

# \*\*\* CS 106A FINAL EXAM SYNTAX REFERENCE \*\*\*

## Math (A&S 5.1)

```
double d = Math.pow(2, 5); // 32.0
Math.abs(n), Math.ceil(n), Math.floor(n), Math.log(n), Math.log10(n),
Math.max(a, b), Math.min(a, b), Math.pow(b, e), Math.round(n), Math.sqrt(n),
Math.sin(r), Math.cos(r), Math.tan(r), Math.toDegrees(r), Math.toRadians(d)
```

## RandomGenerator (A&S 6.1)

```
RandomGenerator rg = RandomGenerator.getInstance();
rg.nextBoolean() returns a random true/false result;
rg.nextBoolean(probability) pass an optional probability from 0.0 - 1.0, or default to 0.5
rg.nextColor() a randomly chosen Color object
rg.nextDouble(min, max) returns a random real number between min and max, inclusive
rg.nextInt(min, max) returns a random integer between min and max, inclusive
```

## String (A&S Ch. 8)

```
String s = "hello";
s.charAt(i) the character in this String at a given index
s.contains(str) true if this String contains the other's characters inside it
s.endsWith(str) true if this String ends with the other's characters
s.equals(str) true if this String is the same as str
s.equalsIgnoreCase(str) true if this String is the same as str, ignoring capitalization
s.indexOf(str) first index in this String where given String begins (-1 if not found)
s.lastIndexOf(str) last index in this String where given String begins (-1 if not found)
s.length() number of characters in this String
s.replace(s1, s2) a new string with all occurrences of s1 changed to s2
s.startsWith(str) true if this String begins with the other's characters
s.substring(i, j) characters in this String from index i (inclusive) to j (exclusive)
s.toLowerCase() a new String with all lowercase or uppercase letters
s.toUpperCase()
```

## Character/char (A&S Ch. 8)

```
char c = Character.toUpperCase(s.charAt(i));
```

Character.isDigit( <i>ch</i> ), .isLetter( <i>ch</i> ), .isLowerCase( <i>ch</i> ), .isUpperCase( <i>ch</i> ), .isWhitespace( <i>ch</i> )	methods that accept a <b>char</b> and return <b>boolean</b> values of <b>true</b> or <b>false</b> to indicate whether the character is of the given type
Character.toLowerCase( <i>ch</i> ), .toUpperCase( <i>ch</i> )	accepts a character and returns lower/uppercase version of it

## Scanner

```
Scanner input = new Scanner(new File("filename")); // scan an input file
Scanner tokens = new Scanner(string); // scan a string
```

sc.next(),	sc.nextLine()	read/return the next token (word) or entire line of input as a string
sc.nextInt(),	sc.nextDouble()	read/return the next token of input as an int or double
sc.hasNext(),	sc.hasNextLine(),	ask about whether a next token/line exists, or
sc.hasNextInt(),	sc.hasNextDouble()	what type it is, without reading it
sc.close()		closes the scanner

## Program, ConsoleProgram, GraphicsProgram

```
public class Name extends ProgramType { ... }
init() executes before window appears; use to set up graphical components
run() executes after window appears; use for animation loops, file loading, etc.
```

## ConsoleProgram

```
public class Name extends ConsoleProgram { ... }
readInt("prompt"), Prompts/reprompts for a valid int or double, and returns it
readDouble("prompt")
readLine("prompt"); Prompts/reprompts for a valid String, and returns it
```

<code>readBoolean("prompt", "yesString", "noString");</code>	Prompts/reprompts for either <b>yesString</b> or <b>noString</b> (case-insensitive). Returns <b>true</b> if they enter <b>yesString</b> , <b>false</b> if they enter <b>noString</b> .
<code>promptUserForFile("prompt", "directory");</code>	Prompts for a filename, re-prompting until input is a file that exists in the given directory. Returns the full file path (" <b>directory/filename</b> ").
<code>println("text");</code>	Prints the given text to the console, followed by a newline ("n").
<code>print("text");</code>	Prints the given text to the console.

## GraphicsProgram

```
public class Name extends GraphicsProgram { ... }
```

<code>add(shape);</code>	displays the given graphical shape/object in the window
<code>add(shape, x, y);</code>	displays the given graphical shape/object in the window at <b>x, y</b>
<code>getElementAt(x, y)</code>	returns graphical object at the given x/y position, if any (else <b>null</b> )
<code>getHeight(), getWidth()</code>	the height and width of the graphical window, in pixels
<code>pause(ms);</code>	halts for the given # of milliseconds
<code>remove(shape);</code>	removes the graphical shape/object from window so it will not be seen
<code>setCanvasSize(w, h);</code>	sets canvas's onscreen size
<code>setBackground(color);</code>	sets canvas background color

## Graphical Objects (A&S Ch. 9)

```
GRect rect = new GRect(10, 20, 50, 70);
```

<code>new GLabel("text", x, y)</code>	text with bottom-left (baseline) at (x, y) <b>Note:</b> x, y are optional
<code>new GLine(x1, y1, x2, y2)</code>	line between points (x1, y1), (x2, y2)
<code>new GOval(x, y, w, h)</code>	largest oval that fits in a box of size w * h with top-left at (x, y) <b>Note:</b> x, y are optional
<code>new GRect(x, y, w, h)</code>	rectangle of size w * h with top-left at (x, y) <b>Note:</b> x, y are optional
<code>obj.getColor(), obj.getFillColor()</code>	returns the color used to color the shape outline or interior
<code>obj.getX(), obj.getY(), obj.getWidth(), obj.getHeight()</code>	returns the left x, top y coordinates, width, and height of the shape
<code>obj.move(dx, dy);</code>	adjusts location by the given amount
<code>obj.setBackground(Color);</code>	sets overall window's background color
<code>obj.setFilled(boolea);</code>	whether to fill the shape with color
<code>obj.setFillColor(Color);</code>	what color to fill the shape with
<code>obj.setColor(Color);</code>	what color to outline the shape with
<code>obj.setLocation(x, y);</code>	change the object's x/y position
<code>obj.setSize(w, h);</code>	change the objects width*height size
<code>new GImage("filename", x, y)</code>	image from the given file, drawn at (x, y) <b>Note:</b> x,y are optional
<code>new GImage(pixelArray)</code>	image from the given 2D array of <b>int</b> pixels
<code>gimage.getPixelArray(), setPixelArray(a)</code>	return/set 2D array of <b>ints</b> representing pixels of the image
<code>GImage.getRed(px), getGreen(px), getBlue(px)</code>	returns the individual red/green/blue components of a given <b>int</b> pixel
<code>GImage.createRGBPixel(r, g, b)</code>	creates and returns an <b>int</b> pixel with the given r/g/b values
<code>GImage.createRGBPixel(r, g, b, a)</code>	creates and returns an <b>int</b> pixel with the given r/g/b/alpha values

## Colors

```
rect.setColor(Color.BLUE);
```

```
Color.BLACK, BLUE, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW
Color name = new Color(r, g, b); // red, green, blue from 0-255
```

## Mouse Events (A&S Ch. 10)

```
public void eventMethodName(MouseEvent event) { ...
```

```
events: mouseMoved, mouseDragged, mousePressed, mouseReleased, mouseClicked, mouseEntered, mouseExited
```

<code>e.getX(), e.getY()</code>	the x or y-coordinate of mouse cursor in the window
---------------------------------	---

## Array (A&S Ch. 11)

```
int[] arr = new int[5]; int[][] pixels = new int[5][2];
```

<code>new type[length]</code>	creates a new 1D array of the given type and length
<code>new type[rows][cols]</code>	creates a new 2D array of the given type and number of rows and cols

<code>arr[i], arr[i][j], ...</code>	returns the element at index <i>i</i> , index ( <i>i,j</i> ), etc.
<code>arr.length</code>	returns the length of the array
<code>Arrays.toString(arr)</code>	returns a string representing the array, such as <b><i>“[10, 30, -25, 17]”</i></b>
<code>Arrays.sort(arr)</code>	sorts the elements in place (no return value)
<code>Arrays.equals(arr1, arr2)</code>	returns true if the arrays contain the same elements in the same order
<code>Arrays.fill(arr, value)</code>	sets every element to the given value
<code>Arrays.deepToString(arr)</code>	returns a string representing the multidimensional array, such as <b><i>“[[0, 1, 2], [1, 2, 3], [2, 3, 4]]”</i></b>
<code>Arrays.deepEquals(arr1, arr2)</code>	returns true if the multidimensional arrays contain the same elements in the same order.

## ArrayList (11.8)

## HashMap (13.2)

`ArrayList<Integer> list = new ArrayList<>();`

<code>L.add(value);</code> <code>L.add(index, val);</code>	append to end of list; or insert at index, shifting right
<code>L.clear();</code>	removes all elements
<code>L.contains(value)</code>	true if value is in the list
<code>L.equals(L2)</code>	true if same elements
<code>L.get(index)</code>	returns value at given index
<code>L.indexOf(value)</code> <code>L.lastIndexOf(val)</code>	first/last index where given value is found (or -1 if not found)
<code>L.isEmpty()</code>	true if the list has no elements
<code>L.remove(index);</code>	removes value at given index, shifting subsequent values left
<code>L.remove(val);</code>	removes first occurrence of value
<code>L.set(index, val);</code>	replaces value at given index
<code>L.size()</code>	number of elements in the list
<code>L.toString()</code>	string representation of list such as <b><i>“[10, -2, 43]”</i></b>

`HashMap<String, Double> map = new HashMap<>();`

<code>M.put(key, value);</code>	adds a pair between the given key and value, replacing any old pair for that key
<code>M.clear();</code>	removes all elements
<code>M.containsKey(key)</code>	returns true if the given key is a key of a pair in this map
<code>M.equals(map2)</code>	true if same key/value pairs
<code>M.get(key)</code>	returns value paired with key, or null
<code>M.keySet()</code>	a collection of all keys in the map
<code>M.isEmpty()</code>	true if the map contains no pairs
<code>M.remove(key);</code>	removes pair for the given key, if there is one; does nothing if not
<code>M.values()</code>	collection of all values in map
<code>M.size()</code>	returns number of pairs in map
<code>M.toString()</code>	returns a string representation such as <b><i>“{a=b, c=d, e=f}”</i></b>

// **collection** is a HashMap key/value set, array, or ArrayList  
for (**type name** : **collection**) { ...

## Interactors (A&S 10.5-10.6)

`JButton button = new JButton(“Click me!”);`

<code>new JButton(“text”)</code>	button displaying the given text
<code>addActionListeners()</code>	sets up program to hear action events on all added buttons
<code>new JLabel(“text”)</code>	label displaying the given text
<code>new JTextField(width)</code>	text field with the given width (in characters)
<code>textField.addActionListener(this)</code>	sets up program to hear an action event when ENTER key typed
<code>.getText(), .setText(text)</code>	get/set the text being displayed in the button/label/text field
<code>add(component, region)</code>	adds the given interactor in the given window region (e.g. SOUTH or EAST)

`public void actionPerformed(ActionEvent event) { ...`

<code>e.getActionCommand()</code>	a string representing the event that occurred (e.g. text of clicked button)
<code>e.getSource()</code>	the component that caused the event