of the player's skis as an int.
        - For each class which has no sub-types itself, add an instance field to the class particular to the type which is also used in the equals methods. For example, a curler might have a field “trouserPattern” or “brushLength,” that other classes don’t have. There should be a field that is unique to that class that no other class has.
    - Provide implementations of the equals and methods for each concrete class within package edu.nyu.cs9053.midterm.hierarchy
        - Note, these methods are testing equality for the objects and so should only include checks on type information and object specific values.
        - Note, make sure code from the super classes is used by the subclasses rather than just rewriting it in the subclasses.

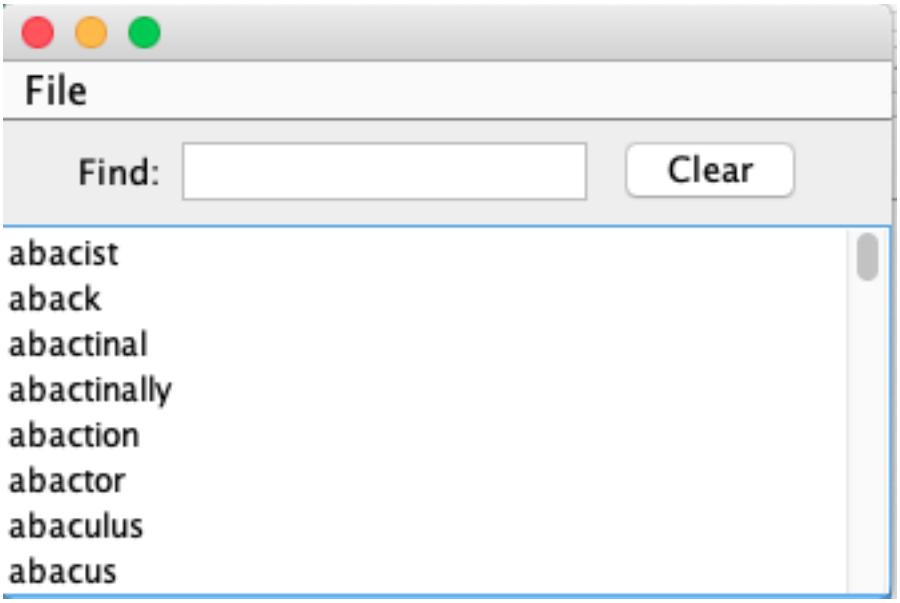
Implementation

- Ensure your code is correct by compiling and testing it
- A portion of your grade will be based upon readability and organization of your code.
    - Follow the naming guidelines of lecture
    - Break large functions into multiple functions based on logical organizations

2) Java Swing

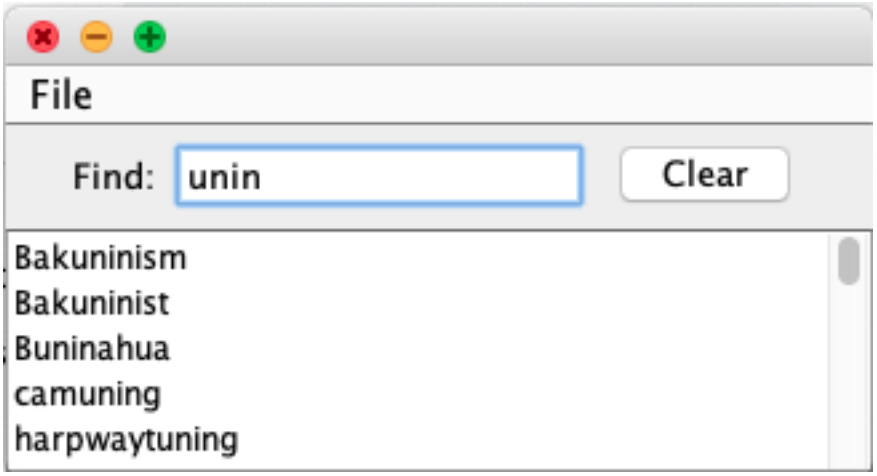
You should build an interface that looks like this:





The interface should have the same layout as shown above. When the program is run, it starts empty. You should then be able to load a list of words from the file `data/words`, provided in the Java project for this midterm. The first 100 words are in the file `data/words100`, if you want to experiment because that takes far less time to load.

The Find text field contains the user's query. When the query is blank, the list box displays the entire word list, as shown above. Whenever the query changes, the list box immediately updates to display all words that contain the query text:



The text box should update constantly as the user types. Pressing Enter should not be necessary. (Hint: this requires you to use a listener that receives every change to the text field; see the JTextField class overview for a hint about which listener to use. It was also mentioned in lecture).

If none of the words contain the query, the text box should be empty.

The Clear button should clear the query field, restoring the list box to displaying all words again.

An outline is contained in the file `WordFinder.java`. To search, you will use the class `WordList.java`. `WordList.java` is loaded using the method `WordList.load`, which takes a `FileInputStream`. All this will be handled in the `OpenFileListener`, the structure of which, including the file selection process, is written. You need to figure out how to get the results out of `WordList`, based on your search term.

Hint: to scroll a `JTextArea` all the way to the top, use the method `setCaretPosition(0)`

Hint: you can provoke an action in a Java Swing objection using the method `postActionEvent();`