**Assignment 4**

Problem Description:

Define a class named Time for encapsulating a time. The class contains the following:

1. A data field of the long time that stores the elapsed time since midnight, Jan 1, 1970.

2. A no-arg constructor that constructs a Time for the current time.

3. A constructor with the specified hour, minute, and second to create a Time.

4. A constructor with the specified elapsed time since midnight, Jan 1, 1970.

5. The getHour() method that returns the current hour in the range 0-23.

6. The getMinute() method that returns the current minute in the range 0-59.

7. The getSecond() method that returns the current second in the range 0-59.

8. The getSeconds() method that returns the elapsed total seconds.

9. The toString() method that returns a string such as "1 hour 2 minutes 1 second" and "14 hours 21 minutes 1 second".

10. Implement the Comparable<Time> interface to compare this Time with another one based on their elapse seconds. The compareTo method returns the difference between this object’s elapse seconds and the another’s.

11. Implement the Cloneable interface to clone a Time object.

Write a test program that produces the following sample run:

<Sample Run>

Enter time1 (hour minute second): 331 34 674 <Enter>

19 hours 45 minutes 14 seconds

Elapsed seconds in time1: 1194314

Enter time2 (elapsed time): 93889345 <Enter>

16 hours 22 minutes 25 seconds

Elapsed seconds in time2: 93889345

time1.compareTo(time2)? -92695031

time3 is created as a clone of time1

time1.compareTo(time3)? 0

<End Sample Run>

<Sample Run>

Enter time1 (hour minute second): 1 2 3 <Enter>

1 hour 2 minutes 3 seconds

Elapsed seconds in time1: 3723

Enter time2 (elapsed time): 193032 <Enter>

5 hours 37 minutes 12 seconds

Elapsed seconds in time2: 193032

time1.compareTo(time2)? -189309

time3 is created as a clone of time1

time1.compareTo(time3)? 0

<End Sample Run>

**Analysis:**

* In the given problem we have to create class named time for encapsulating time.

This class contains

1. A data field of long time that stores the elapsed time since midnight,Jan 1, 1970.
2. No argument constructor that constructs time for current time.
3. Constructor with specified hour, minute and seconds to create a time.
4. Constructor with specified elapsed time since midnight jan 1,1970.
5. getHour() method that returns the current hour in the range 0-23
6. getMinute() method that returns the current minute in range 0-59
7. getSecond () method that returns the current second in range 0-59
8. getSeconds () returns elapsed seconds.
9. The toString method that return a string
10. Implement the comparable<time> interface to compare this tme with another one based on their elapse seconds.
11. The compare to method must return the difference between this objects elapsed seconds and the another’s
12. Also implement the cloneable interface to clone the time object.

* We have to put the sample inputs from the user and calculate the elapsed time and display results.

**Design:**

**UML Diagram for the problem**

|  |
| --- |
| Time |
| -hours: long  -minutes: long  -seconds: long |
| +time(hours:long,minutes:long,seconds:int)  +gethour():long  +getminute():long  +getsecond():long  +getseconds():long  +time(time:long):void  +tostring():string  +comapareto(time )  +time clone() |

**Coding:**

/\* Basavraj Jaliminche

\* id- 8800149

\* Assignment 4

\* PROG8580-Computer Programming

\*/

**package** Assignment4;

//here time class implements comparable and clonable

**public** **class** time **implements** Comparable<time>, Cloneable {

**private** **long** t;

**public** time() {

t = System.*currentTimeMillis*();

}

// constructor spacified with hour,minute,seconds

**public** time(**long** hr, **long** min, **long** sec) {

t = ((((hr \* 60) + min) \* 60) + sec) \* 1000;

}

**public** **int** getHour() {

// hours in the between 0 to 23.

**return** (**int**) (t / (1000 \* 60 \* 60)) % 24;

}

**public** **int** getMinute() {

// minutes between 0 to 59

**return** (**int**) (t / (1000 \* 60)) % 60;

}

**public** **int** getSecond() {

// seconds between 0 to 59.

**return** (**int**) (t / 1000) % 60;

}

**public** **int** getSeconds() {

// returns the elapsed time seconds.

**return** (**int**) (t / 1000);

}

**public** time(**long** time) {

t = time \* 1000;

}

@Override

**public** String toString() {

**return** String.*format*("%d hours %d minutes %d seconds", getHour(), getMinute(), getSecond());

}

@Override

**public** **int** compareTo(time timeCompare) {

**return** (**int**) (**this**.getSeconds() - timeCompare.getSeconds());

}

// time clone throws exception

**public** time Clone() **throws** CloneNotSupportedException {

**return** (time) **super**.clone();

}

}

/\* Basavraj Jaliminche

\* id- 8800149

\* Assignment 4

\* PROG8580-Computer Programming

\*/

**package** Assignment4;

**import** java.util.Scanner;

**public** **class** encasptime {

**public** **static** **void** main(String[] args) **throws** CloneNotSupportedException {

// scanner class

Scanner sc = **new** Scanner(System.***in***);

// defining variables

**int** hrs, mins, secs;

// Taking input from the user

System.***out***.print("<sample run>\n");

System.***out***.print("Enter time1 (hour minute second): ");

hrs = sc.nextInt();

mins = sc.nextInt();

secs = sc.nextInt();

time Time1 = **new** time(hrs, mins, secs);

// for taking output used tostring method

System.***out***.println(Time1.toString());

System.***out***.println("Elapsed seonds in Time 1: " + Time1.getSeconds());

System.***out***.println("Enter Time 2 (Elapsed Time): ");

**long** mills = sc.nextInt();

time Time2 = **new** time(mills);

System.***out***.println(Time2);

System.***out***.println("Elapsed seconds in Time 2: " + Time2.getSeconds());

System.***out***.println("Time 1.compareTo(Time 2)? " + Time1.compareTo(Time2));

time T3 = Time1.Clone();

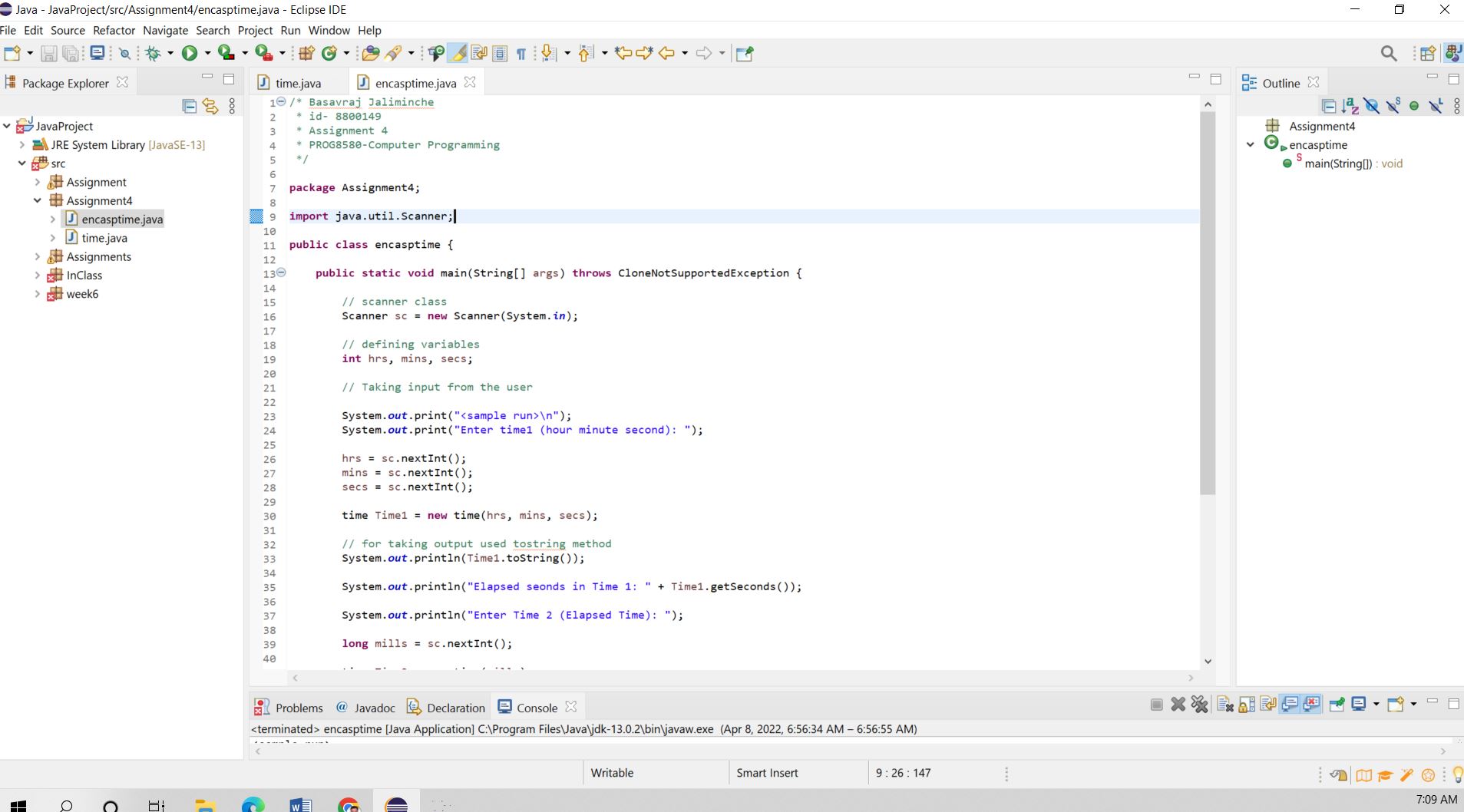
System.***out***.println("Time 3 is created as a clone of Time 1 Time1.compareTo(Time 3)? " + Time1.compareTo(T3));

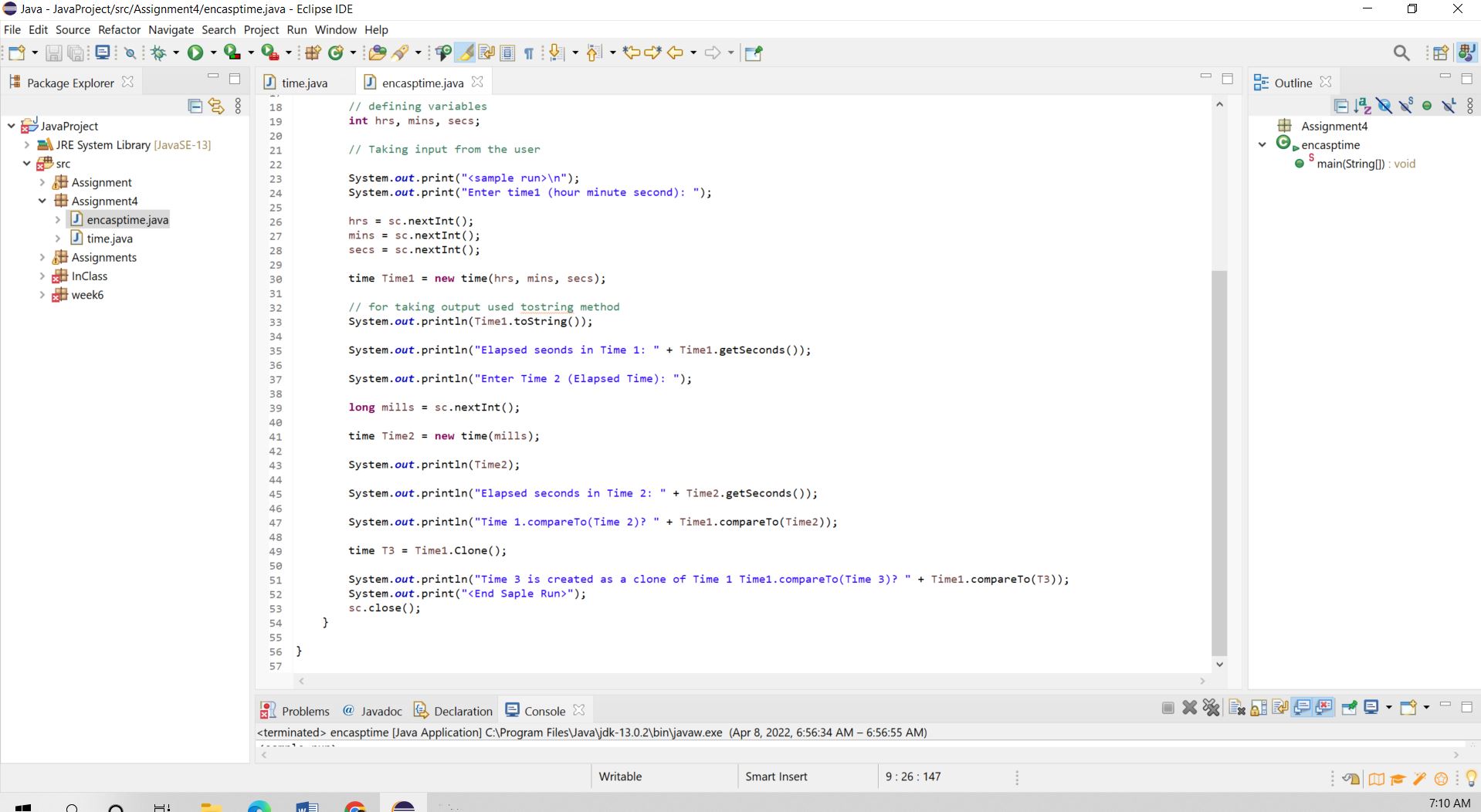
System.***out***.print("<End Saple Run>");

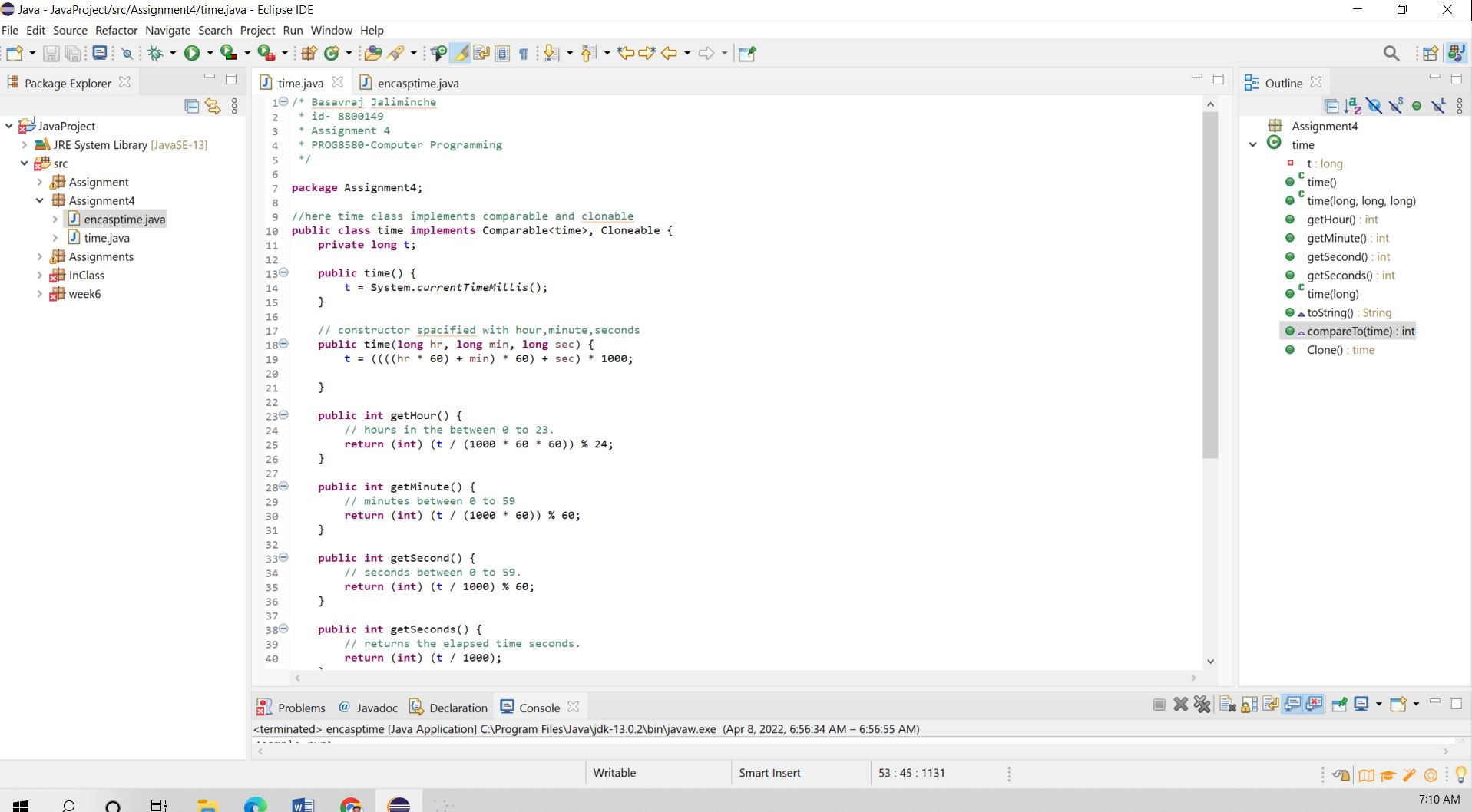
sc.close();

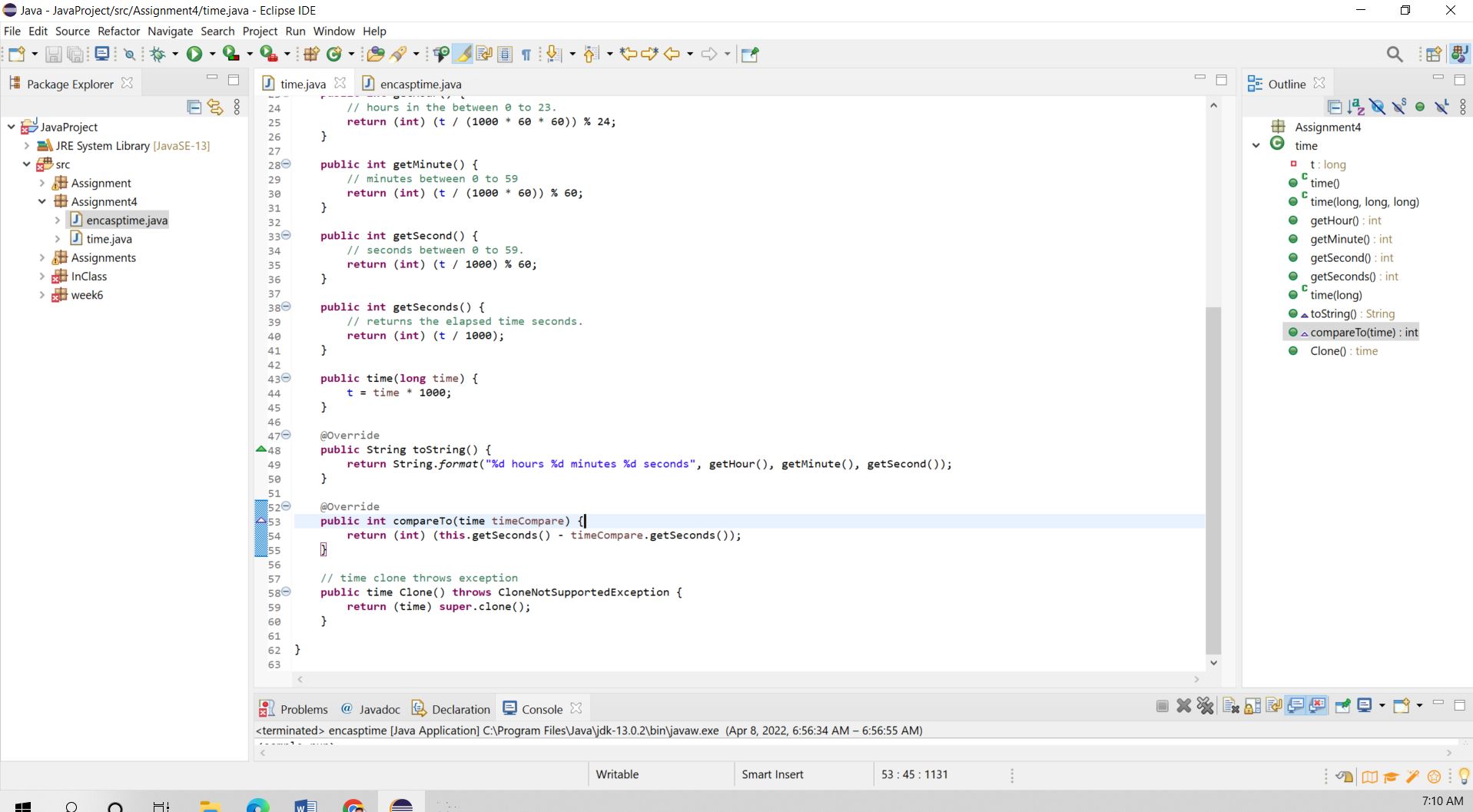
}

}



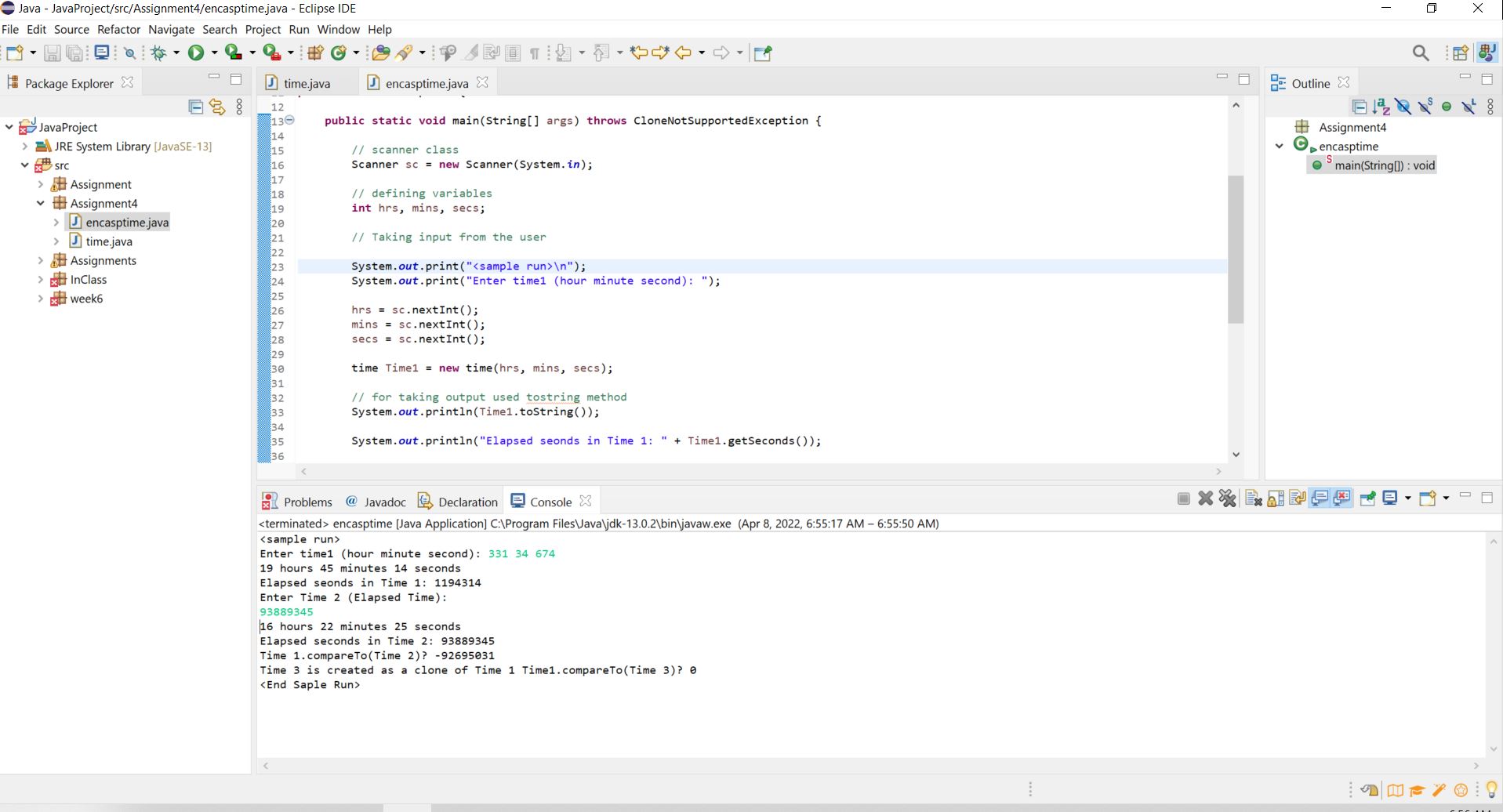






**Testing:**

Test 1:



Test2:

