# \*\*\* 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()  rg.nextBoolean(probability) | returns a random true/false result; 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()  s.toUpperCase() | a new String with all lowercase or uppercase letters |

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

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

|  |  |
| --- | --- |
| Character.isDigit(ch), .isLetter(ch),  .isLowerCase(ch), .isUpperCase(ch),  .isWhitespace(ch) | methods that accept a char and return boolean values of true or false to indicate whether the character is of the given type |
| Character.toLowerCase(ch),  .toUpperCase(ch) | 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(),  sc.hasNextInt(), sc.hasNextDouble() | ask about whether a next token/line exists, or  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"***), readDouble("***prompt"***) | Prompts/reprompts for a valid int or double, and returns it |
| readLine("***prompt"***); | Prompts/reprompts for a valid String, and returns it |
| readBoolean("***prompt"***, "***yesString"***, "***noString"***); | Prompts/reprompts for either ***yesString*** or ***noString*** (case-insensitive). Returns **true** if they enter ***yesString***, **false** if they enter ***noString***. |
| promptUserForFile(***"prompt"***, ***"directory"***); | Prompts for a filename, re-prompting until input is a file that exists in the given directory. Returns the full file path (***“directory/filename”***). |
| println("***text"***); | Prints the given text to the console, followed by a newline (‘\n’). |
| print("***text"***); | Prints the given text to the console. |

### GraphicsProgram

public class Name extends GraphicsProgram { ... }

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

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

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

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

|  |  |
| --- | --- |
| e.getX(), e.getY() | 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];

|  |  |
| --- | --- |
| new type[length] | creates a new 1D array of the given type and length |
| new type[rows][cols] | creates a new 2D array of the given type and number of rows and cols |
| arr[i], arr[i][j], ... | returns the element at index i, index (i,j), etc. |
| arr.length | returns the length of the array |
| Arrays.toString(arr) | returns a string representing the array, such as ***“[10, 30, -25, 17]”*** |
| Arrays.sort(arr) | sorts the elements in place (no return value) |
| Arrays.equals(arr1, arr2) | returns true if the arrays contain the same elements in the same order |
| Arrays.fill(arr, value) | sets every element to the given value |
| Arrays.deepToString(arr) | returns a string representing the multidimensional array, such as  ***“[[0, 1, 2], [1, 2, 3], [2, 3, 4]]”*** |
| Arrays.deepEquals(arr1, arr2) | 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<>(); HashMap<String, Double> map = new HashMap<>();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L.add(value); L.add(index, val); | append to end of list; or insert at index, shifting right |  | M.put(key, value); | adds a pair between the given key and value, replacing any old pair for that key |
| L.clear(); | removes all elements |  | M.clear(); | removes all elements |
| L.contains(value) | true if value is in the list |  | M.containsKey(key) | returns true if the given key is a key of a pair in this map |
| L.equals(L2) | true if same elements |  | M.equals(map2) | true if same key/value pairs |
| L.get(index) | returns value at given index |  | M.get(key) | returns value paired with key, or null |
| L.indexOf(value) L.lastIndexOf(val) | first/last index where given value is found (or -1 if not found) |  | M.keySet() | a collection of all keys in the map |
| L.isEmpty() | true if the list has no elements |  | M.isEmpty() | true if the map contains no pairs |
| L.remove(index); | removes value at given index, shifting subsequent values left |  | M.remove(key); | removes pair for the given key, if there is one; does nothing if not |
| L.remove(val); | removes first occurrence of value |  |  |  |
| L.set(index, val); | replaces value at given index |  | M.values() | collection of all values in map |
| L.size() | number of elements in the list |  | M.size() | returns number of pairs in map |
| L.toString() | string representation of list such as "[10, -2, 43]" |  | M.toString() | returns a string representation such as "{a=b, c=d, e=f}" |

// ***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!”);

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

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

|  |  |
| --- | --- |
| e.getActionCommand() | a string representing the event that occurred (e.g. text of clicked button) |
| e.getSource() | the component that caused the event |