COMP102 Test 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)
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, function) // Add a button, function is a method with no arguments
// Typical usage:
  UI.addButton("Load", this::loadData);
  UI.addButton("Quit", UI::quit);
```

```
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 num1 = UI.askInt("A number");     // The smallest of two numbers
   int num2 = UI.askInt("Another number");
   int smallest = Math.min(num1,num2);
```

```
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 d1 = Math.random() * 100;        // find maximum of two doubles
   double d2 = Math.random() * 100;
   double max = Math.max(d1,d2);
```

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)                      // Returns substring starting from index j to end
                                                    //  first character is at index 0
public String substring(int j, int k)               // Returns substring from index j to index k-1
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 correctAns are all variables of type String
   if ( answer.equals(correctAns) ) {
      UI.println("Answer correct");
   } else {
      UI.println("Answer incorrect");
   }
   if ( name.startsWith("Peter") || name.startsWith("Richard")) {
      UI.println("Can I call you " + name.substring(4) + "?");
   }
```

Comparison Operators:

```
==    // is the same value. Use .equals(...) on objects (such as Strings) usually
!=    // is not the same value.
<  >  <=  >=
```

Logical Operators:

```
&&                  // and
||                  // or
!                   // not (prefix operator)
// Typical usage:
    if ( (x > 10 && x < 20 ) || (y != 100) {  ....  }
    if ( (!name.equals("Karsten") && answer.equals(correctAns) { .... }
```