Semester 2 2024
Astroinformatics II

Tutorial 6: Object Oriented Programming with C++

In this tutorial session, we will extend our C++ code to include object-oriented concepts.

1 Classes

C++ extends C by the usage of classes.

In C++, object-oriented programming allows us to bundle together data members (such as variables, arrays, etc.) and its related functions into a single entity. This programming feature is known as encapsulation.

We start here with a basic class that describes an astronomical object:

```
class AstroObj {
    public:
        double coordRa;
        double coordDec;
        double mag_g;
        double mag_r;
        double mag_i;
}
```

We can instantiate the class by declaring an object of type `AstroObj`:

```
obj1 AstroObj;
```

We then set the public attributes:

```
obj1.ra = 35.7;
obj1.dec = 76.9;
mag_g = 14.2;
mag_r = 14.0;
mag_i = 13.2;
```

Task:

Put this together into a C++ program, compile and run it.

2 Constructors and Attributes

Above, we are setting the values after creating an object.
Also, there is no check whether values entered make sense.
We can improve this.

Tasks:

a) Extend the program by a constructor that sets the five public attributes.

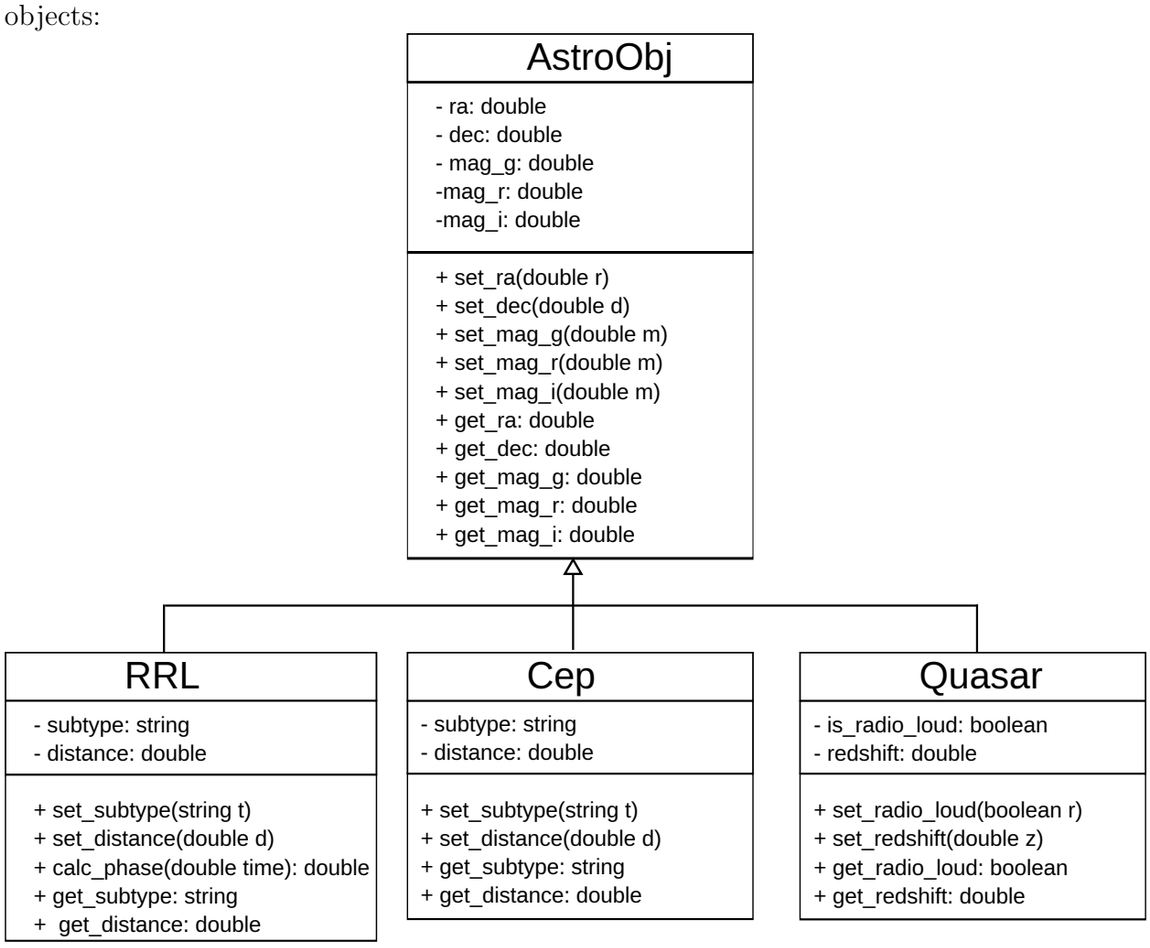
b) Extend the program then by a method that sets the attributes `ra`, `dec`, `mag_g`, `mag_r`, `mag_i`. Check online which values are allowed for `ra` and `dec`. For the magnitudes, we assume that they are (e.g., as given by a survey) between 10.0 and 23.0.

c) You now made those two changes, constructor and the methods that set the values (also called *setter methods*). Extend the program so that it works completely, e.g. one should create an object given the five attributes, and get and set them only through methods.

3 Inheritance

Inheritance in C++ allows us to create a new class (derived class) from an existing class (base class).
The derived class inherits features from the base class and can have additional features of its own.

The class *AstroObj* does not have very specific attributes.
It serves as a base class for inherited classes. The following UML class diagram shows a simple inheritance schema that can be used to represent different types of astronomical objects:



Tasks:

a) Create the inherited classes as described in the UML class diagram. The method `calc_phase` can be a "stub", i.e. does not need to have content.

b) Write a program that creates an object from each class, and outputs its attributes.