# \*\*\* CS 106A MIDTERM SYNTAX REFERENCE \*\*\*

*This document lists some of the common methods and syntax that you will use on the exam. For more, consult your textbook. (v1.3.2)*

### Karel the Robot (Karel reader Ch. 1-6)

public class Name extends SuperKarel { ... }

|  |  |
| --- | --- |
| turnLeft(); turnRight(); turnAround(); | rotates Karel 90º counter-clockwise, clockwise, or 180º |
| move(); | moves Karel forward in current direction by one square |
| pickBeeper(); | picks up a beeper if present on Karel's corner; else error |
| putBeeper(); | places a beeper, if present in beeper bag; else error |
| frontIsClear(), frontIsBlocked() | Is there a wall in front of Karel? |
| leftIsClear(), leftIsBlocked() | Is there a wall to Karel's left (counter-clockwise)? |
| rightIsClear(), rightIsBlocked() | Is there a wall to Karel's right (clockwise)? |
| beepersPresent(), noBeepersPresent() | Are there any beepers on Karel's current corner? |
| beepersInBag(), noBeepersInBag() | Are there any beepers in Karel's beeper bag? |
| facingNorth(), notFacingNorth(), facingEast(), notFacingEast(), facingSouth(), notFacingSouth(), facingWest(), notFacingWest() | Is Karel facing north, south, east, or west? |

### 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 |

### 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 GImage("filename", x, y) | image from the given file, drawn at (x, y) |
| new GLabel("text", x, y) | text with bottom-left at (x, y) |
| 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) |
| new GRect(x, y, w, h) | rectangle of size w \* h with top-left at (x, y) |
| 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 |

### 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 |