# CS221: Data Structures Intro to Classes Homework

### Due – See canvas for due dates (10 points)

*You may collaborate with other students when working this homework. Each turn in your own copy to canvas.*

1. Define a class named **Clock.** A clock can store seconds, minutes and hour (24 hour form). Include a default constructor that sets the clock to 12:00:00
2. Write a method that returns the string “AM” or “PM” depending on the times stored in your clock.
3. Show a variable declaration statement that creates a variable of type clock. Show how you would call your method you wrote in part 2.
4. Define the term *encapsulation*.

#4:

Encapsulation is essentially bundling data and the methods that modify it together in a class, in order to restrict direct access to some or all this data. This allows other programmers to use this class without understanding its inner workings.

I answered 1, 2, and 3 in a short program I wrote. I have copied and pasted the code into this document, but I recommend reading it in full on my GitHub: <https://github.com/John-Ingram/CS_221/blob/master/HW1.cpp>

The code was compiled successfully on gcc version 9.3.0 (ubuntu)

1&2:

|  |
| --- |
|  |
|  |

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include <fstream> |
|  |

|  |
| --- |
| #include <iomanip> |
|  |

|  |
| --- |
| #include <string> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| using namespace std; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class Clock |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| private: |
|  |

|  |
| --- |
| int seconds, minutes, hours; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| // Override default constructor |
|  |

|  |
| --- |
| Clock(){ |
|  |

|  |
| --- |
| this->seconds = 0; |
|  |

|  |
| --- |
| this->minutes = 0; |
|  |

|  |
| --- |
| this->hours = 12; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // getters |
|  |

|  |
| --- |
| int getSeconds() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return this->seconds; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int getMinutes() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return this->minutes; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int getHours() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return this->hours; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // setters |
|  |

|  |
| --- |
| void setTime(int hours, int minutes, int seconds){ |
|  |

|  |
| --- |
| // verify inputs |
|  |

|  |
| --- |
| if(hours > 24 || hours < 0){ |
|  |

|  |
| --- |
| throw "Hour must be a value between 0 and 24"; |
|  |

|  |
| --- |
| // time can't go over 24:00:00 |
|  |

|  |
| --- |
| if(hours == 24 && minutes > 0 && seconds > 0){ |
|  |

|  |
| --- |
| throw "Value can't be over 24:00:00"; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| }else if (minutes > 60 || minutes < 0) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| throw "Minutes must be a value between 0 and"; |
|  |

|  |
| --- |
| }else if (seconds > 60 || seconds < 0) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| throw "Seconds must be a value between 0 and"; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| this->hours = hours; |
|  |

|  |
| --- |
| this->minutes = minutes; |
|  |

|  |
| --- |
| this->seconds = seconds; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| // Return AM or PM |
|  |

|  |
| --- |
| string naTime(){ |
|  |

|  |
| --- |
| if(hours < 12) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| return "AM"; |
|  |

|  |
| --- |
| }else{ |
|  |

|  |
| --- |
| return "PM"; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

};

#3:

|  |
| --- |
| int main(int argc, char const \*argv[]) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| Clock myClock; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| cout << myClock.naTime(); |
|  |

|  |
| --- |
| return 0; |
|  |

}