Viability of a Molar Mass and Uncertainty Calculator in Java

Kadin Green and Eddie Ironsmith (Faculty Advisor)
College of Science and Mathematics
Stephen F. Austin State University, Nacogdoches, Texas

Abstract or Introduction

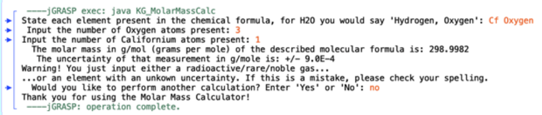
The calculation of the molar mass of any given compound is a very simple process, though it can be somewhat repetitive and for complex structures it may take some amount of time. Furthermore, the uncertainty values are easy to calculate, but they are difficult to easily get ahold of reliably to certain extents of significant figures for some elements. It may also be somewhat difficult for students new to chemistry to calculate molar mass on the fly. As such a simple program was written to automatically calculate the molar mass and uncertainty in the measurement thereof for the user. Such a program could also be used as a simple teaching tool to encourage students to test their calculations against those of the code.

Overview of Process

The process by which the program functions is very streamlined. The program starts by asking the user to input the various types of elements present in the compound. It then asks the user for the number of atoms present for each element. Then after a quick calculation it outputs the results, alongside an uncertainty value pulled from the J. Phys.Chem. Ref.Data 2007, 36, 485. After outputting the molar mass and the uncertainty, the program asks the user if they would like to do another calculation, looping if they answer yes and ending the program if they answer no.

Additional Notes

The program was written with the intent to be capable of being used with any element on the periodic table, as well as deuterium. However, some elements such as those of noble gasses, those that are manmade or radioactive, and those that have an unknown uncertainty may not be suitable for calculations. Due to them not being likely to appear in a chemical formula. If they do appear the program spots this and gives a friendly reminder to check spelling in case it is a mistake. The program is also written to accept any capitalized or uncapitalized names of elements, as well as capitalized and uncapitalized elemental symbols such as Br. Shown below (Figure 1) is an example of code being run for chromium(III) oxide. In this scenario however, the user misspells Cr for chromium as Cf, inadvertently using californium. In the example the program recognizes this and points it out in case it is a mistake. The user could then spot this and run the calculation again, this time being more careful with spelling.



```
 ----jGRASP exec: java KG_MolarMassCalc
State each element present in the chemical formula, for H2O you would say 'Hydrogen, Oxygen': Cf Oxygen
 Input the number of Oxygen atoms present: 3
Input the number of Californium atoms present: 1
  The molar mass in g/mol (grams per mole) of the described molecular formula is: 298.9982
   The uncertainty of that measurement in g/mole is: +/- 9.0E-4
Warning! You just input either a radioactive/rare/noble gas...
...or an element with an unkown uncertainty. If this is a mistake, please check your spelling.
  Would you like to perform another calculation? Enter 'Yes' or 'No': no
Thank you for using the Molar Mass Calculator!
 ----jGRASP: operation complete.
```

Figure 1. Code Execution Example

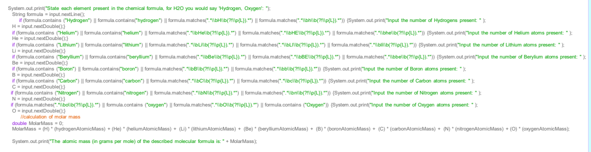
Methodology

The code itself uses very simple concepts that a beginner starting their introductory course to computer science principles could implement after a month or two. Although there are a few examples of more "complex" code, meaning code that would require maybe three or four months of the same introductory class. The code starts out by creating set values for the molar mass of each element, the uncertainty of each element, and a counter for the amount of each element. Then after prompting the user to input the types of elements present, the code detects which elements are present and uses that information to ask the user how many atoms of each element there are in the molecular formula. The program then sets the counters equal to the number inputted for each element. A simple calculation is then done to find the molar mass of the compound by multiplying the number of elements times their respective molar masses before adding them all together. The program then prints out the results and asks if the user would like to do another. The coding started out simple, attempting to understand the basics of how this could be accomplished by creating a code that would calculate molar mass for a compound using elements from hydrogen to oxygen, this code can be seen in Figure 2. After the initial test and some thinking, the code was refined, expanded, improved, and changed to include uncertainty values as well as molar mass, this code can be seen through the QR code in Figure 3. Finally, after much more expansion of the "final" product was made in the form of the current code, though further optimizations are very possible. This code can be seen through the QR code in Figure 4.

Figure 3: First Code QR code

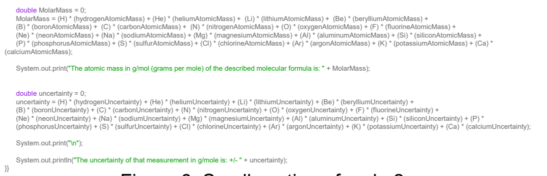
Figure 3: Second Code QR code

Figure 4: Third Code QR Code



```
System.out.print("State each element present in the chemical formula, for H2O you would say 'Hydrogen, Oxygen': ");
  String formula = input.nextLine();
     if (formula.contains ("Hydrogen") || formula.contains("hydrogen") || formula.matches(".*\\bH\\b(?!\\p{L}).*") || formula.matches(".*\\bh\\b(?!\\p{L}).*")) {System.out.print("Input the number of Hydrogens present: " );
  H = input.nextDouble();}
  if (formula.contains ("Helium") || formula.contains("helium") || formula.matches(".*\\bHe\\b(?!\\p{L}).*") || formula.matches(".*\\bHE\\b(?!\\p{L}).*") || formula.matches(".*\\bhe\\b(?!\\p{L}).*")) {System.out.print("Input the number of Helium atoms present: " );
  He = input.nextDouble();}
  if (formula.contains ("Lithium") || formula.contains("lithium") || formula.matches(".*\\bLi\\b(?!\\p{L}).*") || formula.matches(".*\\bLI\\b(?!\\p{L}).*") || formula.matches(".*\\bli\\b(?!\\p{L}).*")) {System.out.print("Input the number of Lithium atoms present: " );
  Li = input.nextDouble();}
  if (formula.contains ("Beryllium") || formula.contains("beryllium") || formula.matches(".*\\bBe\\b(?!\\p{L}).*") || formula.matches(".*\\bBE\\b(?!\\p{L}).*") || formula.matches(".*\\bbe\\b(?!\\p{L}).*")) {System.out.print("Input the number of Berylium atoms present: " );
  Be = input.nextDouble();}
  if (formula.contains ("Boron") || formula.contains("boron") || formula.matches(".*\\bB\\b(?!\\p{L}).*") || formula.matches(".*\\bb\\b(?!\\p{L}).*")) {System.out.print("Input the number of Boron atoms present: " );
  B = input.nextDouble();}
  if (formula.contains ("Carbon") || formula.contains("carbon") || formula.matches(".*\\bC\\b(?!\\p{L}).*") || formula.matches(".*\\bc\\b(?!\\p{L}).*")) {System.out.print("Input the number of Carbon atoms present: " );
  C = input.nextDouble();}
 if (formula.contains ("Nitrogen") || formula.contains("nitrogen") || formula.matches(".*\\bN\\b(?!\\p{L}).*") || formula.matches(".*\\bn\\b(?!\\p{L}).*")) {System.out.print("Input the number of Nitrogen atoms present: " );
  N = input.nextDouble();}
 if (formula.matches(".*\\bo\\b(?!\\p{L}).*") || formula.contains ("oxygen") || formula.matches(".*\\bO\\b(?!\\p{L}).*") || formula.contains ("Oxygen")) {System.out.print("Input the number of Oxygen atoms present: " );
  O = input.nextDouble();}
      //calculation of molar mass
  double MolarMass = 0;
  MolarMass = (H) * (hydrogenAtomicMass) + (He) * (heliumAtomicMass) +  (Li) * (lithiumAtomicMass) +  (Be) * (berylliumAtomicMass) +  (B) * (boronAtomicMass) +  (C) * (carbonAtomicMass) +  (N) * (nitrogenAtomicMass) + (O) * (oxygenAtomicMass);

  System.out.print("The atomic mass (in grams per mole) of the described molecular formula is: " + MolarMass);
```

Figure 5: Small section of code 1

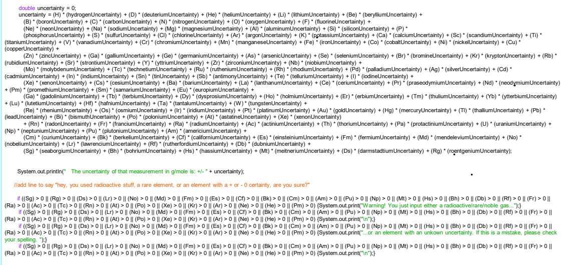


```
    double MolarMass = 0;
    MolarMass = (H) * (hydrogenAtomicMass) + (He) * (heliumAtomicMass) +  (Li) * (lithiumAtomicMass) +  (Be) * (berylliumAtomicMass) +
    (B) * (boronAtomicMass) +  (C) * (carbonAtomicMass) +  (N) * (nitrogenAtomicMass) + (O) * (oxygenAtomicMass) + (F) * (fluorineAtomicMass) +
    (Ne) * (neonAtomicMass) + (Na) * (sodiumAtomicMass) + (Mg) * (magnesiumAtomicMass) + (Al) * (aluminumAtomicMass) + (Si) * (siliconAtomicMass) +
    (P) * (phosphorusAtomicMass) + (S) * (sulfurAtomicMass) + (Cl) * (chlorineAtomicMass) + (Ar) * (argonAtomicMass) + (K) * (potassiumAtomicMass) + (Ca) *
  (calciumAtomicMass);

    System.out.print("The atomic mass in g/mol (grams per mole) of the described molecular formula is: " + MolarMass);


    double uncertainty = 0;
    uncertainty = (H) * (hydrogenUncertainty) + (He) * (heliumUncertainty) + (Li) * (lithiumUncertainty) + (Be) * (berylliumUncertainty) +
    (B) * (boronUncertainty) + (C) * (carbonUncertainty) + (N) * (nitrogenUncertainty) + (O) * (oxygenUncertainty) + (F) * (fluorineUncertainty) +
    (Ne) * (neonUncertainty) + (Na) * (sodiumUncertainty) + (Mg) * (magnesiumUncertainty) + (Al) * (aluminumUncertainty) + (Si) * (siliconUncertainty) + (P) *
    (phosphorusUncertainty) + (S) * (sulfurUncertainty) + (Cl) * (chlorineUncertainty) + (Ar) * (argonUncertainty) + (K) * (potassiumUncertainty) + (Ca) * (calciumUncertainty);

    System.out.print("\n");

    System.out.println("The uncertainty of that measurement in g/mole is: +/- " + uncertainty);
  }}
```

Figure 6: Small section of code 2

Discussion

This program was written in a limited time frame with only a basic understanding of the introductory sections of coding for Java. Given more time and even just a novice-level understanding of coding, much more could be completed. Current things that can be improved on in the program include changing the way the code recognizes elements to allow one to input a chemical formula directly. Such as Sr(OH)2, this could allow a user to skip out having to count and state the number of atoms present for each element. This would make calculating molar mass for smaller compounds more efficient, though for incredibly large compounds in which a chemical formula is not given directly, and rather a structure is given, the current form may be more applicable. Applying this would also allow one to cut out a large portion of what would be thereafter unnecessary code. Other things that could be done include condensing the large list of doubles for uncertainties, molar masses, and element symbols. The program could be used as a teaching tool as is by giving a student pre-determined formulas, telling them to calculate the molar mass for them, then checking if their answer is within perhaps 1% of the actual value. However, it could be further streamlined if made specifically for this function, by having the program randomly generate a number, use that number to choose from a pre-made list of formulas, ask the user to calculate the molar mass and input their answer, and then provide the correct answer and any percent difference. The source currently being used to draw molar masses of elements and uncertainties of elements is the same source a current textbook for a quantitative analysis course at SFA uses, though a more recent source than 2007 may be more accurate.



```
      double uncertainty = 0;
      uncertainty = (H) * (hydrogenUncertainty) + (D) * (deuteriumUncertainty) + (He) * (heliumUncertainty) + (Li) * (lithiumUncertainty) + (Be) * (berylliumUncertainty) +
        (B) * (boronUncertainty) + (C) * (carbonUncertainty) + (N) * (nitrogenUncertainty) + (O) * (oxygenUncertainty) + (F) * (fluorineUncertainty) +
        (Ne) * (neonUncertainty) + (Na) * (sodiumUncertainty) + (Mg) * (magnesiumUncertainty) + (Al) * (aluminumUncertainty) + (Si) * (siliconUncertainty) + (P) *
        (phosphorusUncertainty) + (S) * (sulfurUncertainty) + (Cl) * (chlorineUncertainty) + (Ar) * (argonUncertainty) + (K) * (potassiumUncertainty) + (Ca) * (calciumUncertainty) + (Sc) * (scandiumUncertainty) + (Ti) *
(titaniumUncertainty) + (V) * (vanadiumUncertainty) + (Cr) * (chromiumUncertainty) + (Mn) * (manganeseUncertainty) + (Fe) * (ironUncertainty) + (Co) * (cobaltUncertainty) + (Ni) * (nickelUncertainty) + (Cu) *
(copperUncertainty) +
        (Zn) * (zincUncertainty) + (Ga) * (galliumUncertainty) + (Ge) * (germaniumUncertainty) + (As) * (arsenicUncertainty) + (Se) * (seleniumUncertainty) + (Br) * (bromineUncertainty) + (Kr) * (kryptonUncertainty) + (Rb) *
(rubidiumUncertainty) + (Sr) * (strontiumUncertainty) + (Y) * (yttriumUncertainty) + (Zr) * (zirconiumUncertainty) + (Nb) * (niobiumUncertainty) +
        (Mo) * (molybdenumUncertainty) + (Tc) * (technetiumUncertainty) + (Ru) * (rutheniumUncertainty) + (Rh) * (rhodiumUncertainty) + (Pd) * (palladiumUncertainty) + (Ag) * (silverUncertainty) + (Cd) *
(cadmiumUncertainty) + (In) * (indiumUncertainty) + (Sn) * (tinUncertainty) + (Sb) * (antimonyUncertainty) + (Te) * (telluriumUncertainty) + (I) * (iodineUncertainty) +
        (Xe) * (xenonUncertainty) + (Cs) * (cesiumUncertainty) + (Ba) * (bariumUncertainty) + (La) * (lanthanumUncertainty) + (Ce) * (ceriumUncertainty) + (Pr) * (praseodymiumUncertainty) + (Nd) * (neodymiumUncertainty)
+ (Pm) * (promethiumUncertainty) + (Sm) * (samariumUncertainty) + (Eu) * (europiumUncertainty) +
        (Ga) * (gadoliniumUncertainty) + (Tb) * (terbiumUncertainty) + (Dy) * (dysprosiumUncertainty) + (Ho) * (holmiumUncertainty) + (Er) * (erbiumUncertainty) + (Tm) * (thuliumUncertainty) + (Yb) * (ytterbiumUncertainty)
+ (Lu) * (lutetiumUncertainty) + (Hf) * (hafniumUncertainty) + (Ta) * (tantalumUncertainty) + (W) * (tungstenUncertainty) +
        (Re) * (rheniumUncertainty) + (Os) * (osmiumUncertainty) + (Ir) * (iridiumUncertainty) + (Pt) * (platinumUncertainty) + (Au) * (goldUncertainty) + (Hg) * (mercuryUncertainty) + (Tl) * (thalliumUncertainty) + (Pb) *
(leadUncertainty) + (Bi) * (bismuthUncertainty) + (Po) * (poloniumUncertainty) + (At) * (astatineUncertainty) + (Xe) * (xenonUncertainty)
        + (Rn) * (radonUncertainty) + (Fr) * (franciumUncertainty) + (Ra) * (radiumUncertainty) + (Ac) * (actiniumUncertainty) + (Th) * (thoriumUncertainty) + (Pa) * (protactiniumUncertainty) + (U) * (uraniumUncertainty) +
(Np) * (neptuniumUncertainty) + (Pu) * (plutoniumUncertainty) + (Am) * (americiumUncertainty) +
        (Cm) * (curiumUncertainty) + (Bk) * (berkeliumUncertainty) + (Cf) * (californiumUncertainty) + (Es) * (einsteiniumUncertainty) + (Fm) * (fermiumUncertainty) + (Md) * (mendeleviumUncertainty) + (No) *
(nobeliumUncertainty) + (Lr) * (lawrenciumUncertainty) + (Rf) * (rutherfordiumUncertainty) + (Db) * (dubniumUncertainty) +
        (Sg) * (seaborgiumUncertainty) + (Bh) * (bohriumUncertainty) + (Hs) * (hassiumUncertainty) + (Mt) * (meitneriumUncertainty) + (Ds) * (darmstadtiumUncertainty) + (Rg) * (roentgeniumUncertainty);


      System.out.println("   The uncertainty of that measurement in g/mole is: +/- " + uncertainty);

     //add line to say "hey, you used radioactive stuff, a rare element, or an element with a + or - 0 certainty, are you sure?"

      if ((Sg) > 0 || (Rg) > 0 || (Ds) > 0 || (Lr) > 0 || (No) > 0 || (Md) > 0 || (Fm) > 0 || (Es) > 0 || (Cf) > 0 || (Bk) > 0 || (Cm) > 0 || (Am) > 0 || (Pu) > 0 || (Np) > 0 || (Mt) > 0 || (Hs) > 0 || (Bh) > 0 || (Db) > 0 || (Rf) > 0 || (Fr) > 0 ||
(Ra) > 0 || (Ac) > 0 || (Tc) > 0 || (Rn) > 0 || (At) > 0 || (Po) > 0 || (Xe) > 0 || (Kr) > 0 || (Ar) > 0 || (Ne) > 0 || (He) > 0 || (Pm) > 0) {System.out.print("Warning!  You just input either a radioactive/rare/noble gas...");}
      if ((Sg) > 0 || (Rg) > 0 || (Ds) > 0 || (Lr) > 0 || (No) > 0 || (Md) > 0 || (Fm) > 0 || (Es) > 0 || (Cf) > 0 || (Bk) > 0 || (Cm) > 0 || (Am) > 0 || (Pu) > 0 || (Np) > 0 || (Mt) > 0 || (Hs) > 0 || (Bh) > 0 || (Db) > 0 || (Rf) > 0 || (Fr) > 0 ||
(Ra) > 0 || (Ac) > 0 || (Tc) > 0 || (Rn) > 0 || (At) > 0 || (Po) > 0 || (Xe) > 0 || (Kr) > 0 || (Ar) > 0 || (Ne) > 0 || (He) > 0 || (Pm) > 0) {System.out.print("\n");}
      if ((Sg) > 0 || (Rg) > 0 || (Ds) > 0 || (Lr) > 0 || (No) > 0 || (Md) > 0 || (Fm) > 0 || (Es) > 0 || (Cf) > 0 || (Bk) > 0 || (Cm) > 0 || (Am) > 0 || (Pu) > 0 || (Np) > 0 || (Mt) > 0 || (Hs) > 0 || (Bh) > 0 || (Db) > 0 || (Rf) > 0 || (Fr) > 0 ||
(Ra) > 0 || (Ac) > 0 || (Tc) > 0 || (Rn) > 0 || (At) > 0 || (Po) > 0 || (Xe) > 0 || (Kr) > 0 || (Ar) > 0 || (Ne) > 0 || (He) > 0 || (Pm) > 0) {System.out.print("...or an element with an unkown uncertainty.  If this is a mistake, please check
your spelling.  ");}
      if ((Sg) > 0 || (Rg) > 0 || (Ds) > 0 || (Lr) > 0 || (No) > 0 || (Md) > 0 || (Fm) > 0 || (Es) > 0 || (Cf) > 0 || (Bk) > 0 || (Cm) > 0 || (Am) > 0 || (Pu) > 0 || (Np) > 0 || (Mt) > 0 || (Hs) > 0 || (Bh) > 0 || (Db) > 0 || (Rf) > 0 || (Fr) > 0 ||
(Ra) > 0 || (Ac) > 0 || (Tc) > 0 || (Rn) > 0 || (At) > 0 || (Po) > 0 || (Xe) > 0 || (Kr) > 0 || (Ar) > 0 || (Ne) > 0 || (He) > 0 || (Pm) > 0) {System.out.print("\n");}
```

Figure 7: Small section of code 3

Conclusion

In conclusion, where this specific code may have ample room for improvement, it serves as a good basis and proof of concept for calculators to find molar masses and uncertainties quicker, as well as to potentially be used as a learning tool.