New York University
Introduction to Computer Science
Sample Exam

Instructions:

KEEP TEST BOOKLET CLOSED UNTIL YOU ARE INSTRUCTED TO BEGIN.

Omit one page from this exam. To select the page, cross out the entire page in one large ‘X’. If you do not select a question for omission, the last question of the exam will be omitted for you.

This exam is double sided (front and back)!

No calculators, notes, textbooks, or any other aids are allowed except writing utensils (pens, pencils, crayons, erasers, etc.) or other aids provided to you by the instructor. If you need extra scratch paper, please pick it up from the front of the class.

You should also be provided with an appendix that provides helpful documentation.

All answers must be submitted on (or attached to) this exam. All answers must be clearly legible.

Total Points Available: 100

Total Missed:

Final Score:
True or False (10 points):
Instructions: Circle either True or False based on the validity of the statement.

1. In object oriented programming, the superclass inherits properties from the subclass.
   true                              false

2. High level languages usually are easy to understand because they abstract hardware away from the programmer.
   true                              false

3. A String is a mutable data type.
   true                              false

4. In programming, a keyword is data that can be stored or pointed to (referred to) by variables.
   true                              false

5. An abstract class can have both concrete and abstract methods
   true                              false
Multiple Choice (10 points):
Instructions: Circle the letter of the best answer.

1. Which keyword is used to immediately end a function?
   A. else
   B. return
   C. break
   D. continue

2. Which of the following is similar to punctuation and spacing in a natural (spoken) language.
   A. semantics
   B. sequence
   C. syllable
   D. syntax

3. Data for this type of data is stored in the Heap memory (select all that apply):
   A. int
   B. int[]
   C. boolean
   D. double
   E. String

4. Given the following declaration, what type of data would integers[1] return?
   int[][] integers = {{1, 2, 3, 4}, {4, 3, 2, 1}}
   A. a multi-dimensional array
   B. a reference
   C. an integer
   D. an Integer object

5. What functionality do the 'try, catch, finally' keywords provide?
   A. getting user input/output
   B. function set up and tear down
   C. exception handling
   D. none of the above
**Short Answer (10 points):**

Instructions: In your own words, answer the questions as best as possible in one or two sentences.

1. Explain what a Java interface is.

2. What is method overriding?

3. List two AWT layout managers and describe how they differ (drawings are acceptable).

4. Explain what Java's `toString()` method is, what it does, and what would have one.

5. If you were designing a car racing game, what are 2 classes you might create and at least two attributes (data/functions) that each class contains?
Entomology - Study of Bugs (10 points):

Instructions: Find 3 bugs in the following program (there are more than three):
1) Specify the line number of the bug.
2) Check the box next to the type of error; either “Logic” for logic errors or “Other” for (syntax, type, etc.).
3) Write a short explanation or fix for the error.

Note: There can be more than one bug per line and assume all comments are accurate.

```java
public class Paint extends JPanel implements ActionListener, MouseMotionListener {
    private static final SPACING = 10; // spacing between recursive drawings
    JPanel canvas; // Drawing canvas needs to be accessible to all methods

    void Paint() {
        // This Panel is composed of a drawing canvas and a toolbar
        canvas = new JPanel();
        JPanel toolbar = new JPanel();

        // Set layout and add panels to this JPanel
        this.add(toolbar, BorderLayout.NORTH);
        add(canvas, BorderLayout.NORTH);

        // Add buttons to toolbar
        JButton recursiveButton = new JButton("Draw recursive graphics");
        JButton resetButton = new JButton("Reset");
        toolbar.add(resetButton);
        recursiveButton.addActionListener(this);
        canvas.addMouseListener(this);
    }

    // Draws squares recursively inside of one another on the canvas
    private void drawSquares(int x, int y, int size) {
        // Base case
        if (size == 0) return;
        // Draw outer square
        Graphics g = canvas.getGraphics();
        g.drawRect(x, y, size, size);
        // Recursive calls approaching the base case
        size += 2 * SPACING;
        drawSquares(x+SPACING, y+SPACING, size);
    }

    public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand() == "Reset") repaint();
        else {
            // Start drawing just inside the canvas
            drawSquares(SPACING, SPACING, this.getHeight - 2 * SPACING);
        }
    }
}
```

<table>
<thead>
<tr>
<th>Bug #</th>
<th>Line</th>
<th>Type of Error</th>
<th>Explanation or Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ ]</td>
<td>Other</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ ]</td>
<td>Other</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ ]</td>
<td>Other</td>
<td></td>
</tr>
</tbody>
</table>
OOP, I did it again... (10 points):

Instructions: What is the output if the following code is run?

```java
public class Output {
    public static void main(String[] args) {
        System.out.println("count: " + Container.numContainers);
        Box myBox = new Box();
        System.out.println("count: " + Container.numContainers);
        System.out.println(myBox);
        Box yourBox = new Box();
        yourBox.mass = 10;
        System.out.println("density: " + yourBox.getDensity());
        yourBox = myBox;
        System.out.println("count: " + Container.numContainers);
        myBox.mass = 100;
        System.out.println("density: " + yourBox.getDensity());
        Container container = myBox;
        System.out.println(myBox);
        System.out.println(yourBox);
        System.out.println(container);
    }
}

abstract class Container {
    double mass = 1;
    static int numContainers = 0;
    Container() {
        numContainers++;
    }
    public double getDensity() {
        return mass / getVolume();
    }
    abstract double getVolume();
}

class Box extends Container {
    double length = 1;
    double width = 1;
    double height = 1;
    Box() {
        super();
    }
    public double getVolume() {
        return length * width * height;
    }
    public String toString() {
        return "Box (" + length + ", " + width + ", " + height + ")";
    }
}

Answer:
Comment the Code (10 points):

Instructions: The following code was written, but the documentation was left out and function names were poorly choosen. Analyze the code to determine what each code block does and provide appropriate documentation. **Fill in ALL the missing comments.** Remember that Javadocs (** */) should explain WHAT the code does, and inline comments (//) should explain either how or why the code does what it does.

```java
/**
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 */
public static boolean so(int[] list) {
    //
    //
    for (int i = 0; i < list.length - 1; i++) {
        //
        //
        if (list[i] > list[i + 1])
            return false;
    }
    //
    //
    return true;
}
/**
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 * 
 */
public static int se(int[] list, int key) {
    //
    //
    for (int i = 0; i < list.length; i++) {
        //
        //
        if (list[i] == key) {
            return i;
        }
    }
    //
    //
    return -1;
}
```
Fill in the Code (10 points):

// A basic notepad application for reading from and writing to a file.

public class Notepad implements __________________________ {

    public static final __________________________ DEFAULT_FILE = "notepad.txt";
    JTextArea textArea;

    public static void main(String[] args) {
        new __________________________;
    }

    Notepad() {
        // Create window
        JFrame frame = __________________________;
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        // Create a scrollable textarea
        textArea = new JTextArea(10, 40);
        JScrollPane pane = new JScrollPane(textArea);
        frame.add(pane, BorderLayout.CENTER);
        // Create a simple toolbar and add it to the window
        JPanel toolbar = new JPanel();
        JButton saveButton = new JButton("Save");
        saveButton.addActionListener(this);
        toolbar.add(saveButton);
        saveButton.add(toolbar, BorderLayout.NORTH);
        try { // Read contents from default file
            Scanner fileScanner = new Scanner(new File(DEFAULT_FILE));
            // Read till EOF (end-of-file)
            String textInput = fileScanner.useDelimiter("\Z").next();
            _________________.close();
            // Update textArea with contents from the file
            textArea.setText(__________________________);
        } catch (_________________________ e) {
            System.err.println("Could not read from stored file.");
        }
        // Display window
        frame.setVisible(true);
    }

    @override // From the ActionListener interface
    public void actionPerformed(ActionEvent event) {
        ______________________________
        { // Save contents of the notepad to the file
            PrintWriter writer = new PrintWriter(new File(DEFAULT_FILE));
            writer._______________(textArea.getText());
            writer.close();
        } catch (FileNotFoundException e) {
            System.err.println("Error: Could not write to file.");
        }
    }
}
Write the Code (10 points):

Given the following code, write the equals(Object) method for this class so that it can check if another card is equivalent to itself:

```java
/**
 * A crude representation of a playing card. It extends a <code>JButton</code>
 * so it supports standard <code>ActionListener</code> type events and inherits
 * other JButton properties.
 *
 * Currently it adds functionality for storing a <code>Color</code> for the
 * front and the back of a the card for basic game playing. Assume JButton has
 * it's own equals(Object) method.
 */
public class Card extends JButton {
    private static final int borderSize = 2;
    boolean isfaceUp;
    Color frontColor;
    Color backColor;

    /**
     * Default constructor doesn't have any color
     */
    Card() {
        isfaceUp = false;
        frontColor = getBackground();
        backColor = getBackground();
        setText("");
        setBackground(backColor);
        setBorder(BorderFactory.createLineBorder(getForeground(), borderSize));
        setOpaque(true);
        setContentAreaFilled(true);
    }
}
```