

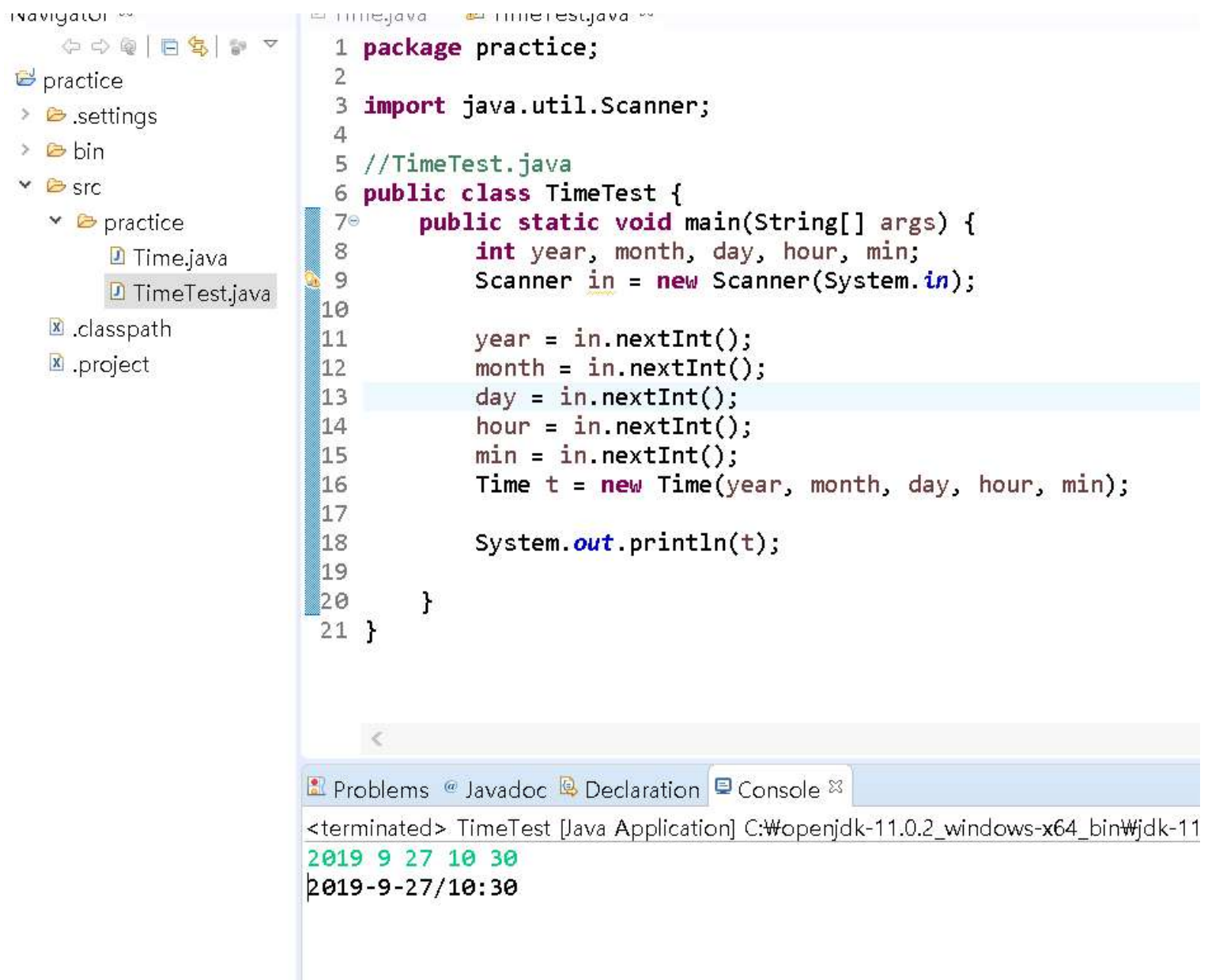
## 객체지향프로그래밍 -5주차 실습 활동지 (2019년 10월 1일)

성명: 안창희 학과: 소프트웨어 학번: 201723272 반/그룹: 318호

※ 본 실습활동지를 작성함에 있어 다른 학생의 문서로부터 일부 또는 전체를 복사하였습니까  
예( ) 아니오(O) (복사 하였다면 예에 체크하고 아니라면 아니오에 체크하시오)

1. 아래 코드 Time class (Time.java)를 이용하여 물음에 답하시오. (필요시 library import할 것)

가) 다음 코드는 위 Time class 객체를 하나 만들어 출력하는 프로그램이다. 아래 프로그램을 주어진 두가지 입력 데이터에 대해 실행해보고 문제점을 설명하시오. [실행화면 및 설명 포함] (2점)



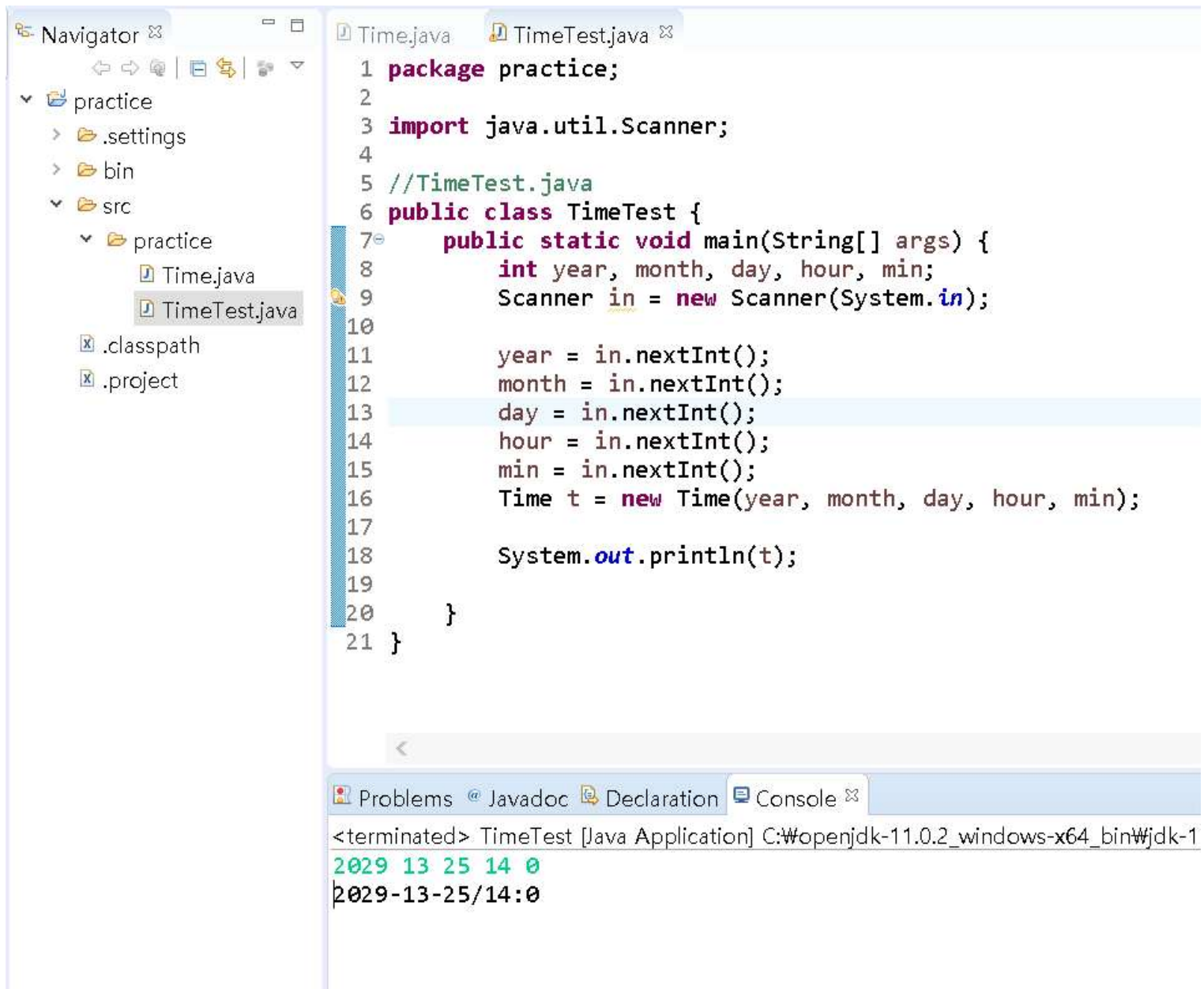
The screenshot shows an IDE with a project named 'practice'. The 'src' folder contains a 'practice' sub-folder with 'Time.java' and 'TimeTest.java'. The 'TimeTest.java' file is open, showing the following code:

```
1 package practice;
2
3 import java.util.Scanner;
4
5 //TimeTest.java
6 public class TimeTest {
7     public static void main(String[] args) {
8         int year, month, day, hour, min;
9         Scanner in = new Scanner(System.in);
10
11         year = in.nextInt();
12         month = in.nextInt();
13         day = in.nextInt();
14         hour = in.nextInt();
15         min = in.nextInt();
16         Time t = new Time(year, month, day, hour, min);
17
18         System.out.println(t);
19
20     }
21 }
```

The 'Console' tab at the bottom shows the output of the program:

```
<terminated> TimeTest [Java Application] C:\wopenjdk-11.0.2_windows-x64_bin\jdk-11
2019 9 27 10 30
2019-9-27/10:30
```

입력데이터 1 =>결과



## 입력데이터 2 =>결과

문제점: month는 1부터 12, day는 month에 따라 1부터 31, hour은 0부터 23, min은 0부터 59의 값을 가져야 하는데 이 모든 각각의 데이터 값에 대한 자료형이 int이고 입력에 대한 제한이 없어 무슨 값을 입력받더라도 년, 월, 일, 시, 분 에 해당하는 값에 입력받은 값이 들어가고 표현되게 된다. 따라서 각 데이터를 나열하여 표현할 수만 있고, 각 데이터의 입력을 받는 규격이나 처리에 대한 규칙이 없어 사람이 입력하고 난 후 프로그램 상에서는 그저 int형 데이터의 나열에 지나지 않게 되는 것이 문제이다. 따라서 이를 규격화 할 수 있는 규칙을 추가해야한다.

나) (가)의 문제점을 해결하기 위해 constructor에서 각 입력 값의 유효성을 체크하는 코드를 추가하고자 한다. 입력오류가 발생할 경우에 어떤 처리방안이 있는지 설명하시오. 단, Java의 Exception class를 사용하지 않음 [설명 포함] (3점)

문제점을 분석했을 때 떠오른 가장 간단한 처리방법은 각 입력 값에 대한 조건을 삽입하여 입력 형식이 년, 월, 일, 시, 분에 부합하는 숫자만 받아들이도록 하는 방법이었다.

구체적으로 말하자면, TimeTest에서 5개 변수를 가지고 constructor를 사용하여 Time object를 만들지만, object를 생성하는 과정에서 constructor 내에 if문을 통해 변수의 값이 정해진 조건을 만족하지 않을 경우 프로그램을 종료하거나 에러 문을 표시하는 것을 예로 들 수 있다.

다) (가)에서와 같은 입력오류가 발생하는 경우를 constructor내에서 처리하지 않고 main함수에서 처리하고자 한다. 아래 코드가 작동하도록 Time class에 checkValidity() 함수를 추가하시오. 힌트) staticmethod임. leap year 검사는 java.util.GregorianCalendar class의 isLeapYear() method를 이용해도 좋다. [checkValidity method 및 실행화면 포함] (5점)

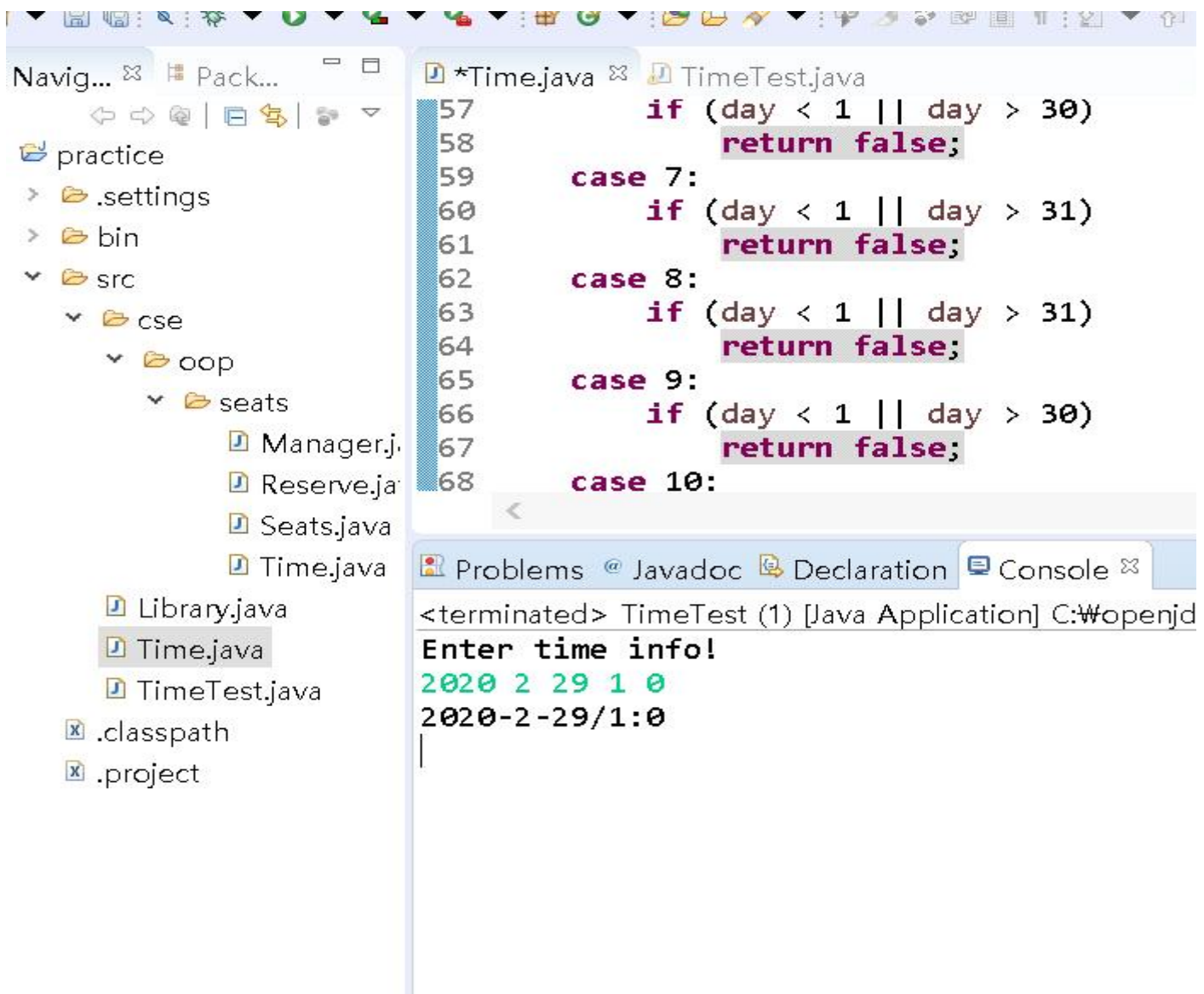
```
23
24
25 public static boolean checkValidity(int year, int month, int day, int hour, int min) {
26
27     // year가 2019~9999 사이의 수인가
28     if (year < 2019 || year > 9999)
29         return false;
30     // month가 1~12 사이의 수인가
31     if (month < 1 || month > 12)
32         return false;
33     // month 값에 대한 day 범위가 적절한가
34     switch (month) {
35     case 1:
36         if (day < 1 || day > 31)
37             return false;
38     case 2:
39         // year가 윤년인가 아닌가
40         GregorianCalendar gregori = new GregorianCalendar();
41         if (gregori.isLeapYear(year) == true) // 윤년일경우 29일까지
42             if (day < 1 || day > 29)
43                 return false;
44         if (gregori.isLeapYear(year) == false) // 윤년아닐경우 28일까지
45             if (day < 1 || day > 28)
46                 return false;
47     case 3:
48         if (day < 1 || day > 31)
49             return false;
50     case 4:
51         if (day < 1 || day > 30)
52             return false;
```

```
53     case 5:
54         if (day < 1 || day > 31)
55             return false;
56     case 6:
57         if (day < 1 || day > 30)
58             return false;
59     case 7:
60         if (day < 1 || day > 31)
61             return false;
62     case 8:
63         if (day < 1 || day > 31)
64             return false;
65     case 9:
66         if (day < 1 || day > 30)
67             return false;
68     case 10:
69         if (day < 1 || day > 31)
70             return false;
71     case 11:
72         if (day < 1 || day > 30)
73             return false;
74     case 12:
75         if (day < 1 || day > 31)
76             return false;
77 }
```

```

78 // hour값이 0~23인가
79 if (hour < 0 || hour > 23)
80     return false;
81 // min값이 0 또는 30인가
82 if (min != 0 && min != 30)
83     return false;
84 // 조건에 모두 부합하면 true 출력
85 return true;
86 }
87
88 }

```



결과 1, 윤년을 허용



Library.java  
Time.java  
TimeTest.java  
.classpath  
.project

```

69         if (day < 1 || day > 31)
70             return false;

```

Problems @ Javadoc Declaration Console

<terminated> TimeTest (1) [Java Application] C:\w\openjdk-11.0.2\_winc  
Enter time info!  
2029 13 25 14 0  
Error: enter time info again!  
  
2020 2 29 1 0  
2020-2-29/1:0

결과 2. 잘못된 데이터가 입력될 경우 다시 입력을 요청

라) 위 `checkValidity()` 함수가 `non-static` 함수로 정의한다면 어떨지에 대해 생각해 보시오. [설명 포함] (2점)

`checkValidity` 함수는 `main` 영역에서 데이터를 검사하여 `constructor`를 사용해 `Time` 클래스의 객체를 생성하기 이전에 데이터의 유효성 여부를 따지기 위해 만든 함수이다.

하지만 `non-static` 함수로 정의한다고 가정하면, 이미 만들어진 `Time` 클래스의 `object`를 필요로 하게 된다. 만약 이때 매개체가 되는 `object`가 새로 생성하려는 `object`인 경우 본래 목적인 `object`를 생성하기 전에 적합한 데이터를 거르는 조건에 부합하지 않게 된다.

혹은 다른 `object`를 매개로 쓸 경우 매개체인 `object`에서 가져온 데이터로 `checkValidity`를 구현하는 알고리즘에는 영향을 줄 사항이 없으므로 결국 `static`으로 구현한 방식과 비슷하게 매개변수를 쓰게 될 것이다.

따라서 애초에 구현 목적이 바뀌지 않는 이상 `checkValidity` 함수는 `static` 함수로 정의되어야 한다고 생각한다.

마) 아래 코드와 같은 방식으로 `Time` `object`를 생성하도록 `Time` class에 적절한 `constructor`를 추가하시오. 단, 추가하는 `constructor`는 기존의 `constructor` 코드를 호출할 것. 그리고 아래 코드를 테스트하도록 `TimeTest` class의 `main` 함수의 끝부분에 아래 코드를 삽입할 것. 힌트) `constructor overloading`을 사용할 것. [Constructor 코드 및 실행화면 포함] (3점)

\*Time.java TimeTest.java

```
10
11 public Time(int y, int m, int d, int hh, int mm) {
12     year = y;
13     month = m;
14     day = d;
15     hour = hh;
16     min = mm;
17 }
18
19 public Time(int y, int m, int d) {
20     year = y;
21     month = m;
22     day = d;
23     this.hour = 12;
24     this.min = 0;
25 }
26
```

Overloading한 코드 부분

The screenshot shows an IDE with a project explorer on the left, a code editor in the center, and a console at the bottom.

**Project Explorer:** Shows a project structure with folders like 'cse' and 'seats', and files like 'Manager.j', 'Reserve.ja', 'Seats.java', 'Time.java', and 'TimeTest.java'.

**Code Editor:** Displays the following Java code:

```
12     year = in.nextInt();
13     month = in.nextInt();
14     day = in.nextInt();
15     hour = in.nextInt();
16     min = in.nextInt();
17     valid = Time.checkValidity(year, month, day, hour, min);
18     if (valid == false)
19         System.out.println("Error: enter time info again!");
20 } while (!valid);
21 Time t = new Time(year, month, day, hour, min);
22 System.out.println(t);
23
24 Time t2 = new Time(year, month, day); // hh=12, mm=0 (default values)
25 System.out.println(t2);
26
```

**Console:** Shows the execution output:

```
<terminated> TimeTest (1) [Java Application] C:\openjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\javaw.exe (20
Enter time info!
2019 12 31 1 0
2019-12-31/1:0
2019-12-31/12:0
```

실행결과부분