Aaren Patel, Elliot Niemann, Alay Patel

I pledge my honor that I have abided by the Stevens Honor System

GitHub for files as well: https://github.com/ENiems/CS-555-Refactor/tree/main

User Story

Title: Homework Planner		
Acceptance Test (hwScheduleTest): • Verify that tasks can be added • Verify that tasks can be completed • Verify that tasks can be deleted • Verify that due dates can be edited • Verify tasks can be seen • Verify that overdue tasks can be shown	Priority: 1	Story Points: 3
Description: As a student, I want to be able to track and update a planner to keep track of my tests and assignments due dates.		

Manual Test Cases

Manual Test #1:

Test Case ID: 1

Description: Verify that an assignment can be added with a date, marked as incomplete by default, marked as complete, and then verified as complete. Followed by deleting the task.

Preconditions: Empty task list, Date input is after the current date.

Steps:

1. Start the program
2. Select “Add upcoming assignment” from the menu.
3. Enter “Homework 1” for the assignment.
4. Enter “Math” for the subject.
5. Enter “12” for the month.
6. Enter “26” for the day.
7. Enter “2099” for the year.
8. Verify that the assignment list contains the assignment, the date, and the assignment is marked as incomplete by selecting “View upcoming schedule”.
9. Select “Mark assignment as completed” from the menu.
10. Enter “1” for the assignment we previously created.
11. Verify that the assignment list contains the same assignment marked as complete by selecting “View upcoming schedule”.
12. Select “Delete assignment” from the menu.
13. Enter “1” for the assignment we previously created.
14. Verify that the assignment list no longer contains any assignments by selecting “View upcoming schedule”.

Expected Result:

1. After step 8, the assignment should be added to the upcoming assignments as not submitted with the input date.
2. After step 11, the assignment should now be marked as submitted.
3. After step 14, the assignment list should be empty.

Manual Test #2:

Test Case ID: 2

Description: Verify that a test can be added with a date, marked as incomplete by default, marked as complete, and then verified as complete. Followed by deleting the task.

Preconditions: Empty task list, Date input is after the current date.

Steps:

1. Start the program
2. Select “Add upcoming test” from the menu.
3. Enter “CS 555 Test” for the assignment.
4. Enter “Computer Science” for the subject.
5. Enter “12” for the month.
6. Enter “26” for the day.
7. Enter “2099” for the year.
8. Verify that the assignment list contains the test, the date, and the assignment is marked as incomplete by selecting “View upcoming schedule”.
9. Select “Mark assignment as completed” from the menu.
10. Enter “1” for the test we previously created.
11. Verify that the assignment list contains the same test marked as complete by selecting “View upcoming schedule”.
12. Select “Delete assignment” from the menu.
13. Enter “1” for the assignment we previously created.
14. Verify that the assignment list no longer contains any assignments by selecting “View upcoming schedule”.

Expected Result:

1. After step 8, the test should be added to the upcoming assignments as not submitted with the input date.
2. After step 11, the test should now be marked as submitted.
3. After step 14, the assignment list should be empty.

Manual Test #3:

Test Case ID: 3

Description: Verify that an assignment can be seen as overdue only if the current date is past the due date, and the assignment was not submitted.

Preconditions: Empty task list, Date input for task 1 is after the current date, Date input for task 2 is before the current date.

Steps:

1. Start the program
2. Select “Add upcoming assignment” from the menu.
3. Enter “Homework 2” for the assignment.
4. Enter “Math” for the subject.
5. Enter “12” for the month.
6. Enter “26” for the day.
7. Enter “2099” for the year.
8. Select “Add upcoming assignment” from the menu.
9. Enter “Homework 1” for the assignment.
10. Enter “Math” for the subject.
11. Enter “12” for the month.
12. Enter “26” for the day.
13. Enter “1” for the year.
14. Select “Mark assignment as completed” from the menu.
15. Enter “1” for the assignment we created second.
16. Verify that the assignment list contains the assignment with Homework 2 marked as incomplete by selecting “View upcoming schedule”.
17. Verify that the overdue list contains no assignments as Homework 1 is complete by selecting “View overdue assignments”.
18. Select “Update assignment due date” from the menu.
19. Enter “2” for the assignment we created first.
20. Enter “12” for the month.

21. Enter “26” for the day.
22. Enter “5” for the year.
23. Verify that the overdue list contains only assignment Homework 2 by selecting “View overdue assignments”.

Expected Result:

1. After step 16, the assignment list contains the assignment titled Homework 2.
2. After step 17, the overdue list is empty.
3. After step 23, the overdue list contains the assignment titled Homework 2.

BAD CODE

Assignment.java [File for the Assignment object]

```java
import java.time.LocalDate;

public class Assignment {
	private String name;
	private String subject;
	private String type;
	private boolean completion;
	private LocalDate due;

	public Assignment(String assign, String subject, String type, int month, int day, int
year) {
		this.name = assign;
		this.subject = subject;
		this.type = type;
		this.completion = false;
		this.due = LocalDate.of(year, month, day);
	}

	public String getAssignment() {
		return name;
	}
	public void setTask(String input) {
		this.name = input;
	}
	public boolean getStatus() {
		return completion;
	}
	public void setStatus(boolean input) {
		completion = input;
	}
	public LocalDate getDue() {
		return due;
	}
	public void setDue(int month, int day, int year) {
		this.due = LocalDate.of(year, month, day);
	}
	public String getSubject() {
		return subject;
	}
	public String getType() {
		return type;
	}
}
```

homeworkSchedule.java [File that handles the schedule]

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.temporal.TemporalAdjusters;

public class homeworkSchedule {
    List<Assignment> schedule;
    public homeworkSchedule() { this.schedule = new ArrayList<>(); }

    public void addAssignment (String a, String b, int c, int d, int e) {
        schedule.add(new Assignment(a, b, "homework", c, d, e));
        sortSchedule();
    }

    public void addTest (String a, String b, int c, int d, int e) {
        schedule.add(new Assignment(a, b, "test", c, d, e));
        sortSchedule();
    }

    public void completeAssignment(int a) {
        if(a >= 1 && a <= schedule.size()) schedule.get((a - 1)).setStatus(true);
        else System.out.println("Invalid task number");
    }

    public void deleteAssignment(int a) {
        if(a >= 1 && a <= schedule.size()) schedule.remove((a - 1));
        else System.out.println("Invalid task number");
    }

    public void updateDate(int a, int b, int c, int d) {
        if(a >= 1 && a <= schedule.size()) schedule.get((a - 1)).setDue(b,c,d);
        else System.out.println("Invalid task number");
    }

    public void printFullList() {
        for(int i=0;i<schedule.size();i++) {
            Assignment curr = schedule.get(i);
            System.out.print((i+1) + " - " + curr.getAssignment());
            if(curr.getStatus()) System.out.println(" - Submitted");
            else System.out.println(" - Not Submitted");
        }
    }

    public void printUpcoming() {
        LocalDate now = LocalDate.now();
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
```

```
            LocalDate due = curr.getDue();
            if (due.isAfter(now) || due.isEqual(now)) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus()) System.out.println(" - Submitted");
                else System.out.println(" - Not Submitted");
            }
        }
    }

    public void printThisWeek() {
        LocalDate today = LocalDate.now();
        LocalDate start = today.with(TemporalAdjusters.previousOrSame(DayOfWeek.SUNDAY));
        LocalDate end = start.plusDays(6);
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if ((due.isAfter(start) || due.isEqual(start)) && (due.isBefore(end) ||
due.isEqual(end))) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus()) System.out.println(" - Submitted");
                else System.out.println(" - Not Submitted");
            }
        }
    }

    public void printThisMonth() {
        LocalDate today = LocalDate.now();
        LocalDate start = today.with(TemporalAdjusters.firstDayOfMonth());
        LocalDate end = today.with(TemporalAdjusters.lastDayOfMonth());
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if ((due.isAfter(start) || due.isEqual(start)) && (due.isBefore(end) ||
due.isEqual(end))) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus()) System.out.println(" - Submitted");
                else System.out.println(" - Not Submitted");
            }
        }
    }

    public void printOverdue() {
        LocalDate now = LocalDate.now();
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if (due.isBefore(now) && !curr.getStatus()) {
                System.out.println(curr.getDue() + " - " + curr.getAssignment());
            }
        }
    }
```

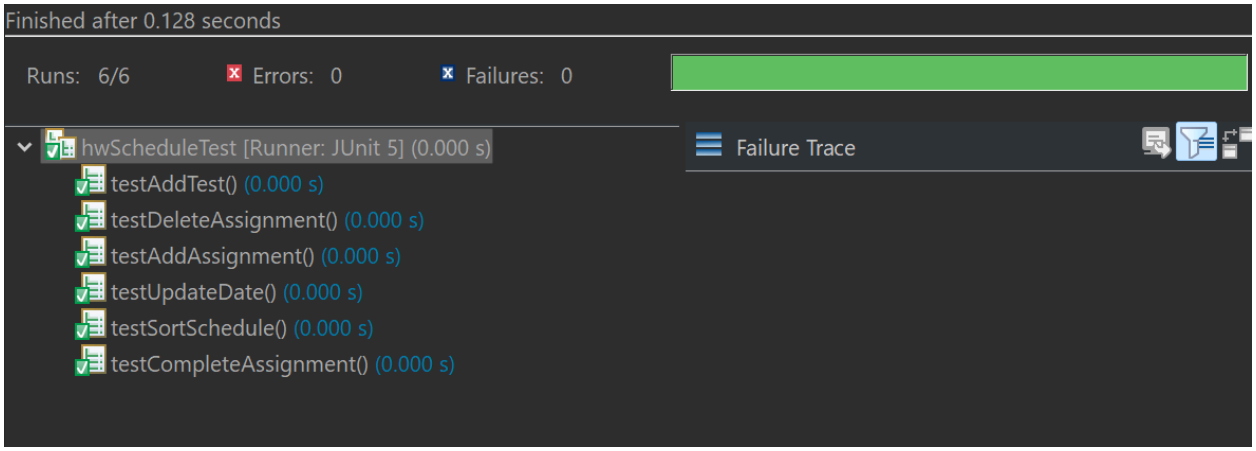
```
    private void sortSchedule() {
        Collections.sort(schedule, Comparator.comparing(Assignment::getDue));
    }
}
```

userInput.java [File that handles the user's input for a schedule]

```
import java.util.Scanner;

public class userInput {
    public static void main(String[] args) {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nChoose an action:");
            System.out.println("1. Add upcoming assignment");
            System.out.println("2. Add upcoming test");
            System.out.println("3. Mark assignment as submitted");
            System.out.println("4. Delete assignment");
            System.out.println("5. Update assignment due date");
            System.out.println("6. View upcoming schedule");
            System.out.println("7. View this week");
            System.out.println("8. View this month");
            System.out.println("9. View overdue assignments");
            System.out.println("10. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            System.out.println();
            switch (choice) {
            case 1:
              hwSchedule.printFullList();
               System.out.print("Enter assignment name: ");
               String assignName = scanner.nextLine();
               System.out.print("Enter subject: ");
               String subject = scanner.nextLine();
               System.out.print("Enter month (1-12): ");
               int month = scanner.nextInt();
               System.out.print("Enter day (1-31): ");
               int day = scanner.nextInt();
               System.out.print("Enter year: ");
               int year = scanner.nextInt();
               scanner.nextLine();
               hwSchedule.addAssignment(assignName, subject, month, day, year);
               break;
            case 2:
              hwSchedule.printFullList();
               System.out.print("Enter test name: ");
               String testName = scanner.nextLine();
               System.out.print("Enter subject: ");
               String testSubject = scanner.nextLine();
               System.out.print("Enter month (1-12): ");
               int testMonth = scanner.nextInt();
               System.out.print("Enter day (1-31): ");
               int testDay = scanner.nextInt();
               System.out.print("Enter year: ");
               int testYear = scanner.nextInt();
```

```
                scanner.nextLine();
                hwSchedule.addTest(testName, testSubject, testMonth, testDay, testYear);
                break;
            case 3:
               hwSchedule.printFullList();
                System.out.print("Enter assignment number to mark as submitted: ");
                int complNum = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.completeAssignment(complNum);
                break;
            case 4:
               hwSchedule.printFullList();
                System.out.print("Enter assignment number to delete: ");
                int delNum = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.deleteAssignment(delNum);
                break;
            case 5:
               hwSchedule.printFullList();
               System.out.print("Enter assignment number to update: ");
               int updateNum = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Enter new month (1-12): ");
                int newMonth = scanner.nextInt();
                System.out.print("Enter new day (1-31): ");
                int newDay = scanner.nextInt();
                System.out.print("Enter new year: ");
                int newYear = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.updateDate(updateNum, newMonth, newDay, newYear);
                break;
            case 6:
                hwSchedule.printUpcoming();
                break;
            case 7:
                hwSchedule.printThisWeek();
                break;
            case 8:
                hwSchedule.printThisMonth();
                break;
            case 9:
                hwSchedule.printOverdue();
                break;
            case 10:
               return;
            default:
               System.out.println("Invalid action");
            }
        }
    }
}
```

JUNIT TESTS




```java
import org.junit.jupiter.api.Test;
import java.time.LocalDate;
import static org.junit.jupiter.api.Assertions.*;

public class hwScheduleTest {

    @Test
    void testAddAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", "SSW555", 11, 18, 2024);
        assertEquals(1, hwSchedule.schedule.size());
        assertEquals("Homework 1", hwSchedule.schedule.get(0).getAssignment());
    }

    @Test
    void testAddTest() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addTest("Test 1", "SSW555", 11, 25, 2024);
        assertEquals(1, hwSchedule.schedule.size());
        assertEquals("Test 1", hwSchedule.schedule.get(0).getAssignment());
    }

    @Test
    void testCompleteAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", "SSW555", 11, 18, 2024);
        hwSchedule.completeAssignment(1);
        assertTrue(hwSchedule.schedule.get(0).getStatus());
    }

    @Test
    void testDeleteAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", "SSW555", 11, 18, 2024);
```

```
        hwSchedule.deleteAssignment(1);
        assertEquals(0, hwSchedule.schedule.size());
    }

    @Test
    void testUpdateDate() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", "SSW555", 11, 18, 2024);
        hwSchedule.updateDate(1, 11, 20, 2024);
        assertEquals(LocalDate.of(2024, 11, 20), hwSchedule.schedule.get(0).getDue());
    }

    @Test
    void testSortSchedule() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 2", "SSW555", 11, 25, 2024);
        hwSchedule.addAssignment("Homework 1", "SSW555", 11, 18, 2024);
        assertEquals("Homework 1", hwSchedule.schedule.get(0).getAssignment());
    }
}
```

Bad Smells

Smell #1:

Lack of comments to make code understandable

To Fix: Add neat concise comments

Smell #2:

homeworkSchedule.java has terrible variable names

To Fix: Rename the variables

Smell #3:

Excess variables in homeworkSchedule.java

To Fix: Remove the extra fields, [Subject]

New Manual Test Cases

Manual Test #1:

Test Case ID: 1

Description: Verify that an assignment can be added with a date, marked as incomplete by default, marked as complete, and then verified as complete. Followed by deleting the task.

Preconditions: Empty task list, Date input is after the current date.

Steps:

1. Start the program
2. Select “Add upcoming assignment” from the menu.
3. Enter “Homework 1” for the assignment.
4. Enter “12” for the month.
5. Enter “26” for the day.
6. Enter “2099” for the year.
7. Verify that the assignment list contains the assignment, the date, and the assignment is marked as incomplete by selecting “View upcoming schedule”.
8. Select “Mark assignment as completed” from the menu.
9. Enter “1” for the assignment we previously created.
10. Verify that the assignment list contains the same assignment marked as complete by selecting “View upcoming schedule”.
11. Select “Delete assignment” from the menu.
12. Enter “1” for the assignment we previously created.
13. Verify that the assignment list no longer contains any assignments by selecting “View upcoming schedule”.

Expected Result:

1. After step 7, the assignment should be added to the upcoming assignments as not submitted with the input date.
2. After step 10, the assignment should now be marked as submitted.
3. After step 13, the assignment list should be empty.

Manual Test #2:

Test Case ID: 2

Description: Verify that a test can be added with a date, marked as incomplete by default, marked as complete, and then verified as complete. Followed by deleting the task.

Preconditions: Empty task list, Date input is after the current date.

Steps:

1. Start the program
2. Select “Add upcoming test” from the menu.
3. Enter “CS 555 Test” for the assignment.
4. Enter “12” for the month.
5. Enter “26” for the day.
6. Enter “2099” for the year.
7. Verify that the assignment list contains the test, the date, and the assignment is marked as incomplete by selecting “View upcoming schedule”.
8. Select “Mark assignment as completed” from the menu.
9. Enter “1” for the test we previously created.
10. Verify that the assignment list contains the same test marked as complete by selecting “View upcoming schedule”.
11. Select “Delete assignment” from the menu.
12. Enter “1” for the assignment we previously created.
13. Verify that the assignment list no longer contains any assignments by selecting “View upcoming schedule”.

Expected Result:

1. After step 7, the test should be added to the upcoming assignments as not submitted with the input date.
2. After step 10, the test should now be marked as submitted.
3. After step 13, the assignment list should be empty.

Manual Test #3:

Test Case ID: 3

Description: Verify that an assignment can be seen as overdue only if the current date is past the due date, and the assignment was not submitted.

Preconditions: Empty task list, Date input for task 1 is after the current date, Date input for task 2 is before the current date.

Steps:

1. Start the program
2. Select “Add upcoming assignment” from the menu.
3. Enter “Homework 2” for the assignment.
4. Enter “12” for the month.
5. Enter “26” for the day.
6. Enter “2099” for the year.
7. Select “Add upcoming assignment” from the menu.
8. Enter “Homework 1” for the assignment.
9. Enter “12” for the month.
10. Enter “26” for the day.
11. Enter “1” for the year.
12. Select “Mark assignment as completed” from the menu.
13. Enter “1” for the assignment we created second.
14. Verify that the assignment list contains the assignment with Homework 2 marked as incomplete by selecting “View upcoming schedule”.
15. Verify that the overdue list contains no assignments as Homework 1 is complete by selecting “View overdue assignments”.
16. Select “Update assignment due date” from the menu.
17. Enter “2” for the assignment we created first.
18. Enter “12” for the month.
19. Enter “26” for the day.
20. Enter “5” for the year.

21. Verify that the overdue list contains only assignment Homework 2 by selecting “View overdue assignments”.

Expected Result:

4. After step 14, the assignment list contains the assignment titled Homework 2.
5. After step 15, the overdue list is empty.

After step 21, the overdue list contains the assignment titled Homework 2.

GOOD CODE

Assignment.java [File for the Assignment object]

```java
import java.time.LocalDate;

public class Assignment {
    // Private member variable to store the name of the assignment
    private String name;

    // Final member variable to store the type of assignment (cannot be changed once set)
    private final String type;

    // Boolean to track the completion status of the assignment (true if completed, false if not)
    private boolean completion;

    // Stores the due date of the assignment as a LocalDate object
    private LocalDate due;

    // Constructor to initialize assignment details including name, type, and due date
    public Assignment(String assign, String type, int month, int day, int year) {
        this.name = assign;
        this.type = type;
        this.completion = false; // Sets default completion status to false
        this.due = LocalDate.of(year, month, day); // Initializes due date
    }

    // Getter for the assignment name
    public String getAssignment() {
        return name;
    }

    // Setter to update the assignment name
    public void setTask(String input) {
        this.name = input;
    }

    // Getter for the assignment completion status
    public boolean getStatus() {
        return completion;
    }

    // Setter to update the completion status of the assignment
    public void setStatus(boolean input) {
        completion = input;
    }

    // Getter for the due date of the assignment
    public LocalDate getDue() {
        return due;
```

```
    }

    // Setter to update the due date of the assignment
    public void setDue(int month, int day, int year) {
        this.due = LocalDate.of(year, month, day);
    }

    // Getter for the assignment type (cannot be changed after initialization)
    public String getType() {
        return type;
    }
}
```

homeworkSchedule.java [File that handles the schedule]

```
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.temporal.TemporalAdjusters;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class homeworkSchedule {
    // List to store assignments in the schedule
    List<Assignment> schedule;

    // Constructor that initializes the schedule list
    public homeworkSchedule() {
        this.schedule = new ArrayList<>();
    }

    // Adds a homework assignment to the schedule and sorts by due date
    public void addAssignment(String assignment, int month, int date, int year) {
        schedule.add(new Assignment(assignment, "homework", month, date, year));
        sortSchedule();
    }

    // Adds a test assignment to the schedule and sorts by due date
    public void addTest(String assignment, int month, int date, int year) {
        schedule.add(new Assignment(assignment, "test", month, date, year));
        sortSchedule();
    }

    // Marks an assignment as completed based on its number in the list
    public void completeAssignment(int assignmentNum) {
        if (assignmentNum >= 1 && assignmentNum <= schedule.size())
            schedule.get(assignmentNum - 1).setStatus(true);
        else
            System.out.println("Invalid task number");
    }

    // Removes an assignment from the schedule based on its number in the list
    public void deleteAssignment(int assignmentNum) {
        if (assignmentNum >= 1 && assignmentNum <= schedule.size())
            schedule.remove(assignmentNum - 1);
        else
            System.out.println("Invalid task number");
    }

    // Updates the due date of a specific assignment in the list
    public void updateDate(int assignmentNum, int month, int date, int year) {
        if (assignmentNum >= 1 && assignmentNum <= schedule.size())
            schedule.get(assignmentNum - 1).setDue(month, date, year);
        else
```

```
            System.out.println("Invalid task number");
    }

    // Prints a full list of assignments with their submission status
    public void printFullList() {
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            System.out.print((i + 1) + " - " + curr.getAssignment());
            if (curr.getStatus())
                System.out.println(" - Submitted");
            else
                System.out.println(" - Not Submitted");
        }
    }

    // Prints only upcoming assignments, including today's, with their status
    public void printUpcoming() {
        LocalDate now = LocalDate.now();
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if (due.isAfter(now) || due.isEqual(now)) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus())
                    System.out.println(" - Submitted");
                else
                    System.out.println(" - Not Submitted");
            }
        }
    }

    // Prints assignments due this week, from Sunday to Saturday, with their status
    public void printThisWeek() {
        LocalDate today = LocalDate.now();
        LocalDate start = today.with(TemporalAdjusters.previousOrSame(DayOfWeek.SUNDAY));
        LocalDate end = start.plusDays(6);
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if ((due.isAfter(start) || due.isEqual(start)) && (due.isBefore(end) ||
due.isEqual(end))) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus())
                    System.out.println(" - Submitted");
                else
                    System.out.println(" - Not Submitted");
            }
        }
    }

    // Prints assignments due this month, with their status
    public void printThisMonth() {
```

```
        LocalDate today = LocalDate.now();
        LocalDate start = today.with(TemporalAdjusters.firstDayOfMonth());
        LocalDate end = today.with(TemporalAdjusters.lastDayOfMonth());
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if ((due.isAfter(start) || due.isEqual(start)) && (due.isBefore(end) ||
due.isEqual(end))) {
                System.out.print(curr.getDue() + " - " + curr.getAssignment());
                if (curr.getStatus())
                    System.out.println(" - Submitted");
                else
                    System.out.println(" - Not Submitted");
            }
        }
    }

    // Prints overdue assignments that have not been completed
    public void printOverdue() {
        LocalDate now = LocalDate.now();
        for (int i = 0; i < schedule.size(); i++) {
            Assignment curr = schedule.get(i);
            LocalDate due = curr.getDue();
            if (due.isBefore(now) && !curr.getStatus()) {
                System.out.println(curr.getDue() + " - " + curr.getAssignment());
            }
        }
    }

    // Sorts the assignments in the schedule by their due date
    private void sortSchedule() {
        Collections.sort(schedule, Comparator.comparing(Assignment::getDue));
    }
}
```

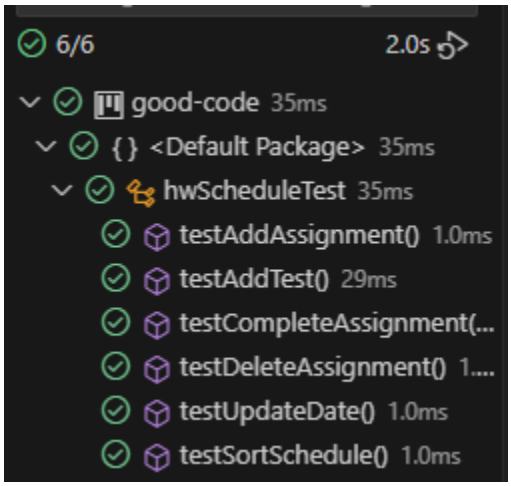
userInput.java [File that handles the user's input for a schedule]

```
import java.util.Scanner;

public class userInput {
    public static void main(String[] args) {
        // Create an instance of homeworkSchedule to manage assignments
        homeworkSchedule hwSchedule = new homeworkSchedule();
        Scanner scanner = new Scanner(System.in);

        // Main program loop, providing options for the user to interact with the schedule
        while (true) {
            System.out.println("\nChoose an action:");
            System.out.println("1. Add upcoming assignment");
            System.out.println("2. Add upcoming test");
            System.out.println("3. Mark assignment as submitted");
            System.out.println("4. Delete assignment");
            System.out.println("5. Update assignment due date");
            System.out.println("6. View upcoming schedule");
            System.out.println("7. View this week");
            System.out.println("8. View this month");
            System.out.println("9. View overdue assignments");
            System.out.println("10. Exit");
            System.out.print("Enter your choice: ");

            // Get the user's choice and clear the line for any next input
            int choice = scanner.nextInt();
            scanner.nextLine();
            System.out.println();

            // Perform action based on user choice
            switch (choice) {
                case 1: // Add a new homework assignment
                    hwSchedule.printFullList();
                    System.out.print("Enter assignment name: ");
                    String assignName = scanner.nextLine();
                    System.out.print("Enter month (1-12): ");
                    int month = scanner.nextInt();
                    System.out.print("Enter day (1-31): ");
                    int day = scanner.nextInt();
                    System.out.print("Enter year: ");
                    int year = scanner.nextInt();
                    scanner.nextLine();
                    hwSchedule.addAssignment(assignName, month, day, year);
                    break;

                case 2: // Add a new test
                    hwSchedule.printFullList();
                    System.out.print("Enter test name: ");
                    String testName = scanner.nextLine();
                    System.out.print("Enter month (1-12): ");
                    int testMonth = scanner.nextInt();
```

```
                System.out.print("Enter day (1-31): ");
                int testDay = scanner.nextInt();
                System.out.print("Enter year: ");
                int testYear = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.addTest(testName, testMonth, testDay, testYear);
                break;

            case 3: // Mark an assignment as submitted
                hwSchedule.printFullList();
                System.out.print("Enter assignment number to mark as submitted: ");
                int complNum = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.completeAssignment(complNum);
                break;

            case 4: // Delete an assignment
                hwSchedule.printFullList();
                System.out.print("Enter assignment number to delete: ");
                int delNum = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.deleteAssignment(delNum);
                break;

            case 5: // Update the due date of an assignment
                hwSchedule.printFullList();
                System.out.print("Enter assignment number to update: ");
                int updateNum = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Enter new month (1-12): ");
                int newMonth = scanner.nextInt();
                System.out.print("Enter new day (1-31): ");
                int newDay = scanner.nextInt();
                System.out.print("Enter new year: ");
                int newYear = scanner.nextInt();
                scanner.nextLine();
                hwSchedule.updateDate(updateNum, newMonth, newDay, newYear);
                break;

            case 6: // View upcoming assignments
                hwSchedule.printUpcoming();
                break;

            case 7: // View assignments due this week
                hwSchedule.printThisWeek();
                break;

            case 8: // View assignments due this month
                hwSchedule.printThisMonth();
                break;

            case 9: // View overdue assignments
```

```
                    hwSchedule.printOverdue();
                    break;

                case 10: // Exit the program
                    return;

                default: // Handle invalid inputs
                    System.out.println("Invalid action");
            }
        }
    }
}
```

JUNIT TESTS




```java
import org.junit.jupiter.api.Test;
import java.time.LocalDate;
import static org.junit.jupiter.api.Assertions.*;

public class hwScheduleTest {

    @Test
    void testAddAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", 11, 18, 2024);
        assertEquals(1, hwSchedule.schedule.size());
        assertEquals("Homework 1", hwSchedule.schedule.get(0).getAssignment());
    }

    @Test
    void testAddTest() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addTest("Test 1", 11, 25, 2024);
        assertEquals(1, hwSchedule.schedule.size());
        assertEquals("Test 1", hwSchedule.schedule.get(0).getAssignment());
    }

    @Test
    void testCompleteAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
```

```
        hwSchedule.addAssignment("Homework 1", 11, 18, 2024);
        hwSchedule.completeAssignment(1);
        assertTrue(hwSchedule.schedule.get(0).getStatus());
    }

    @Test
    void testDeleteAssignment() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", 11, 18, 2024);
        hwSchedule.deleteAssignment(1);
        assertEquals(0, hwSchedule.schedule.size());
    }

    @Test
    void testUpdateDate() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 1", 11, 18, 2024);
        hwSchedule.updateDate(1, 11, 20, 2024);
        assertEquals(LocalDate.of(2024, 11, 20),
hwSchedule.schedule.get(0).getDue());
    }

    @Test
    void testSortSchedule() {
        homeworkSchedule hwSchedule = new homeworkSchedule();
        hwSchedule.addAssignment("Homework 2", 11, 25, 2024);
        hwSchedule.addAssignment("Homework 1", 11, 18, 2024);
        assertEquals("Homework 1", hwSchedule.schedule.get(0).getAssignment());
    }
}
```