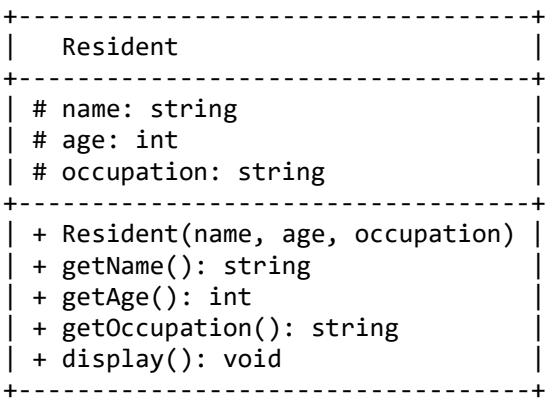
Chapter 17. Lab: Managing Springfield Residents using Vectors

Objective
In this lab experience you will practice **C++ classes, vectors, and iterators** by creating a system to manage **Springfield residents**. This will include:

- Creating a **Resident** class
- Using a std::vector to store objects
- Traversal and modification of the vector.
- Sorting and searching within the vector

---

1. UML Diagram
Here is a UML representation of the Resident class:



---

2. Create the Resident Class
Define a class with attributes for **name, age, and occupation**, along with relevant methods. Use the Lazy-class approach.

3. Complete the tasks listed in the following skeleton main function

```
int main()
{
    vector<Resident> residents;
    residents.push_back(Resident("Homer Simpson", 39, "Nuclear Safety Inspector"));
    residents.push_back(Resident("Marge Simpson", 36, "Housewife"));
    residents.push_back(Resident("Bart Simpson", 10, "Student"));
    residents.push_back(Resident("Lisa Simpson", 8, "Student"));
    residents.push_back(Resident("Maggie Simpson", 1, "Baby"));

    cout << "All residents:" << endl;
    for (Resident r : residents) {
        r.display();
    }

    //TODO: Add two more residents to the vector
    // (e.g. "Ned Flanders", 60, "Store owner")
    // (e.g. "Mr. Skinner", 64, "School Principal")

    cout << "\n\nResidents over 30 years old:" << endl;
    //TODO: Display residents over 30 years old

    cout << "\n\nResidents who are students:" << endl;
    //TODO: Display residents who are students

    cout << "\nSorted by age:" << endl;
    //TODO: Sort the residents by age and display them
    //Use Bubblesort or Selectionsort.

    cout << "\nSearch for a resident by name:" << endl;
    //TODO: Search for a resident by name (ask operator to enter a name)

    cout << "\nAll done!" << endl;
}
```