1. Define the Box class with the following fields: a length, a width and a height being real values, and a string describing the contents of the box (empty by default). Define internal exception classes EmptyError and CapacityError inheriting from the logic_error class. Implement the following public methods of the class:
   - a constructor with the box dimensions as the parameters,
   - getCapacity – computing the capacity of the box,
   - setContents – setting a given string as the contents of the box (empty string makes the box empty), the second parameter is the volume of the contents – if it is too large the CapacityError should be thrown,
   - getContents – returning the contents of the box (or throwing the EmptyError exception if the box is empty),
   - isEmpty – checking whether the box is empty,
   - countAll – returning the number of boxes available (it should be a static method),
   - the shift operator (<<) – printing the sizes and the contents of the box.

   Define the ColouredBox and the WeightedBox classes derived from the Box class. Implement the necessary constructors with additional parameters (a color of the box and the maximal weight of the box contents respectively) and proper getters and setters. Define additional exception class WeightError for the contents weight overflowed. Override the shift operator (<<) in each derived class. Add other members which are necessary to implement the classes. Write a program which tests all the classes.

2. Define the Clock class representing clocks operating on hours (0-23), minutes (0-59) and seconds (0-59). Define exception classes HourError, MinuteError and SecondError for time overflows. Implement the following public methods of the class:
   - the default constructor with three parameters corresponding to the time components (with zeros by default, throwing the exceptions if necessary),
   - set & get methods for the time components (i.e., getHours, setHours, etc., throwing the necessary exceptions for getters),
   - the shift operator (<<) to output the current time in a convenient way.

   Define the AlarmClock class inheriting from the Clock with an additional array/list of alarm times (empty by default). Implement the following public methods of the class:
   - the default constructor like in the base class,
   - addAlarm – adding a new alarm time for the clock,
   - eraseAlarms – erasing all the alarms set,
   - enableAlarms & disableAlarms – turning the alarms on/off,
   - the shift operator (<<) to show the current time and all the alarm times set.

   Define the RadioAlarmClock class inheriting from the AlarmClock with additional radio frequency (87,5-108 MHz). Define the exception class FrequencyError. Implement the following public methods of the class:
   - the default constructor like in the base class,
   - setFrequency & getFrequency methods for the radio frequency (throw the FrequencyError if necessary),
   - enableRadio & disableRadio – turning the radio on/off,
   - the shift operator (<<) to show all the time and radio settings.

   Add any other members which are necessary to implement the classes. Write a program which tests all the classes.
   As an alternative solution, try to define the RadioSet class representing the radio receiver with the proper methods and then define the RadioAlarmClock as derived from two base classes – the AlarmClock and the RadioSet.