Name: __________________________

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.
   Is a list of abstract methods that classes which are defined as that type will be required to implement.

2. What is method overriding?
   When a subclass has a method with the exact same specification as an ancestor class. This new method would take precedence if a method of that specification were called on an object of that subclass' type.

3. List two AWT layout managers and describe how they differ (drawings are acceptable).
   BorderLayout – Panels split between North/South/East/West, and a main Center panel.
   GridLayout – Organized using rows and columns.

4. Explain what Java's toString() method is, what it does, and what would have one.
   All objects have toString() methods since the base class java.lang.Object has a toString() method. By default, it prints out the hashCode for the object, but the method is usually overridden to provide a more human readable representation of the object. It returns this String representation of the object.

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?
   Car – speed, position, color, etc.
   Player – wins, losses, car, best_lap_time, etc.
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.addMouseMotionListener(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);
        }
    }
}
```

Line 1: Syntax: Missing imports
Line 2: Syntax: Missing 'int' keyword
Line 5: Syntax: Constructors have no return type – remove 'void'
Line 12: Logic: adding two items to the same part layout instead use BorderLayout.CENTER
Line 18: Logic: there is no action listener for the resetButton
Line 29: Syntax: Does not approach base case instead size -= 2 * SPACING
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:
count: 0
count: 1
Box (1.0, 1.0, 1.0)
density: 10.0
count: 2
density: 100.0
Box (1.0, 1.0, 1.0)
Box (1.0, 1.0, 1.0)
Box (1.0, 1.0, 1.0)
Comment the Code (10 points):

Instructions: The following code was written, but the documentation was left out and function names were poorly chosen. 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.

```
/**
 * Checks if <code>list</code> is ordered from smallest to largest.
 * @param list array to check
 * @return true if array is ordered from smallest to largest; otherwise false
 */
public static boolean so(int[] list) {
    // Loop through all adjacent elements one pair at a time (should run length-1 times)
    for (int i = 0; i < list.length - 1; i++) {
        // Compares current index to next index and reports back if out of order
        if (list[i] > list[i + 1])
            return false;
    }
    // If every neighboring pair is checked, it's sorted.
    return true;
}

/**
 * Search <code>list</code> for the value in <code>key</code>. 
 * @param list the array to searched
 * @param key the value to search for
 * @return the index of <code>key</code> in <code>list</code>.  Returns -1 if the value is not found in the array.
 */
public static int se(int[] list, int key) {
    // For every index
    for (int i = 0; i < list.length; i++) {
        // See if key was found at current index
        if (list[i] == key)
            return i;
    }
    // Key not found
    return -1;
}
```
Fill in the Code (10 points):

// A basic notepad application for reading from and writing to a file.
public class Notepad implements ActionListener {
    public static final String DEFAULT_FILE = "notepad.txt";
    JTextArea textArea;
    public static void main(String[] args) {
        new Notepad();
    }

    Notepad() {
        // Create window
        JFrame frame = new JFrame();
        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();
            fileScanner.close();

            // Update textArea with contents from the file
            textArea.setText(textInput);
        } catch (IOException 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) {
        try { // Save contents of the notepad to the file
            PrintWriter writer = new PrintWriter(new File(DEFAULT_FILE));

            writer.println(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.
 */

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);
    }

    @Override
    public boolean equals(Object o) {
        if (o == null)
            return false;
        if (! (o instanceof Card))
            return false;
        if (isfaceUp != o.isfaceUp)
            return false;
        if (! frontColor.equals(o.frontColor))
            return false;
        if (! backColor.equals(o.backColor))
            return false;
        return super.equals(o);
    }
}
```