C++ - Assignment 5

Practicing inheritance, smart pointers and friendship

Charanjit Kaur & Caterina Doglioni, PHYS30762 - OOP in C++ 2025

The University of Manchester

How we are going to mark this assignment

- This is the outcome of a constructive discussion with students in the lab on Thursday
- It may appear that in this course *we tell you exactly how to do something in the brief, you do that thing, and you get the marks*
  - This is **not** how we assess this course
    - You can easily ask ChatGPT to do that, and we are not interested in teaching to large language models/employers don't have to give a salary to these
  - This is also not how any other course is marked (think of essays/presentations/problems that can be solved in multiple ways)
- We are giving you guidelines to design and write good code
- The lectures tell you everything you need to do in terms of implementation, but we also evaluate the **quality of your code** (similar to the quality of your e.g. lab reports) in terms of **how easy it is for the user of your classes to use your code**
  - In this assignment and in the project, you have to think about this aspect more (including more emphasis on good comments) - and we won't tell you exactly what to do.
    - The best way to do this is put yourself in your user's shoes and ask questions such as: *do they understand your logic? Could they use your code erroneously?*

The assignment: introduction

- We will simulate simplified decays of **radioactive nuclei**
  - Cs-137, Na-22, Co-60 (and include stable Fe-56, for comparison)
- As a consequence of their radioactivity, these nuclei produce **photons** of a given energy
- Photons interact with matter through:
  - photoelectric effect
  - Compton scattering
  - pair production
- In this assignment, you will simulate the radioactive nuclei, the photon emission, and their interactions with matter

Part 1a. Classes for nuclei

- Implement a “*nucleus*” class

You will have to name the classes, data members and properties according to the house style

- Implement the *”stable nucleus”* and *“radioactive nucleus”* classes are derived from the nucleus class —> you must build an inheritance chain

- These classes have the following properties and methods (note: not all classes have all the properties! think about what is common, and what is unique)
  - **Atomic mass, atomic number, half-life**, **nucleus type** (He, Co, Cs, Na are possible), **whether it has decayed**
  - a **smart pointer** to a *Photon* object
  - **setters, getters, decay()**, **print_data()**

These functions must have parameters and implementation that you decide, we are not going to tell you exactly what parameters you need as we want to evaluate your thinking/design too

- You must use the **rule of 5** in presence of dynamic memory management

Part 1b. Particle classes

- Implement a “*particle*” class

You will have to name the classes, data members and properties according to the house style

- Implement the *”electron”* and *“photon”* classes, derived from the particle class —> you must build an inheritance chain

- These classes have the following properties and methods (note: not all classes have all the properties! think about what is common, what is unique,)

  - **rest mass, energy**
  - **electron: a vector of smart pointers to photons (for radiation)**
  - **photon: a vector of smart pointers to electrons (for pair-production)**
  - **setters, getters, print_data()**

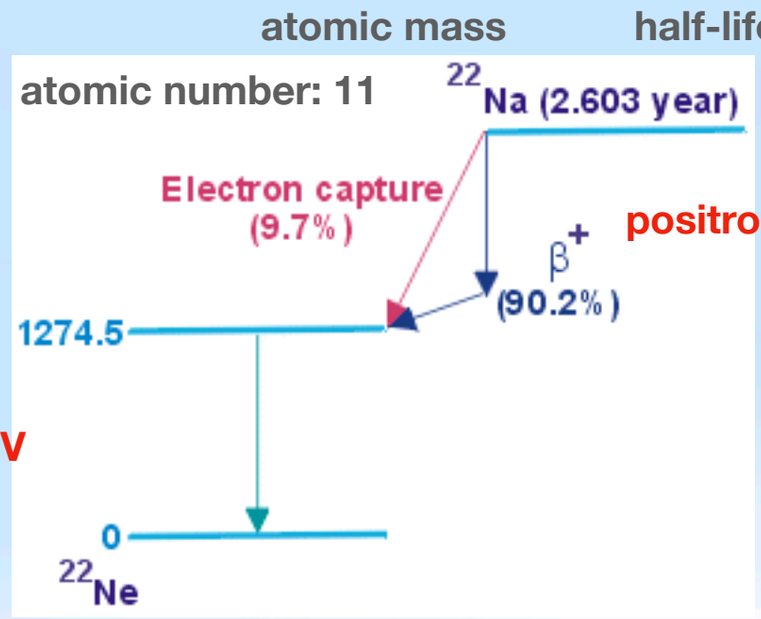
These functions must have parameters and implementation that you decide, we are not going to tell you exactly what parameters you need as we want to evaluate your thinking/design too

Part 2. Decaying nuclei

- The **decay()** function of a nucleus must
  - return one or more photons with the correct energy (see diagrams in the next slide)
    - If the nucleus can produce photons with more than one energy, you should return photons with all the possible values
    - It should call the print_data() function of the photon
  - set the “whether the nucleus has decayed” property to True
- ***Note***: you should not be able to call the decay() function for a stable nucleus

Part 2. Decaying nuclei

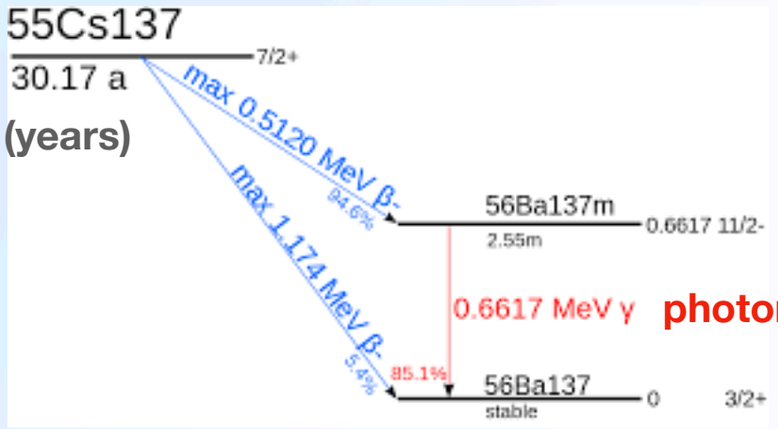
https://ns.ph.liv.ac.uk/~ajb/radiometrics



positron (challenge)

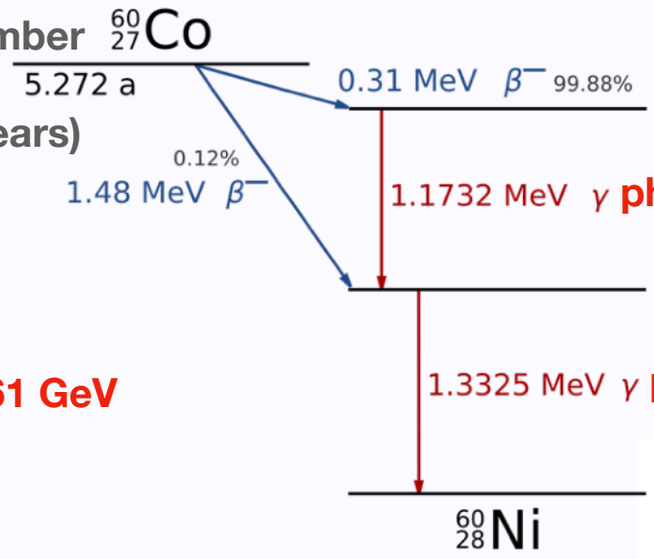
photon energy: 1274.5 MeV

atomic number
atomic mass
half-life (years)



photon energy: 661 GeV

atomic mass
atomic number
half-life (years)



photon energy: 1173 MeV

photon energy: 1333 MeV

Part 3a. Photons interacting with matter

- You should have three **friend** functions acting on photons
  - “photoelectric effect” function: just returns the photon energy
  - “Compton effect” function: changes the energy of the photon based on the following (Eɣ’ is the new photon energy, use MeV as units, c=1, m_e = 511 keV, you can vary or fix theta in the function)

$$E_{\gamma'} = \frac{E_\gamma}{1 + (E_\gamma / m_e c^2)(1 - \cos\theta)}$$

- ”Pair production” function: here we won’t return/calculate the energy, we will just return a vector of two “electron” particles
  - this is only allowed if the energy of the photon is > twice the mass of the electron, if not output a clear error message and return an empty vector

We won’t tell you exactly what to do here, you have to think what is best and we will mark according to how well you are practicing concepts from the pre-lectures

Part 3b. Electrons radiating photons

- You should have one **friend** function acting on electrons
  - "radiate" function: returns one of the photons in the vector of photons within the electron
    - *think about what makes sense if you were doing a simulation: will the electron keep radiating forever? how many photons do you want in the electron? What energy should they have?*
    - *Explain your thinking and implementation in a comment above the function*

Again, we won't tell you exactly what to do here, you have to think what is best in terms of design and experience for the user of your classes

Part 4. Show that it all works in main()

- Instantiate four nuclei: Na, Cs, Co, Fe - this can be in a vector to use polymorphism, but it can also be done in other ways, you choose
- The Cs decay leads to a photon with E = 661 keV
- The Na decay leads to a photon with E = 1274.5 keV
  - [challenge] implement also the positron emission, what happens in this case? (For this, you have to look back to your nuclear physics course notes - don't do anything else for the other nuclei)
- The Co decay leads to emit two photons with E = 1173 keV and E = 1333 keV
- Let each of these photons interact with matter, with one of the three interactions in the friend functions
  - Show that one of these photons cannot pair-produce (to check your error message)
  - Show that one of the electrons that have been pair-produced radiates (at least) another photon (unrelated to the original photon from the nucleus)

You won't be marked down for using a different rounding

Link to join the GitHub repository:

https://classroom.github.com/a/6kGhv0vx