Contrôle continu 1 (50% der Endnote):

Aufgabe 1: Fachklassen Thermometerlogik, JUnit5

Schreiben Sie eine Fachklasse für ein Thermometer. Gehen Sie dabei testgetrieben vor.

Von der Fachklasse Thermometer darf es nur genau eine Instanz geben. Die Thermometer-Klasse kann Temperaturwerte von -20° C bis 50°C mit zehntel Genauigkeit verarbeiten, sofern diese im Modus „aktiv“ ist. Temperaturwerte außerhalb dieses Intervalls oder mit höherer Genauigkeit dürfen nicht verarbeitet werden. Liegen Temperaturwerte vor, so kann die Fachklasse Thermometer die aktuelle, die maximale, die minimale Temperatur und die Durchschnittstemperatur seit Messaufzeichnung liefern. Eine neue Messaufzeichnung startet, wenn das Thermometer aktiv geschaltet oder zurückgesetzt wurde. Das Thermometer kann auch wieder deaktiviert werden.

Halten Sie schriftlich fest, was gut bei dieser Programmieraufgabe lief, welche Punkte Schwierigkeiten machten und was Sie bei künftigen Programmieraufgaben anders machen würden.

Beispielhafte Testfälle

Eingangstemperaturwerte	**Verhalten Fachklasse**
Bei aktiven Thermometer die Temperaturen 11.1, 22.2, 33.3 setzen	Aktuelle Temperatur: 33.3 Maximale Temperatur: 33.3 Minimale Temperatur: 11.1 Durchschnittstemperatur: 22.2
Bei aktiven Thermometer die Temperatur -21 setzen	Fehler: kein gültige Temperatur
Thermometer inaktiv, Temperatur 20 setzen	Fehler: inaktiv
Thermometer inaktiv, Abfrage Max	Fehler: inaktiv
Thermometer aktiv ohne gültige Temperatur, Abfrage Min	Fehler: keine Temperaturwerte

Tabelle 1: Beispielhafte Testfälle

Beurteilungskriterien

- Qualität der Implementierung der Fachklasse (Singleton, Fehlerbehandlung, Datenhaltung, Schnittstellendesign)
- Qualität der Testimplementierung (Vollständigkeit der Testfälle, Code-Qualität)
- Reflexion TDD

Alternative Aufgabe 1: Fachklassen Protokollnachrichten, JUnit5

An einem Raspberry Pi sind drei normale LEDs und eine RGB-LED angeschlossen. Es soll möglich sein, jede LED einzeln über textbasierte Befehle blinken zu lassen oder das Blinken zu stoppen. Zusätzlich kann die Farbe der RGB-LED geändert werden.

Eine Beschreibung der drei Protokollnachrichten ist in Tabelle 2, Tabelle 3 und Tabelle 4 zu sehen.

Nachrichtenname / Message: BlinkOnRequest			
Comment: Nach Erhalt eines BlinkOnRequest wird die LED mit der ID led mit der An- und Auszeitdauer, wie diese in duration angegeben ist, blinken. Blinkt die LED bereits, so wird die Blinkfrequenz angepasst.			
Parametername	**Type**	**Constraints**	**Comment**
led	ganze Zahl	[0, 2]	Identifikationsnummer der angeschlossenen LED
duration	ganze Zahl	[1, 30]	duration x 100 ms ist die Zeit, die die LED im Wechsel an und aus ist

Tabelle 2 : abstrakte Beschreibung der Nachricht "Blink-An-Anforderung"

Nachrichtenname / Message: BlinkOffRequest			
Comment: Das Blinken der LED mit der ID led wird gestoppt. Blinkt die LED nicht, so wird der Request ignoriert.			
Parametername	**Type**	**Constraints**	**Comment**
led	ganze Zahl	[0, 2]	Identifikationsnummer der angeschlossenen LED

Tabelle 3: abstrakte Beschreibung der Nachricht "Blink-Aus-Anforderung"

Message/Nachrichtenname	RGBColorRequest		
Comment	Nach Erhalt eines RGBColorRequest lässt der Arduino die RGB LED in der angegebenen Farbe leuchten.		
Parameter	**Type**	**Constraints**	**Comment**
red	Integer	[0,255]	Rotanteil der Farbe
green	Integer	[0,255]	Grünanteil der Farbe
blue	Integer	[0,255]	Blauanteil der Farbe

Tabelle 4: Abstrakte Syntax Anfrage für eine RGB-LED

Die Übertragung der Nachrichten findet textbasiert statt. Das Format ist in der BNF in Tabelle 5 beschrieben.

```
        <Message> ::= <BlinkOnRequest> | <BlinkOffRequest> | <RGBColorRequest>
 <BlinkOnRequest> ::= <START_CHARACTER>'On'<space><led><space><duration><STOP_CHARACTER>
<BlinkOffRequest> ::= <START_CHARACTER >'Off'<space><led><STOP_CHARACTER>
<RGBColorRequest> ::= <START_CHARACTER >'R'<red>'G'<green>'B'<blue><STOP_CHARACTER>
<START_CHARACTER> ::= '#'
 <STOP_CHARACTER> ::= '!'
          <space> ::= ' '
            <led> ::= '0' | '1' | '2'
       <duration> ::= '1' | '2' | ... | '29' | '30'
            <red> ::= <color_value>
          <green> ::= <color_value>
           <blue> ::= <color_value>
    <color_value> ::= '0' | '1' | ... | '254' | '255'
```

Tabelle 5 : Backus-Naur-Form der Transfersyntax des Protokolls

Schreiben Sie drei Nachrichtenklassen, die ein Message-Interface mit den Methoden encode und decode Implementieren. Die encode-Methode liefert die textuelle Repräsentation der Nachricht, dem Encoding, so wie diese übertragen wird (z. B. in einem UDP-Paket, einem http-Body, …). Die decode-Methode erzeugt eine Instanz der Nachrichtenklasse aus dem Encoding. Schreiben Sie Unit-Tests für die Nachrichtenklassen. Z. B. sollte die encode-Methode eines BlinkOnRequest für die LED 2 mit 500 ms den Text „#On 2 5!“ liefern.

Halten Sie schriftlich fest, was gut bei dieser Programmieraufgabe lief, welche Punkte Schwierigkeiten machten und was Sie bei künftigen Programmieraufgaben anders machen würden.

Stellen Sie sich vor, dass zusätzlich noch ein JSON – Format für die textbasierte Übertragung der Nachrichten hinzugefügt werden soll. Ist die aktuelle Architektur für diese Erweiterung geeignet? Welche Anpassungen sind bei den Nachrichtenklassen nötig? Wie sieht es mit den Unit-Tests und den Testdaten aus? Was kann davon übernommen werden?

Beurteilungskriterien

- Qualität der Implementierung der Fachklasse (Klassenhierarchie, Interfaces, Fehlerbehandlung, Datenhaltung, Schnittstellendesign)
- Qualität der Testimplementierung (Vollständigkeit der Testfälle, Code-Qualität)
- Reflexion TTD
- Softwareentwurf für JSON Erweiterung

Aufgabe 2: End-To-End Testing

Suchen Sie sich eine Aufgabe aus:

1. Migrieren Sie die Test Suite des Enum-Tutors von Selenium nach Playwright.
2. Erstellen Sie eine minimale Browseranwendung mit Cookies. Testen Sie die Webseite, insbesondere die Cookies, automatisch mit Playwright.

Contrôle continu 2 (50% der Endnote):

Aufgabe 3: Internationalisierung

Internationalisieren Sie die Software, so dass diese zur Laufzeit auf Englisch, auf Französisch oder auf Deutsch umgeschaltet werden kann. Startsprache ist die des Betriebssystems bzw. die letzte eingestellte Sprache.

Suchen Sie sich eines der Programme aus:

1. Konsolenprogramm „Projektleiter“
2. Swing-Programm Studierendenauszeichnung
3. Thermometer-Programm

Achten Sie auf eine gute Softwarearchitektur. Erstellen Sie mit Maven eine runnable-jar Datei.

Aufgabe 4: REST-API

Implementieren Sie einen einfachen REST-Server mit Java ohne Zuhilfenahme eines Frameworks. Die 4 CRUD Operationen sollen möglich sein. Die Daten können und sollten In-Memory gespeichert werden. Auf einen Datenbankzugriff sollte verzichtet werden.

Erstellen Sie von Ihrem REST-API eine strukturierte Dokumentation.

Es findet eine Programmabnahme statt, bei der Sie mit einigen Testszenarien zeigen, wie ihr REST Sever funktioniert.