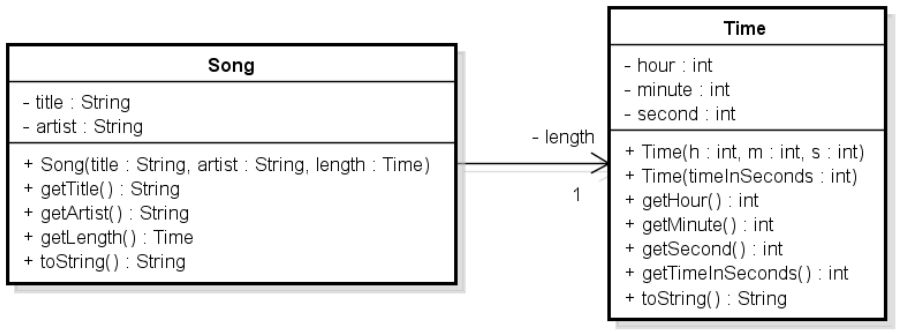
Exercise: Song_v1 (Association)

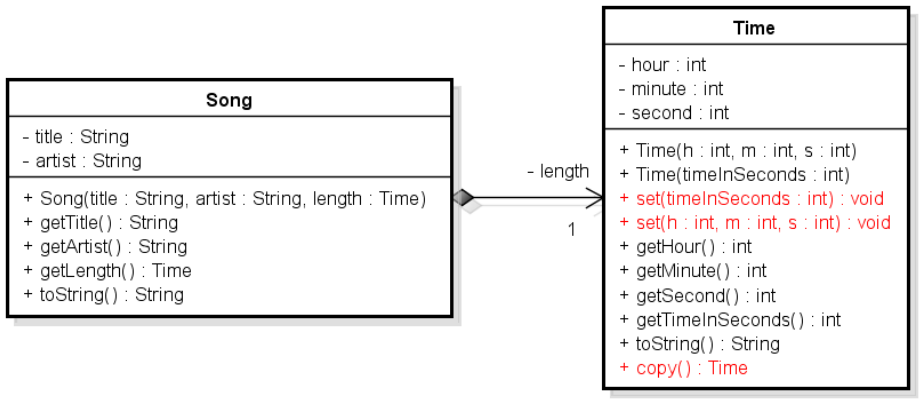
Implement the class `Song` exactly as shown in the UML class diagram below (Note: class `Time` is just a simple version of class `Clock` and is given in appendix at the end of this document):



Implement a test class with a `main` method in which you create at least one `Song` object and call getters and `toString` from class `Song`. Note: you need to create a `Time` object before creating a `Song` object.

Exercise: Song_v2 (Composition)

Modify class `Time` and class `Song` such that they exactly represent the UML class diagram below (Note that this time there is a composition relationship between `Song` and `Time`):

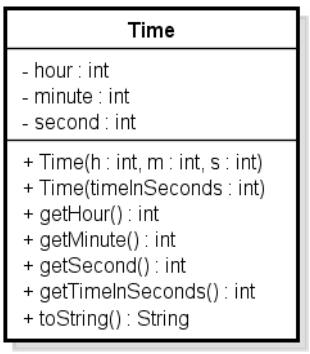


What is the difference between class `Song` from this and the previous exercise?
Why do you think that the previous exercise was with an association instead of a composition relationship?

Implement a test class with a `main` method in which you test your solution.

Appendix A – Class Time (for the association version)

The class `Time` represent a time in hour, minute and second (and is almost like the Clock class except that hour can now exceed 24):



The class `Time` is given in Java code below:

```
public class Time
{
   private int hour;
   private int minute;
   private int second;

   public Time(int hour, int minute, int second)
   {
      this(hour * 3600 + minute * 60 + second);
   }

   public Time(int timeInSeconds)
   {
      if (timeInSeconds < 0)
      {
         timeInSeconds = 0;
      }
      this.hour = timeInSeconds / 3600;
      this.minute = (timeInSeconds % 3600) / 60;
      this.second = (timeInSeconds % 3600) % 60;
   }

   public int getHour()
   {
      return hour;
   }

   public int getMinute()
   {
      return minute;
   }

   public int getSecond()
   {
      return second;
   }

   public int getTimeInSeconds()
   {
      return hour*3600 + minute*60 + second;
   }

   public String toString()
   {
      return String.format("%02d:%02d:%02d", hour, minute, second);
   }
}
```