**package** com.cts.test;

**import** java.util.List;

**import** java.util.Map;

**import** **static** org.junit.Assert.*assertEquals*;

**import** **static** org.junit.Assert.*assertTrue*;

**import** java.time.LocalDateTime;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Rule;

**import** org.junit.Test;

**import** org.junit.rules.ExpectedException;

**import** com.cts.model.DailyAttendance;

**import** com.cts.utility.MarkAttendance;

**import** com.cts.exception.InvalidAttendanceMarkingException;

**public** **class** MarkAttendanceTest {

@Rule

**public** ExpectedException exceptionRule = ExpectedException.*none*();

**static** MarkAttendance *markAttendanceObj*;

List<DailyAttendance> attendanceList=**new** ArrayList<>();

**static** DailyAttendance *d1*;

**static** DailyAttendance *d2*;

**static** DailyAttendance *d3*;

**static** DailyAttendance *d4*;

@Before

**public** **void** setUp() {

*markAttendanceObj* = **new** MarkAttendance();

// code here

//Create few objects of DailyAttendance and add those objects to attendanceList

//Set this list to the attendanceList in MarkAttendance class using setAttendanceList method

*d1*= **new** DailyAttendance(101,"nisha","java","yes",LocalDateTime.*parse*("2018-12-30T09:30"),LocalDateTime.*parse*("2018-12-30T12:30"));

*d2*= **new** DailyAttendance(102,"isha","spring","no",LocalDateTime.*parse*("2018-12-30T12:30"),LocalDateTime.*parse*("2018-12-30T15:00"));

*d3*= **new** DailyAttendance(102,"isha","java","no",LocalDateTime.*parse*("2019-12-31T09:30"),LocalDateTime.*parse*("2019-12-31T12:30"));

*d4*= **new** DailyAttendance(104,"disha","junit","yes",LocalDateTime.*parse*("2018-02-03T12:30"),LocalDateTime.*parse*("2018-02-02T15:00"));

attendanceList.add(*d1*);

attendanceList.add(*d2*);

attendanceList.add(*d3*);

attendanceList.add(*d4*);

*markAttendanceObj*.setAttendanceList(attendanceList);

}

@After

**public** **void** tearDown(){

//code here

}

// test the validateSchedule method when a valid Schedule Yes is passed as parameter to this method.

@Test

**public** **void** test11ValidateAsPerScheduleWhenYes() **throws** InvalidAttendanceMarkingException

{

*assertTrue*(*markAttendanceObj*.validateSchedule("yes"));

}

//test the validateSchedule method when a valid Schedule No is passed as parameter to this method.

@Test

**public** **void** test12ValidateAsPerScheduleWhenNo() **throws** InvalidAttendanceMarkingException

{

*assertTrue*(*markAttendanceObj*.validateSchedule("no"));

}

//test the validateSchedule method when an invalid Schedule is passed to this method.

@Test

**public** **void** test13ValidateAsPerScheduleWhenInvalid() **throws** InvalidAttendanceMarkingException

{

//code here

exceptionRule.expect(InvalidAttendanceMarkingException.**class**);

exceptionRule.expectMessage("Invalid AsPerSchedule");

*markAttendanceObj*.validateSchedule("y");

}

//test the addAttendance method when valid Schedule is provided for the DailyAttendance success

@Test

**public** **void** test14AddAttendanceForValidSchedule() **throws** InvalidAttendanceMarkingException

{

//code here

*assertTrue*(*markAttendanceObj*.addAttendance(104,"disha","junit","yes",LocalDateTime.*parse*("2018-02-04T09:30"),LocalDateTime.*parse*("2018-02-03T12:00")));

}

// test the addAttendance method when invalid Schedule is provided for the DailyAttendance. In this case, addAttendance method is expected to throw InvalidAttendanceMarkingException.

@Test

**public** **void** test15AddAttendanceForInValidSchedule() **throws** InvalidAttendanceMarkingException

{

exceptionRule.expect(InvalidAttendanceMarkingException.**class**);

exceptionRule.expectMessage("Invalid AsPerSchedule");

*markAttendanceObj*.addAttendance(105,"disha","junit","ye",LocalDateTime.*parse*("2018-02-03T09:30"),LocalDateTime.*parse*("2018-02-03T19:30"));

}

//test the viewEmployeeAttendanceById method when a EmployeeID is passed as parameter exists in the attendanceList.

@Test

**public** **void** test16viewEmployeeAttendanceByIdForValidId() **throws** InvalidAttendanceMarkingException

{

//code here

*assertEquals*(*d1*, *markAttendanceObj*.viewEmployeeAttendanceById(101));

}

//test the viewEmployeeAttendanceById method when a EmployeeID is passed as parameter does not exist in the attendanceList.

@Test

**public** **void** test17viewEmployeeAttendanceByIdForInValidId() **throws** InvalidAttendanceMarkingException

{

//code here

exceptionRule.expect(InvalidAttendanceMarkingException.**class**);

exceptionRule.expectMessage("Invalid Employee id");

*markAttendanceObj*.viewEmployeeAttendanceById(110);

}

// test the correctness of the viewAttendanceByDate method. Perform testing for the correctness of the list returned.

@Test

**public** **void** test18viewAttendanceByDate() **throws** InvalidAttendanceMarkingException

{

//code here

List<DailyAttendance> d=**new** ArrayList<DailyAttendance>();

d.add(*d1*);

*assertEquals*(d,*markAttendanceObj*.viewAttendanceByDate(LocalDateTime.*parse*("2018-12-30T09:30"),LocalDateTime.*parse*("2018-12-30T12:30")));

}

// test the correctness of the viewAttendanceByDate method. Perform testing for the not existing date and handle the exception

@Test

**public** **void** test19viewNotMarkedAttendanceParticularDate() **throws** InvalidAttendanceMarkingException

{

//code here

exceptionRule.expect(InvalidAttendanceMarkingException.**class**);

exceptionRule.expectMessage("No Matching Found");

*markAttendanceObj*.viewAttendanceByDate(LocalDateTime.*parse*("2018-12-30T09:30"),LocalDateTime.*parse*("2018-12-30T15:30") );

}

// test the correctness of viewSubjectExpertWise method. Perform testing for the correctness of the map returned.

@Test

**public** **void** test20viewSubjectExpertWise()

{

//code here

Map<String, List<DailyAttendance>> myMaps = **new** HashMap<String, List<DailyAttendance>>();

List<DailyAttendance> subexp1=**new** ArrayList<DailyAttendance>();

List<DailyAttendance> subexp2=**new** ArrayList<DailyAttendance>();

List<DailyAttendance> subexp3=**new** ArrayList<DailyAttendance>();

subexp1.add(*d1*);

subexp2.add(*d2*);

subexp2.add(*d3*);

subexp3.add(*d4*);

myMaps.put("nisha", subexp1);

myMaps.put("isha", subexp2);

myMaps.put("disha", subexp3);

*assertEquals*(myMaps, *markAttendanceObj*.viewSubjectExpertWise());

}

}