New York University
Introduction to Computer Science
Midterm2 - Exam G
Fall 2014

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 page, 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 or other aids provided to you by the instructor. Extra scratch paper is available from the front of the class.

All answers must be submitted on (or attached to) this exam sheet. All answers must be clearly legible.
True or False (10 points):

Instructions: Read the entire statement. Circle whether this statement is either True or False.

1. All classes that are written implicitly extend java.lang.Object and therefore inherit its toString() method.
   
   TRUE
   FALSE

2. A String is a mutable data type.

   TRUE
   FALSE

3. An enumerator (enum) is a data type that can store one of a limited set of values

   TRUE
   FALSE

4. Instance fields (sometimes called object variables) are unique to each instance of an object.

   TRUE
   FALSE

5. A class can have multiple subclasses.

   TRUE
   FALSE
Multiple Choice (10 points):
Instructions: Circle the best answer(s).

1. _____________ is invoked to create an object.
   A. The main method
   B. A constructor
   C. A method with a return type
   D. A method with the void return type

2. Given the code which stores RGB values of pixels in a 2-D space:
   int[][] pixels = { { {0,0,0}, {1,1,1}, {2,2,2} },
                     { {3,3,3}, {4,4,4}, {5,5,5} },
                     { {6,6,6}, {7,7,7}, {8,8,8} } }; 
   Which of the following would store the RGB values {3,3,3}?
   A. pixels[3]
   B. pixels[4]
   C. pixels[0][1]
   D. pixels[1][0]
   E. pixels[2][1]
   F. None of the above

3. Polymorphism means?
   A. data fields are defined as private
   B. a class can store a reference to another class
   C. that a variable of a supertype can refer to a subtype object
   D. that a class can extend another class

4. This keyword means that the method or variable can be accessed as a class method or
   variable (as opposed to an instance method/variable).
   A. public
   B. private
   C. static
   D. extends

5. Which of the following statements are true (circle all that apply)?
   A. You can use super to invoke a super class constructor
   B. You can use super to access parent class variables
   C. You can use super to change your object's data type
   D. You can use super to invoke a parent class method
Short Answer (10 points):

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

1. What is the relationship between a subclass and a superclass?

Subclass has an “is a” type of relationship meaning that instances of the subclass are also instances of the super type. The subclass inherits all the data attributes of a superclass (data/methods). A subclass usually is a special type of the superclass or a more specific type of the super class. A subclass usually adds additional functionality or data in addition to the inherited attributes.

2. What is the difference between an object's == and .equals() method.

== checks if the references to two objects are the same.
.equals() checks if the two objects are equivalent.

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

toString() is an instance method which returns a String representation of an object. It is a method in the java.lang.Object class and therefore every object will have one.

4. In OOP, explain what a "Class" is.

A class is the template or blueprint used to create objects. Classes can have static data which are shared by all instances of the class. Objects are created based on the class specification and each object is it's own instance of that class. Each object has it's own variables and data associated with that specific instance.

5. Explain the difference between a shallow and deep copy.

A shallow copy copies just the reference, while a deep copy copies the data. When making a shallow copy, changes in one object or array will also affect the copy since they both are using the same data. A deep copy duplicates all the data so changes to one copy will not affect changes to the other.
Entomology - Study of Bugs (10 points):

Instructions: Find 3 bugs in the following program (there are more than 3 and there can be more than 1 per line):
1) Specify the line number of the bug.
2) Check the box next to the type of error.
3) Write a short explanation or fix for the error.

```java
public static int linearSearch(int[] array, int value) {
    for (int i = 0; i <= array.length; i++) {
        if (array[i] == value)
            return i;
    }
    return -1;
}
```

```java
public static int binarySearch(int[] array, int value) {
    int lowIndex = 0;
    int highIndex = array.length;
    // Run until found or search space exhausted
    if (lowIndex <= highIndex) {
        int midIndex = (lowIndex + highIndex) / 2;
        if (array[midIndex] == value) { // element found
            return array[midIndex];
        } else if (array[midIndex] > value) { // search the high side
            lowIndex = midIndex + 1;
        } else { // search the low side
            highIndex = midIndex - 1;
        }
    }
    return -1;
}
```

<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>[ ] Syntax</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Runtime</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>[ ] Syntax</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Runtime</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>[ ] Syntax</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ ] Runtime</td>
<td></td>
</tr>
</tbody>
</table>
What is the Output (10 points):

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

```java
public class InsertionSort {
    public static void main(String[] args) {
        int[] array = { 1, 7, 3, 9, 5 };  
        for (int i = 1; i < array.length; i++) {
            int j = i;
            while ((j > 0) && (array[j - 1] > array[j])) {
                swap(array, j, j-1);
                j = j - 1;
            }
            for (int value: array)
                System.out.print(value + " ");
            System.out.println();
        }
    }
    
    /**
    * Takes and array and swaps values from <code>index1</code> and <code>index2</code>
    * @param array array of values
    * @param index1 one of the indexes to swap the contents of
    * @param index2 the second index to swap the contents of
    */
    public static void swap(int[] array, int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }
}
```

Answer:

1 7 3 9 5
1 3 7 9 5
1 3 7 9 5
1 3 5 7 9
Comment the Code (10 points):

```java
import processing.core.PApplet;

/**
 * Creates a window that draws a circle and squares of decreasing size wherever the mouse is clicked.
 */
public class Shapes extends PApplet {
    private static final int SIZE = 100;
    private static final int SPACING = 10;
    private static final int NUM_SQUARES = SIZE / SPACING;

    /**
     * Creates graphics window to interact with the user
     */
    public void setup() {
        size(600, 500);
        background(0);
        noFill();
    }

    /**
     * Updates the graphics window based on user mouse clicks
     */
    public void draw() {
        // Draws something wherever the mouse is pressed.
        if (mousePressed == true) { // mousePressed is a boolean value provided by Papplet
            int xPos = mouseX;      // mouseX is the x location in the window of where the mouse is
            int yPos = mouseY;      // mouseY is the y location in the window of where the mouse is
            d(xPos, yPos);
        }
    }

    /**
     * Draws a circle and series of squares of decreasing size
     * @param x location to center the shapes
     * @param y location to center the shapes
     */
    void d(int x, int y) {
        int size = SIZE;

        // Draw circle
        ellipse(x, y, size, size);

        // Draw squares of decreasing size
        for (int i = 0; i < NUM_SQUARES; i++) {
            rect(x - size/2, y - size/2, size, size);
            size -= SPACING;
        }
    }
}
```
Fill in the Code (10 points):

Instructions: Fill in the missing code

```java
import processing.core.PApplet;
/**
 * A Ball is based on a Particle object (a particle only has x/y position and speed attributes) but
 * adds some additional features fitting for a representation of a ball.
 */
public class Ball
    extends Particle {

    private static final int DEFAULT_SIZE = 10; // Default to this size if non given
    private int size;

    /**
     * Constructor with arguments
     * @param x x location for this ball
     * @param y y location for this ball
     * @param size diameter (in pixels) of this ball
     */
    Ball(int x, int y, int size) {
        super(x, y) // Allows the parent class to do most of the initialization
        this.size = size;
    }

    // Alternate argument constructor, if no size the ball will use a default value
    Ball(int x, int y) {
        // Constructor chaining: calls a different constructor to initialize the data
        this(x, y, DEFAULT_SIZE);
    }

    // Copy constructor
    Ball(Ball otherBall) {
        super(otherBall);
        this.size = otherBall.size;
    }

    public String toString() {
        return super.toString() + "is a ball with size " + size;
    }

    public boolean equals(Ball otherBall) {
        if (!super.equals(ball))
            return false;
        if (this.size != otherBall.size)
            return false;
        return true;
    }

    // Draws this object to the screen using the processing applet passed in.
    void draw(PApplet applet) {
        applet.ellipse(xPos, yPos, size, size);
    }
}
```
Write the Code (10 points):

Write the Java code for a class that has the following data and methods associated with it (format is UML which indicates a variable or method name then a colon then it's type or return type. Comments/Javadocs are not required but can potentially earn you partial credit.

Potion:
--- public static data ------------
VALID_INGREDIENTS: char[]   // Valid ingredients are the characters 'a', 'b', 'c', 'd', 'w'
--- public static methods ------------
isValid(c: char): boolean   // Checks if 'c' is in list of VALID_INGREDIENTS
--- private data -----------------
maxSize: int                // Maximum number of ingredients in a potion
ingredients: String         // The ingredients of a potion (you may use a char[] if desired)
--- public methods --------------
Potion(size: int)           // Initialize empty ingredients String and set max size to size
addIngredient(c: char)      // Adds char c to end of ingredients String only if it is a
                            // valid ingredient and the maximum number of ingredients hasn't
                            // already been reached.
toString(): String          // Gives back the list of ingredients in the potion

```
public class Potion {
    public static char[] VALID_INGREDIENTS = { 'a', 'b', 'c', 'd', 'w' };

    public static boolean isValid(char c) {
        for (char ingredient : VALID_INGREDIENTS) {
            if (c == ingredient) {
                return true
            }
        }
        return false;
    }

    private int maxSize;
    private String ingredients;

    Potion(int size) {
        maxSize = size;
        ingredients = "w";
    }

    public void addIngredient(char c) {
        if ((isValid(c) && (ingredients.length() < maxSize)) {
            ingredients += c;
        }
    }

    public String toString() {
        return ingredients;
    }
}
```
Write the Code (10 points):

In Java code, write a method that satisfies the following Javadoc requirements. Comments/Javadocs are not required but can potentially earn you partial credit.

```java
/**
 * Checks to see if either diagonal of a given matrix (a 2-d array) has all the same values. If the
 * matrix has different number of columns than rows, it will also return false.
 * Examples:
 *   These would return true:          |   These would return false:
 *   1 0 0   0 0 2   3 3 3  |   1 2 3   1 1 1 1   1 0 0 1
 *   0 1 0   0 2 2   3 3 3  |   1 2 3   1 1 1 1   0 0 0 0
 *   0 0 1   0 2 2   3 3 3  |   1 2 3   1 1 1 1   0 0 0 0
 *   2 2 2   3 3 3  |   1 0 0 1
 *
 * @param matrix (a 2-d array of integers)
 * @return true if either diagonal has all matching values; otherwise false
 */
public static boolean matchingDiagonals(int[][] matrix) {
    // Check for square matrix:
    if (matrix.length != matrix[0].length) {
        return false;
    }

    // Check diagonal going from top-left to bottom-right
    boolean diagonal1_matches = true
    for (int i = 0; i < matrix.length; i++) {
        if (matrix[i][i] != matrix[0][0]) {
            diagonal1_matches = false;
            break;
        }
    }

    // Check diagonal going from the bottom-left to the top-right
    boolean diagonal2_matches = true
    for (int i = 0; i < matrix.length; i++) {
        if (matrix[matrix.length-i-1][i] != matrix[matrix.length-1][0]) {
            diagonal2_matches = false;
            break;
        }
    }

    // Return true if either diagonal has all matches
    if (diagonal1_matches || diagonal2_matches) {
        return true;
    } else {
        return false;
    }
}
```
Write the Code (10 points):

In Java code, write a method that satisfies the following Javadoc requirement. Comments/Javadocs are not required but can potentially earn you partial credit.

/**
 * Shuffles an array or an ArrayList (your choice) of objects into random positions (in place).
 * @param array of <code>Object</code>s to shuffle into random position (in-place)
 */

// Using an Array of Objects
public static void shuffle(Object[] array) {
    for (int i = 0; i < array.length; i++) {
        int randomIndex = (int)(Math.random() * array.length);
        swap(array, i, randomIndex);
    }
}

public static void swap(Object[] array, int index1, int index2) {
    Object temp = array[index1];
    array[index1] = array[index2];
    array[index2] = temp;
}

// Using an ArrayList of Objects
public static void shuffle(ArrayList<Object> objects) {
    for (int i = 0; i < objects.size(); i++) {
        int randomIndex = (int)(Math.random() * array.length);
        swap(objects, i, randomIndex);
    }
}

public static void swap(ArrayList<Object> objects, int index1, int index2) {
    Object temp = objects.get(index1);
    objects.set(index1, objects.get(index2));
    objects.set(index2, temp);
}
### ArrayList methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>add(E e)</td>
</tr>
<tr>
<td></td>
<td>Appends the specified element to the end of this list.</td>
</tr>
<tr>
<td>void</td>
<td>add(int index, E element)</td>
</tr>
<tr>
<td></td>
<td>Inserts the specified element at the specified position in this list.</td>
</tr>
<tr>
<td>boolean</td>
<td>addAll(Collection&lt;? extends E&gt; c)</td>
</tr>
<tr>
<td></td>
<td>Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.</td>
</tr>
<tr>
<td>boolean</td>
<td>addAll(int index, Collection&lt;? extends E&gt; c)</td>
</tr>
<tr>
<td></td>
<td>Inserts all of the elements in the specified collection into this list, starting at the specified position.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
</tr>
<tr>
<td></td>
<td>Removes all of the elements from this list.</td>
</tr>
<tr>
<td>Object</td>
<td>clone()</td>
</tr>
<tr>
<td></td>
<td>Returns a shallow copy of this ArrayList instance.</td>
</tr>
<tr>
<td>boolean</td>
<td>contains(Object o)</td>
</tr>
<tr>
<td></td>
<td>Returns true if this list contains the specified element.</td>
</tr>
<tr>
<td>void</td>
<td>ensureCapacity(int minCapacity)</td>
</tr>
<tr>
<td></td>
<td>Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.</td>
</tr>
<tr>
<td>E</td>
<td>get(int index)</td>
</tr>
<tr>
<td></td>
<td>Returns the element at the specified position in this list.</td>
</tr>
<tr>
<td>int</td>
<td>indexOf(Object o)</td>
</tr>
<tr>
<td></td>
<td>Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.</td>
</tr>
<tr>
<td>boolean</td>
<td>isEmpty()</td>
</tr>
<tr>
<td></td>
<td>Returns true if this list contains no elements.</td>
</tr>
<tr>
<td>Iterator&lt;E&gt;</td>
<td>iterator()</td>
</tr>
<tr>
<td></td>
<td>Returns an iterator over the elements in this list in proper sequence.</td>
</tr>
<tr>
<td>int</td>
<td>lastIndexOf(Object o)</td>
</tr>
<tr>
<td></td>
<td>Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.</td>
</tr>
<tr>
<td>ListIterator&lt;E&gt;</td>
<td>listIterator()</td>
</tr>
<tr>
<td></td>
<td>Returns a list iterator over the elements in this list (in proper sequence).</td>
</tr>
<tr>
<td>ListIterator&lt;E&gt;</td>
<td>listIterator(int index)</td>
</tr>
<tr>
<td></td>
<td>Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.</td>
</tr>
<tr>
<td>E</td>
<td>remove(int index)</td>
</tr>
<tr>
<td></td>
<td>Removes the element at the specified position in this list.</td>
</tr>
<tr>
<td>boolean</td>
<td>remove(Object o)</td>
</tr>
<tr>
<td></td>
<td>Returns true if this list contains the specified element.</td>
</tr>
<tr>
<td>boolean</td>
<td>removeAll(Collection&lt;?&gt; c)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>protected void</td>
<td>removeRange(int fromIndex, int toIndex)</td>
</tr>
<tr>
<td>boolean</td>
<td>retainAll(Collection&lt;?&gt; c)</td>
</tr>
<tr>
<td>E</td>
<td>set(int index, E element)</td>
</tr>
<tr>
<td>int</td>
<td>size()</td>
</tr>
<tr>
<td>List&lt;E&gt;</td>
<td>subList(int fromIndex, int toIndex)</td>
</tr>
<tr>
<td>Object[]</td>
<td>toArray()</td>
</tr>
<tr>
<td>&lt;T&gt; T[]</td>
<td>toArray(T[] a)</td>
</tr>
<tr>
<td>void</td>
<td>trimToSize()</td>
</tr>
</tbody>
</table>