HW11 – Unit Testing

**Bugs identified in Clock.java:**

1. **addHour ():**

* **Clock.java:** The addHour method checks if **hours >= 24** before setting **hours** to 0. This is incorrect because **hours** should never reach 24 in a 24-hour format.

public void addHour () {

if (hours >= 24) {

hours = 0;

}

else {

hours++;

}

}

* **CorrectedClock.java:** The check should be **if (hours >= 23)** before incrementing, then after incrementing, **if (hours >= 24)** it should reset to 0.

public void addHour() {

hours++;

if (hours >= 24) {

hours = 0;

}

}

1. **addMinute () and addSeconds():**

* **Clock.java:** Both methods have a recursive call to **addMinute()** when seconds reach 60, which should not happen. Instead, **addMinute()** should be called when seconds reach 60, and similarly, **addHour()** should be called when minutes reach 60.

public void addMinute() {

if (seconds >= 60) {

seconds = 0;

addMinute();

}

else {

seconds++;

}

}

public void addSecond() {

if (seconds >= 60) {

seconds = 0;

addMinute();

}

else {

seconds++;

}

}

* **CorrectedClock**.**java:** In **addMinute()**, it should increment minutes and only call **addHour()** if minutes hit 60 after incrementing. Similarly, **addSecond()** should increment seconds and only call **addMinute()** if seconds hit 60 after incrementing.

public void addMinute() {

if (minutes >= 59) {

minutes = 0;

addHour();

} else {

minutes++;

}

}

public void addSecond() {

if (seconds >= 59) {

seconds = 0;

addMinute();

} else {

seconds++;

}

}

1. **get12HourFormat ():**

* **Clock.java**: The **get12HourFormat** method modifies the instance variable **hours** directly, which would alter the state of the clock object permanently after calling this method.

public String get12HourFormat () {

String format12;

if (hours == 0) {

hours = 12;

}

else if (hours > 12) {

hours = hours - 12;

}

format12 = hours + ":" + pad(minutes) + ":" + pad(seconds) + " " + getSuffix ();

return format12;

}

* **CorrectedClock.java:** It should use a temporary variable to calculate the 12-hour format to avoid modifying the state of the object.

public String get12HourFormat() {

String format12;

int inputtedHours = hours;

if (inputtedHours == 0) {

inputtedHours = 12;

}

else if (inputtedHours > 12) {

inputtedHours = inputtedHours - 12;

}

format12 = pad(inputtedHours) + ":" + pad(minutes) + ":" + pad(seconds) + " " + getSuffix();

return format12;

}

1. **pad ():**

* **Clock.java:** The **pad** method checks **if (value < 9)**, which works correctly for values 0-8, but 9 is also a single-digit number that should be padded.

private String pad (int value) {

String paddedValue;

if (value < 9) {

paddedValue = "0" + value;

}

else {

paddedValue = String.*valueOf*(value);

}

return paddedValue;

}

* **CorrectedClock.java:** The condition should be if **(value <= 9)** to include 9 in the padding process.

private String pad (int value) {

String paddedValue;

if (value <= 9) {

paddedValue = "0" + value;

}

else {

paddedValue = String.*valueOf*(value);

}

return paddedValue;

}

1. **Invalid Time Initialization:**

* **Clock.java:** The constructor allows for setting any integers to hours, minutes, and seconds without validation, meaning you can initialize times like 25:61:61, which are invalid.

public Clock(int hours, int minutes, int seconds) {

this.hours = hours;

this.minutes = minutes;

this.seconds = seconds;

}

* **CorrectedClock.java:** There should be checks to ensure that **hour** is between 0-23, **minute** and **second** are between 0-59. If invalid, it should throw an exception.

public CorrectedClock(int hours, int minutes, int seconds) {

setHours(hours);

setMinutes(minutes);

setSeconds(seconds);

}

public void setHours(int hours) {

if(hours >= 0 && hours < 24) {

this.hours = hours;

} else {

throw new IllegalArgumentException("Hour must be between 0 and 23");

}

}

public void setMinutes(int minutes) {

if(minutes >= 0 && minutes < 60) {

this.minutes = minutes;

} else {

throw new IllegalArgumentException("Minute must be between 0 and 59");

}

}

public void setSeconds(int seconds) {

if(seconds >= 0 && seconds < 60) {

this.seconds = seconds;

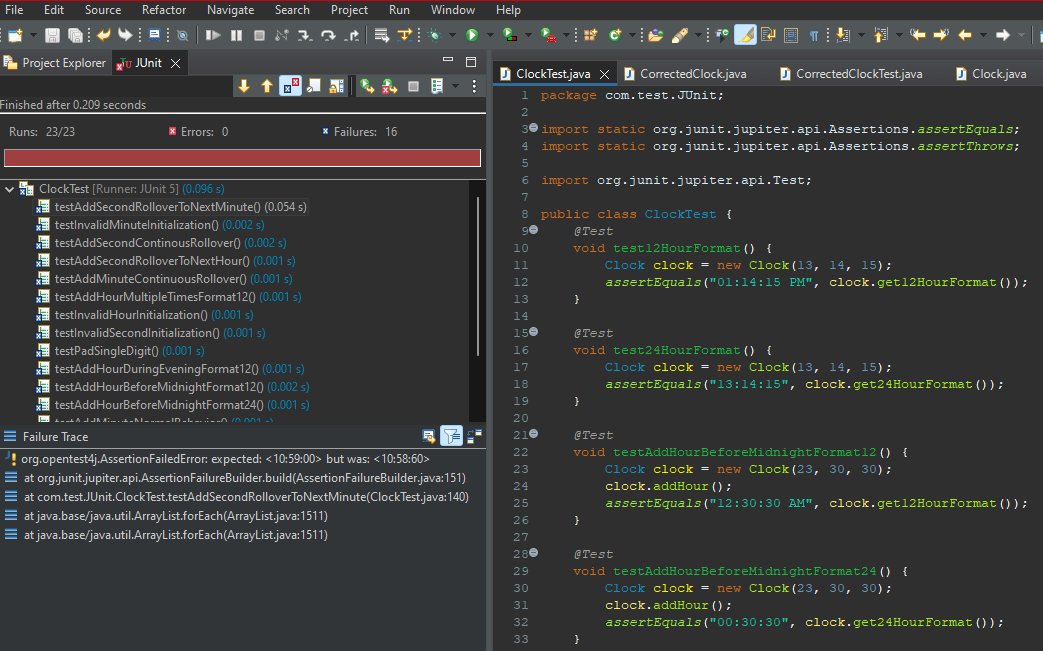
} else {

throw new IllegalArgumentException("Second must be between 0 and 59");

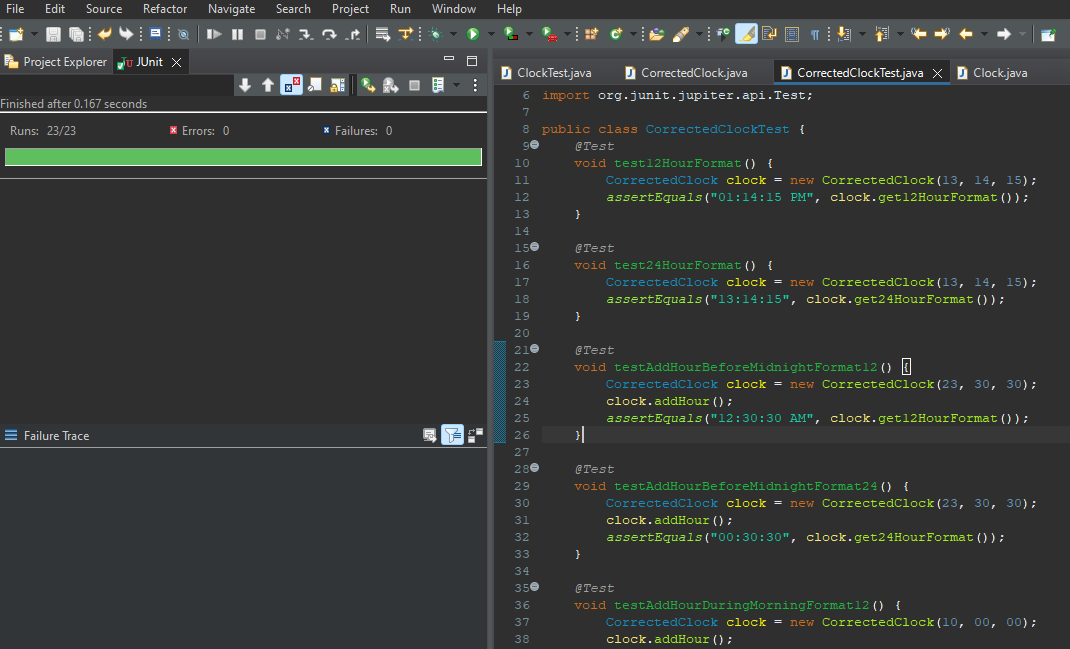
}

}

**jUnit Testing of Clock.java:**

****

**jUnit Testing of Corrected.java:**

****