Name:_____ UNM Username:_____

Answer all questions in the space provided. Write clearly and legibly, you will not get credit for illegible or incomprehensible answers. Print your name at the top of every page. If you include additional scratch pages, put your name on them, too.
This is a closed book exam. However, each student is allowed to bring one page of notes to the exam. Also, you are permitted the use of a "dumb" calculator to perform basic arithmetic.

Question:	1	2	3	4	5	6	7	8	Total
Points:	22	10	8	13	9	10	18	10	100
Score:									

1. Write each answer in the blank provided.

   (a) If I have a member variable in a top level class that is visible to methods within a class nested inside it, but not to methods within another top level class in the same package, what access modifier have I used on the member variable? (2)

   (a) _____

   (b) What is the keyword used to call the parent class version of a method overridden in the child class? (2)

   (b) _____

   (c) What is the keyword used to access a member variable hidden by a local variable? (2)

   (c) _____

   (d) What interface does a class implement if you can use an instance of the class in an enhanced for loop (also known as a for-each loop)? (2)

   (d) _____

   (e) Name a method inherited by all reference types in Java. (2)

   (e) _____

   (f) Name a keyword used in exception handling. (2)

   (f) _____

   (g) What is the value of the following expression? (2)

   ```
   (3 < 2 && 5 % 2 != 0) ? "ABCD".charAt(1) : "EFGH".charAt(2)
   ```

   (g) _____

   (h) If I declare an `int` member variable but do not initialize it, what is its value? (2)

   (h) _____

   (i) If I declare an `Integer` member variable but do not initialize it, what is its value? (2)

   (i) _____

   (j) If I want to perform custom painting in a `JPanel`, what method do I override? (2)

   (j) _____

   (k) The `Map` interface does not extend `Collection`, but does provide several methods to view the map as a collection. Name one of these methods. (2)

   (k) _____

2. Here is a small class that represents a counter that counts up by one from an initial value.

```
public class Counter {

    private int value;

    public Counter(int initialValue) {
        value = initialValue;
    }

    public void increment() {
        value++;
    }

    public final int getValue() {
        return value;
    }
}
```

I want you to extend `Counter` to create a simple but complete class named `DoubleCounter` that represents a counter that counts up by two from an initial value. (10)

You do not need to include comments in your implementation.

The output of the test code below would be:

```
0 0 10 10
1 2 11 12
2 4 12 14
3 6 13 16
4 8 14 18
```

Hint: *Pay attention to the modifiers* in the `Counter` class.

We could test both implementations like this:

```
    public static void main(String[] args) {
        Counter[] counters = { new Counter(0), new DoubleCounter(0),
                               new Counter(10), new DoubleCounter(10) };
        for(int i = 0; i < 5; i++) {
            for(Counter c : counters) {
                System.out.print(c.getValue() + " ");
                c.increment();
            }
            System.out.println();
        }
    }
```

3. Do the following code snippets successfully compile? If not, why not? Select the single correct answer for each.

(a) (2)

```
public class MyClass {
    private int x = 10;
    public static void main(String[] args) {
        x++;
        System.out.println(x);
    }
}
```

A. Cannot access private variable x from a public method.
B. Cannot access x without an instance of MyClass.
C. Variable x is out of scope in the main method.
D. Variable x is a constant, so cannot be incremented in main.
E. Some other error.
F. This code will successfully compile.

(b) (2)

```
public class MyListener implements ActionListener {
    private int count = 0;
    public void actionPerformed( ActionEvent ev ) {
        count++
    }
}
```

A. Missing constructor for MyListener class.
B. Should use "extends" instead of "implements" in first line.
C. Cannot access private variable count from a public method.
D. Variable count is out of scope in the actionPerformed method.
E. Some other error.
F. This code will successfully compile.

(c) (2)

```
List<boolean> truthValues;
```

A. The list is not initialized.
B. Cannot use primitive type boolean as generic type parameter.
C. Cannot use interface List as variable type.
D. List is an interface and cannot be instantiated.
E. Some other error.
F. This code will successfully compile.

(d) (2)

```
Set<String> names = new Set<>();
```

A. Cannot use String as generic type parameter.
B. Cannot use interface Set as variable type.
C. Set is an interface and cannot be instantiated.
D. Missing type parameter on right hand side of assignment.
E. Some other error.
F. This code will successfully compile.

4. Consider the following classes. What is the output of this code? (13)

```
public class Parent {

    protected static String a
        = "Boo!";
    protected String b;

    public Parent() {
        this(a);
    }

    public Parent(String a) {
        this.b = "Time to " + a;
    }

    public void test(int c) {
        a = "the answer is " + c;
    }

    public void test(String b,
                     int c) {
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        this.b = a;
    }
}
```

```
public class Child extends Parent {

    protected String c;

    public Child(String b) {
        this.b = b;
        this.c = a;
        a = "Hello";
    }

    public Child(String b, String c) {
        super(a);
        a = b;
        this.c = c;
    }

    public void test(int b) {
        test("Winter", b / 2);
        a = "Break";
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }

    public void test(String a) {
        super.test(42);
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }

    public static void main(String[] args) {
        Parent p = new Child("CS251");
        p.test(5);
        Child c = new Child("Final", "Exam");
        c.test("Goodbye");
        System.out.println(a);
    }
}
```

5. Consider the following program. Assume it has been compiled and is located in the same directory as a file named `test.txt`

```
import java.io.*;
public class ExceptionTest {

    public static void main(String[] args) {

        try (BufferedReader in =
             new BufferedReader(new FileReader(args[0]))) {
            String line = in.readLine();
            System.out.println(line);
        } catch (FileNotFoundException ex) {
            System.out.println("missing");
        } catch (IOException ex) {
            System.out.println("bad IO");
        } catch (Exception ex) {
            System.out.println("problem");
        } finally {
            System.out.println("done");
        }
    }
}
```

(a) If the file `test.txt` contains (3)

```
testing
one two
```

what is the output of running the following command?

```
java ExceptionTest test.txt
```

(b) If the file `bad.txt` does not exist, what is the output of running the following command? (3)

```
java ExceptionTest bad.txt
```

(c) What is the output of running the following command? (3)

```
java ExceptionTest
```

6. Write Java code to create a JButton with text "Click it!" that prints "Click it good!" to the console when it is pressed. Only create the button, you do not have to add it to a layout, show a window, etc. Use an *anonymous class* for the action listener. (10)

7. Consider the following interface describing a to-do list with tasks represented as Strings. (18)

```
import java.util.*;
public interface ToDoInterface {

    /**
     * Adds a task to the todo list.
     * @param task The task to add.
     */
    void addTask(String task);

    /**
     * Removes a task from the todo list after completion.
     * (It is possible the completed task is not the expected next task.)
     * @param task The completed task.
     */
    void completeTask(String task);

    /**
     * Get the next task from the todo list.
     * (Does not remove the task from the todo list.)
     * @return Next task to do, or null if none left.
     */
    String getNextTask();

    /**
     * Get all the tasks on the todo list.
     * @return All tasks we still have to do.
     */
    Collection<String> getAllTasks();
}
```

This problem continues on the next page.

Write a small but complete class named `ToDoList` that implements this interface, handling the tasks in a first-in-first-out order. Use a private member variable of an appropriate collection type to hold the tasks. *Solutions using more than one member variable will not receive full credit.* You do not need to include Javadoc comments.

8. Consider the following program. What would be displayed when it is run? Draw a picture to illustrate. (10)

```
import java.awt.*;
import javax.swing.*;

public class LayoutTest {

    public static void createAndShowGUI() {
        JFrame frame = new JFrame("Title Goes Here");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel1 = new JPanel();
        panel1.add(new JButton("A"));
        panel1.add(new JButton("B"));

        JPanel panel2 = new JPanel();
        panel2.setLayout(new BoxLayout(panel2, BoxLayout.PAGE_AXIS));
        panel2.add(new JButton("C"));
        panel2.add(new JButton("D"));
        panel2.add(new JButton("E"));

        frame.add(new JButton("F"), BorderLayout.CENTER);
        frame.add(panel1, BorderLayout.PAGE_END);
        frame.add(panel2, BorderLayout.LINE_START);

        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    createAndShowGUI();
                }
            });
    }
}
```