ECS100 Java Documentation

Brief and partial documentation of UI and other Java classes and methods

---

UI **class**:

```
  // Text output
 public void clearText()                  // Clears the text pane
 public void print(anything val)          // Prints val with no newline
 public void println(anything val)        // Prints val and newline
 public void println()                    // Prints a newline
 public void printf(String format, ...)   // Prints the format string, inserting remaining arguments at the %'s:
                                          //   %s for Strings.
                                          //   %d for ints, (%3d: use at least 3 characters),
                                          //   %6.2f for 2dp doubles, using at least 6 characters
                                          //   %n is a line separator
  // Text input
 public String askString(String question)          // ask the user for a line of text
 public int askInt(String question)                // ask the user for an integer
 public double askDouble(String question)          // ask the user for a number
 public String askToken(String question)           // ask the user for a single token
 public boolean askBoolean(String question)        // ask the user for a yes/no or true/false value
 public ArrayList<Double> askNumbers(String question)  // ask the user for a list of numbers
 public ArrayList<String> askStrings(String question)  // ask the user for a list of Strings

  // Graphics output
 public void clearGraphics()                       // Clears the graphics pane
 public void setColor(Color c)                     // Change the colour for later drawing commands
 public void setFontSize(double s)                 // Change the size of the Font for later drawString commands
 public void setLineWidth(double width)            // Change the width of lines for later drawing commands

  // draw outlines of shapes
 public void drawLine(double x1, double y1, double x2, double y2)  // Line from (x1,y1) to (x2,y2)
 public void drawRect(double left, double top, double width, double height)
 public void drawOval(double left, double top, double width, double height)
 public void drawArc(double left, double top, double width, double height, double startAngle, double arcAngle)
 public void drawPolygon(double[] xPoints, double[] yPoints, int numPoints)

  // draw solid shapes (not outlines)
 public void fillRect(double left, double top, double width, double height)
 public void fillOval(double left, double top, double width, double height)  // Draws solid oval
 public void fillArc(double left, double top, double width, double height, double startAngle, double arcAngle)
 public void fillPolygon(double[] xPoints, double[] yPoints, int numPoints)

 public void drawString(String s, double left, double baseline)
 public void drawImage(String filename, double left, double top)
 public void drawImage(String filename, double left, double top, double width, double height)
  // erasing and inverting: all the drawXXX commands except drawImage also have an eraseXXX and invertXXX form
```

```
  // Misc
  public void sleep(double millis) // pause program for specified time (milliseconds).
  public void initialise ()        // ensure UI window has been initialised
  public void quit()               // delete UI window; usually halts the program.
  // Event-based input
  public void addButton(String name, mth) // Add a button to input panel, mth is a method with no arguments
  public void addTextField(String s, mth) // Add a textField to input panel, mth is a method with String argument
  public void addSlider(String s, double min, double max, mth)
                                   // Add a slider to input panel, mth is a method with double argument
  public void addSlider(String s, double min, double max, double initial, UISliderListener obj)
  public void setMouseListener(mth) // Set mouseListener, mth is method with String and two doubles
  public void setMouseMotionListener(mth) // Set mouseMotionListener,
  public void setKeyListener(mth) // Set keyListener for Graphics pane, mth is a method with String argument

// Typical usage:
   UI.addButton("Load", this::doLoadData);
   UI.addButton("Quit", UI::quit);
   UI.setMouseListener(this::doMouse);
```

```
Color class:
   public Color(int red, int green, int blue)      // Make a colour; arguments must be 0..255
   public Color(float red, float green, float blue) // Make a colour; arguments must be 0.0 to 1.0
   Color.gray, Color.blue, Color.red,              // Some of the predefined colours
   Color.green, Color.black, Color.white
// Typical usage:
   UI.setColor(Color.blue);
   Color col = new Color((float)Math.random(),(float)Math.random(),(float)Math.random());
```

```
Integer class:
   public static final int MAX_VALUE   // The largest possible int: 2^(31)-1
   public static final int MIN_VALUE   // The smallest possible int: -2^31
// Typical usage:
    int max = Integer.MIN_VALUE;       // find maximum of a file of integers
    while (scan.hasNextInt()){
        int num = scan.nextInt();
        if (num > max){ max = num; }
    }
```

```
 Double class:
    public static final double MAX_VALUE          // The largest possible double: just under 2^(1024)
    public static final double MIN_VALUE          // The smallest possible positive nonzero double
    public static final double POSITIVE_INFINITY // positive infinity (greater than any number)
    public static final double NEGATIVE_INFINITY // negative infinity (less than any number)
    public static final double NaN                // The double that is 'Not a Number'
// Typical usage:
    double min = Double.POSITIVE_INFINITY;        // find minimum of an array of doubles
    for (int i=0; i<numbers.length; i++){
        if (numbers[i] < min){
            min = numbers[i];
        }
    }
```

Math **class**:

```
   public static double sqrt(double x)                 // Returns the square root of x
   public static double min(double x, double y)        // Returns the smaller of x and y
   public static double max(double x, double y)        // Returns the larger of x and y
   public static double abs(double x)                  // Returns the absolute value of x
   public static int min(int x, int y)                 // Returns the smaller of x and y
   public static int max(int x, int y)                 // Returns the larger of x and y
   public static int abs(int x)                        // Returns the absolute value of x
   public static double random()                       // Returns a random number between 0 and 1.0
   public static double hypot(double dx, double dy)    // Returns sqrt(dx*dx + dy*dy)
// Typical usage:
     if ( Math.random()<0.1) { ...                     // do something with probablity 0.1
     int size = (int)(Math.random()*50);               // a random integer between 0 and 49.
     double diagonal = Math.hypot(wd, ht);   same as     = Math.sqrt((wd*wd) + (ht*ht));
```

String **class**:

```
   public int length()                                 // Returns the length (number of characters) of the string
   public boolean equals(String s)                     // Returns true if string has same characters as s
   public boolean equalsIgnoreCase(String s)           // String has same characters as s, ignoring their case
   public String toUpperCase()                         // Returns upper case copy of string
   public String toLowerCase()                         // Returns lower case copy of string
   public boolean startsWith(String patrn)             // Returns true if first part of string matches patrn
   public boolean endsWith(String patrn)               // Returns true if last part of string matches patrn
   public boolean contains(String patrn)               // Returns true if patrn matches some part of the string
   public int compareTo(String other)                  // Returns -ve if this string is alphabetically before other,
                                                       //         +ve if after other, 0 if equal to other
   public String substring(int j, int k)               // Returns substring from index j to index k-1
                                                       //   requires 0 <= j <= k <= length of string
   public String trim ():                              // Returns copy of string with spaces at ends removed
   public int indexOf(String patrn)                    // Returns the index of where patrn first matches
                                                       // Returns -1 if string does not contain patrn anywhere
// Typical usage:
// assume name, answer, and courseCode are all variables of type String
   if ( answer.equals(name) ) { ...       OR     if ( answer.equalsIgnoreCase(name) ) { ...
   System.println("Ans:"+ answer.toLowerCase());
   if ( name.startsWith("Pe") ) { ...
   System.println("Course number =" + courseCode.substring(4, 7));
```

File **class**:

```
public File (String fname)      // Constructor. Creates a File object attached to the file named fname
public boolean exists()         // Returns true if and only if the file already exists

// Typical usage:
File inFile = new File(UIFileChooser.open("File name"));r
if ( inFile.exists() ){
   try {
      Scanner sc = new Scanner(inFile);
      while (sc.hasNext()){
         UI.println(sc.nextLine());
      }
      sc.close();
   } catch(IOException e){UI.println("file reading failed: "+e);}
}
```

Scanner **class**:

```
public Scanner(File f)          // Constructor, for reading from a file
public Scanner(InputStream i)   // Constructor. Note: System.in is an InputStream
public Scanner(String s)        // Constructor, for reading from a string
public boolean hasNext()        // Returns true if there is more to read
public boolean hasNextInt()     // Returns true if the next token is an integer
public boolean hasNextDouble()  // Returns true if the next token is a number
public String next()            // Returns the next token (chars up to a space/line)
public String nextLine()        // Returns string of chars up to next newline
                                //  (next and nextLine throw exception if no more tokens)
public int nextInt()            // Returns the integer value of the next token
                                //  (throws exception if next token is not an integer or no more tokens)
public double nextDouble()      // Returns the double value of the next token
                                //  (throws exception if next token is not a number or no more tokens)
public void close()             // Closes the file (if it is wrapping a File object)

// Typical usage:
Scanner scan = new Scanner(new File("data.txt")); // find total of numbers in a file of integers
double total = 0;
while (scan.hasNextDouble()) {
   total = total + scan.nextDouble();
}
scan.close();
```

UIFileChooser **class**:

```
public static String open()             // Ask user for an existing file
public static String open(String prompt)
public static String save()             // Ask user for a new or existing file
public static String save(String prompt)
// Typical usage:
String imgFileName = UIFileChooser.open("Choose image file");
PrintStream out = new PrintStream(new File(UIFileChooser.save()));
for(Person person : relatives ){ out.println(person.toString()); }
```

Comparison Operators:

```
==    // is the same value. For objects, compares their ID's. Usually use .equals(...) on objects
!=    // is not the same value. use to check if a value is not null
<  >  <=  >=
```

Logical Operators:

```
&&              // and
||              // or
!               // not (prefix operator)
// Typical usage:
    if ( (x>y && !sc.hasNext() ) || ( list.isEmpty()) {  ....
```

Increment *Operators*:

```
++  --          // add (subtract) 1 from variable, returning previous value of variable
+=  -=  *=  /=  // add (etc) value to variable on left
```

Miscellaneous:

```
    break;                  // exit the immediately enclosing loop (for or while)
    return;                 // exit the method (no value returned)
    return <expression>;    // exit the method, returning the value of the expression
------
    if (condition) {
       action
    }
    else if (condition2) {
       action
    }
    else {
       action
    }
--------
    while (condition) {
       body
    }
-------
    for (int i=1; i<=20; i++){
       body
    }
```