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:	14	8	12	26	10	5	15	10	100
Score:									

1. For the following questions, select the single *best* answer by circling your choice.

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

   ```
   1 + 2 * "3" + 4 + 5
   ```

   A. `16` B. `18` C. `"12345"` D. `"13345"` E. `"1645"` F. Some other value.
   G. The value of this expression is undefined.
   H. This expression would result in a compilation error.

   (b) Which type could `foo` be in the following code snippet? (2)

   ```
   boolean b = foo.contains("hello");
   ```

   A. `Collection` B. `Deque` C. `List` D. `Set` E. `Queue` F. `SortedSet`
   G. Any of these. H. None of these.

   (c) Which of the following types could `foo` *not* be in the following code snippet? (2)

   ```
   foo.add("hello");
   ```

   A. `Collection` B. `Deque` C. `List` D. `Map` E. `Queue` F. `Set`
   G. Any of these. H. None of these.

   (d) Which of the following is *not* a keyword used in exception handling? (2)
   A. finally B. catch C. final D. throw E. try F. throws

   (e) What is printed when the following code is compiled and executed? (2)

   ```
   public class StringCompare {
       public static void main(String[] args) {
           String s1 = new String("Test");
           String s2 = new String("Test");
           if (s1==s2) System.out.print("Same ");
           if (s1.equals(s2)) System.out.print("Equals");
       }
   }
   ```

   A. The code compiles, but nothing is displayed upon execution.
   B. `Equals` C. `Same` D. `Same Equals` E. Some other output.
   F. The output of this program is undefined. G. The code fails to compile.

   (f) Which of the following is *not* a keyword in Java? (2)
   A. continue B. do C. default D. enum E. if F. then G. throws

   (g) If you wanted to perform custom painting in a `JPanel`, which method would you override? (2)
   A. `draw` B. `drawComponent` C. `pack` D. `paint` E. `paintComponent`
   F. `redraw` G. `repack` H. `repaint` I. `repaintComponent` J. `refresh`

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

(a) (2)

```
ActionListener listener = ActionListener() {
    public void actionPerformed( ActionEvent ev ) {
      System.out.println( "Action happened" );
    }
};
```

A. ActionListener is an interface and cannot be instantiated.
B. Missing semicolon after call to ActionListener constructor.
C. Extra curly braces around actionPerformed method.
D. Need to add the listener to a JButton or Timer.
E. Some other error.
F. This code will successfully compile.

(b) (2)

```
public class MyClass {
    private static 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.

(c) (2)

```
Set<int> numbers;
```

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

(d) (2)

```
List<String> names = new ArrayList<>();
names.put("Jane");
```

A. Cannot assign an ArrayList to a List variable.
B. Missing type parameter on right hand side of assignment.
C. The put method is not part of the List interface.
D. The String `"Jane"` needs to be assigned to a variable.
E. Some other error.
F. This code will successfully compile.

3. Consider the following classes. What is the output of this code? (12)

```
public class Foo {
    protected double a;
    protected static int b = 2;

    public Foo() {
        this("Java");
    }

    public Foo(String a) {
        this.a = b / 2.0;
        b += a.length();
        System.out.println(a);
    }

    public void test(String a) {
        System.out.println(a);
        System.out.println(b);
        b++;
    }

    public void test(double c) {
        a += c;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        b++;
    }
}
```

```
public class Bar extends Foo {

    protected String c;

    public Bar(String c) {
        this.c = c;
        System.out.println(c);
    }

    public Bar(String a, double b) {
        super(a);
        c = "Summer";
        System.out.println(b);
    }

    public void test(int a) {
        test(a * 1.5);
        System.out.println(b);
        System.out.println(c);
    }

    public void test(String c) {
        b++;
        super.test(c);
    }

    public static void main(String[] args) {
        Bar a = new Bar("Final", 4.7);
        Foo b = new Bar("CS251");
        a.test(5);
        b.test("Exam");
    }
}
```

4. Write the answer in the blank provided.

   (a) 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)

   (a) _____

   (b) What is occurring when I have a static method in a child class that has the same name and parameter list as a method in the parent? (2)

   (b) _____

   (c) What is occurring when I have a non-static method in a child class that has the same name and parameter list as a method in the parent? (2)

   (c) _____

   (d) If I have a variable of type `Object`, an object of any type could be assigned to it. If I call the `toString` method on this variable, how does Java know which version of the method to execute? (2)

   (d) _____

   (e) If I declare a `String` variable but do not initialize it, what is its value? (2)

   (e) _____

   (f) What is the keyword used to call one constructor from another overloaded constructor in the same class? (2)

   (f) _____

   (g) What is the keyword used to call the parent class constructor from a constructor in the child class? (2)

   (g) _____

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

   (h) _____

   (i) I have a class `Demo` located in package `foo` and classes `DemoChild` and `NotChild` located in package `bar`. `DemoChild` extends `Demo` and `NotChild` does not. If I have a member variable inside `Demo` which is visible to methods in `DemoChild` but not to methods in `NotChild`, what access modifier have a used on the member variable? (2)

   (i) _____

   (j) Which abstract data type would be the best choice to associate student ID numbers with GPA values? (2)

   (j) _____

   (k) Name a method present in all classes that implement the `Collection` interface. (2)

   (k) _____

   (l) Name one of the integer data types in Java. (2)

   (l) _____

   (m) Name an unchecked exception from the Java standard libraries. (2)

   (m) _____

5. The following code successfully compiles and runs. What is its output? (10)

```
import java.util.*;

public class MyPoint implements Comparable<MyPoint> {

    private final int x;
    private final int y;

    public MyPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return "(" + x + ", " + y + ")";
    }

    public int compareTo(MyPoint p) {
        int diff = p.x - this.x;
        return diff != 0 ? diff : p.y - this.y;
    }

    public static void main(String[] args) {

        SortedSet<MyPoint> points = new TreeSet<>();
        points.add(new MyPoint(1,2));
        points.add(new MyPoint(2,3));
        points.add(new MyPoint(1,3));
        points.add(new MyPoint(1,2));
        points.add(new MyPoint(2,1));

        System.out.println(points);
    }
}
```

6. Think back to your Snake game. Answer the following questions about your GameManager implementation. (5)

   - What data structure(s) did you use to keep track of the snake and the walls?
   - How did you detect when the snake collided with a wall (or itself)?

7. Consider the following interface describing a gradebook that associates students (represented as Strings) with grade values. (We will make the assumption that student strings are unique.) (15)

```
public interface GradebookInterface {

    /** Update grade for student.
     * Add new gradebook entry if student is unknown.
     * Replace existing association if student already has grade.
     * @param s Student of interest.
     * @param g Grade for the student.
     */
    void update(String s, double g);

    /** Get grade for given student.
     * @param s Student of interest.
     * @return Grade for the student, or -1 if student is unknown.
     */
    double getGrade(String s);

    /** Get gradebook size.
     * @return Number of student entries in gradebook.
     */
    int getSize();

    /** Get average grade for all students in the gradebook.
     * @return Mean student grade, or -1 if no entries in gradebook.
     */
    double getAverage();
}
```

Write a small but complete class that implements this interface. You do not need to include Javadoc comments. Use a single private member variable of an appropriate type to hold the student grade information. *Solutions using more than one member variable will not receive full credit.*

8. The following program successfully compiles and runs. (10)

   What would be displayed when it is run? Draw a picture to illustrate.

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

public class GraphicProgram {

    public static class CustomPaintPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.BLACK);
            g.drawRect(25, 25, 150, 150);
            g.fillOval(50, 50, 100, 100);
        }
    }

    public static void createAndShowGUI() {
        JFrame frame = new JFrame("Final GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

        JPanel paintPanel = new CustomPaintPanel();
        paintPanel.setPreferredSize(new Dimension(200,200));

        frame.add(paintPanel, BorderLayout.CENTER);
        frame.add(buttonPanel, BorderLayout.PAGE_START);
        frame.pack();
        frame.setVisible(true);
    }

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