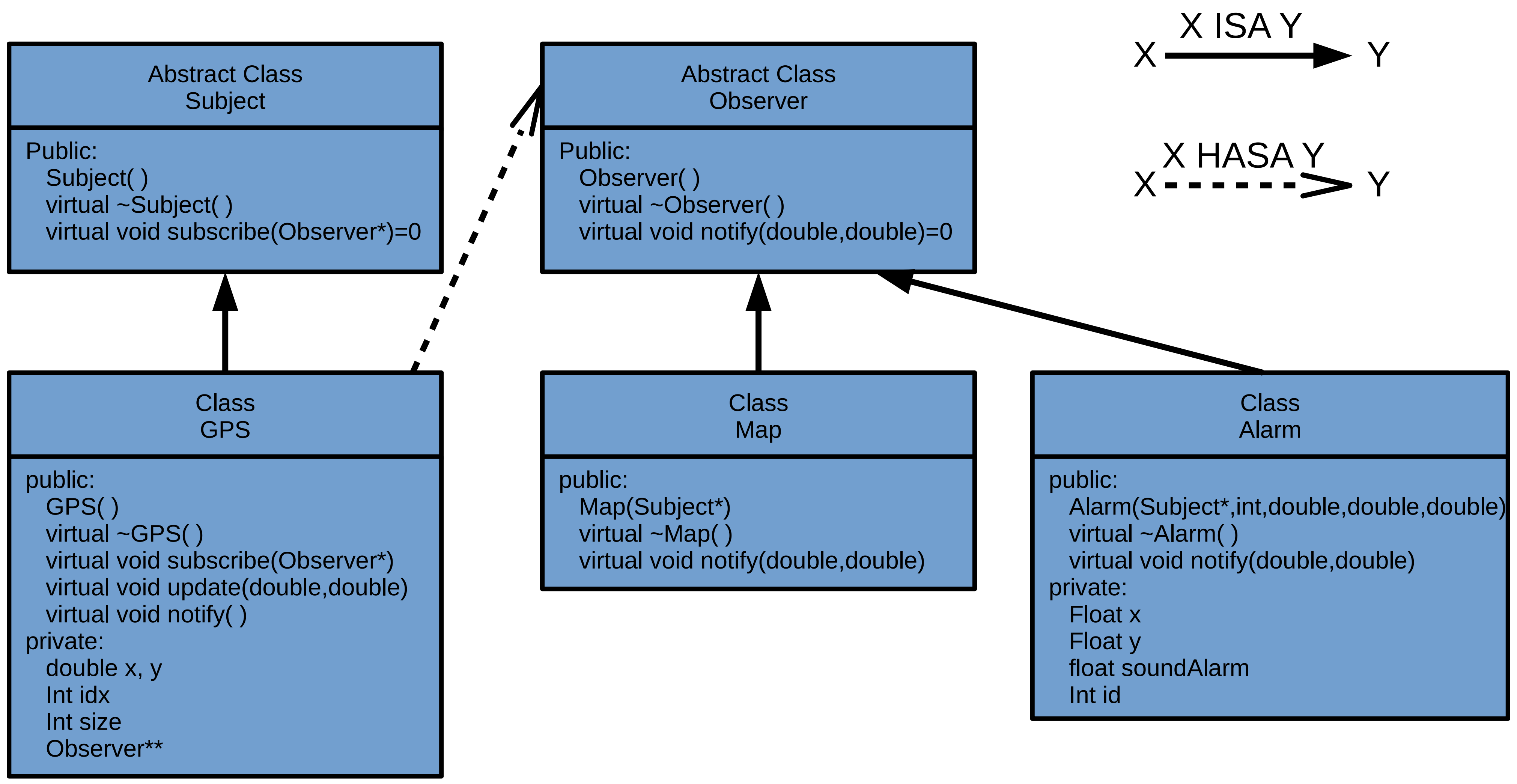
HW6 Instructions

What to turn in: Turn in a compressed, zipped directory named your username. It should contain the files for your HW6a program, and a text file called HW6b with your answers. It should not contain your executable binary file.

HW6a instructions

You will complete a program. The class diagram is given on the next page. What needs to be done for each class is then explained. There are partial implementations of some classes, and the main function is provided.

The program simulates two phone apps – a mapping app and an alarm app. Both apps register with the gps service (the GPS class) and are notified of new GPS locations. The mapping app prints out the new location, and the alarm app sounds an alarm if the device has moved to far.


X ISA Y
X
Y
X HASA Y
X
Y
Abstract Class
Subject
Public:
Subject( )
virtual ~Subject( )
virtual void subscribe(Observer*)=0
Abstract Class
Observer
Public:
Observer( )
virtual ~Observer( )
virtual void notify(double,double)=0
Class
GPS
public:
GPS( )
virtual ~GPS( )
virtual void subscribe(Observer*)
virtual void update(double,double)
virtual void notify( )
private:
double x, y
Int idx
Int size
Observer**
Class
Map
public:
Map(Subject*)
virtual ~Map( )
virtual void notify(double,double)
Class
Alarm
public:
Alarm(Subject*,int,double,double,double)
virtual ~Alarm( )
virtual void notify(double,double)
private:
Float x
Float y
float soundAlarm
Int id

What to do for each class

Subject.h:
read the comment in Subject.h

Observer.h:
Read the comment in Observer.h

GPS.h:
Declare the class according to the descriptor on the previous page.
x and y are the current x and y coordinates.
obs is an array of Observer pointers
size is the size of the obs array
idx is the next element to be written in the obs array

GPS.cpp:
GPS::GPS(int s):
Initialize x and y to 0.0. Initialize size to s. initialize idx to 0. Do all initializations in the initializer list. The obs array allocation is provided.

GPS::~GPS: Free the storage for obs. This has been provided

GPS::subscribe(Observer o)*
Add o to the next entry in the obs array if there is space, and print an error if there is not space. Update idx.

void GPS::update(float xx, float yy);
Set x and y to xx and yy, call notify( )

void GPS::notify( )
For each Observer in obs, call the Observer’s notify function, passing it the value of the GPS object’s x and y.

Map.h:
Declare the class according to the descriptor on the previous page.

Map.cpp:
Map::Map(Subject s):*
Subscribe to the subject s by passing a pointer to this Map object.

Map::~Map: given

void Map::notify(float x, float y)
Print out the new location on the map

What to do for each class

Alarm.h:
Declare the class according to the descriptor on the previous page
The alarm class issues an alarm if the coordinates change too much.
X and y are the initial coordinates. SoundAlarm is how much movement is too much. Id is an integer id of this alarm.

Alarm.cpp
*Alarm::Alarm(Subject * s, int i, float xx, float yy, float alarm)*:
Initialize id to I, x to xx, y to yy, soundAlarm to alarm using an initializer list. Subscribe to subject s using s's subscribe function.

Alarm::~Alarm:: provided

Void Alarm::notify(double xx, double yy):
Find the distance from the point (xx,yy) to the object's (x,y). If the distance is greater than soundAlarm, print out an alarm. Otherwise do nothing.
The C++ sqrt function can be used by including cmath and invoking with with std::sqrt. You do not need to use -lm as an option on your compile.

Main.cpp: it is provided

The output from my program is

```
obs full, size: 3, idx: 3
new map location: (1, 1)
Alarm1 Sounded!  Moved 1.41421 meters
new map location: (4, 4)
Alarm1 Sounded!  Moved 5.65685 meters
Alarm2 Sounded!  Moved 4.24264 meters
```

HW6b

For each line in main.cpp, say what is printed. If the line is an error, answer “Err”, if the line prints nothing, answer “Nothing”