IoT Example - InternetButton

1 SWE Abschluss Übung

1.1 Ziele dieser Übung

In dieser Übung werden Sie gemeinsam in einer Gruppe die Funktionen des InternetButtons ansprechen unter Verwendung der agilen Methode *Scrum* umsetzen. Die Gruppen sind wie bei der letzten Übung. Das Ziel dieser Übung ist, dass Sie zum einen die in der Vorlesung SWE durchgenommenen Ansätze zur Organisation von Softwareprojekten anwenden und darüber hinaus den Stoff aus den vorgelagerten Vorlesungen des Fachbereichs Software Engineering (PR1, PR2) festigen. Darüber hinaus soll die Übung Ihnen auch die Gelegenheit geben, einen Tag lang ein Softwareprojekt „zu erleben“ und daneben auch noch **„Spaß machen“.**

Die Buttons können sie unter https://meet.jit.si/C02SWE2021 beobachten.

1.2 Bewertung

Es wird in **erster Linie das methodische Vorgehen bewertet**; in zweiter Linie erst die Umsetzung der Stories. Achten Sie bei der Umsetzung auf die folgenden Punkte:

- Programmieren Sie in Paaren
- Verwenden Sie **Test First** und **Continuous Integration** (der Klassenname muss mit *Test enden z.B: `Behaviour`**`Test`**)
- Halten Sie sich an den nachfolgenden Übungs-Plan
- Checken Sie alle hier geforderten Artefakte unmittelbar nach deren Erstellen ins GIT ein.
- Zu keinem Zeitpunkt nach der Iteration Zero darf die Quellcodeverwaltung Testfälle, die nicht funktionieren, beinhalten; Der eingecheckte Quellcode muss lauffähig sein
- Am Ende jeder Iteration müssen sich sämtliche Artefakte in der Quellcodeverwaltung befinden.

Benutzen Sie dazu unterhalb Ihres Projektverzeichnisses das Unterverzeichnis ***docs***. Legen Sie für jede Iteration Ihre Dokumente in den Ordnern ***itXXX*** (z.b. it001 für die erste Iteration) ab. Allgemeine Dokumente (Burnddown usw) legen sie direkt in ***docs*** ab.

1.3 Ablauf

Die Aufgabe ist in Epics unterteilt, die jeweils in einzelnen Stories unterteilt sind. Da das Projekt aufbauend ist, müssen die Epics in der angegebenen Reihenfolge umgesetzt werden. Die Stories selber sind nicht gereiht.
Sie starten einmal mit einer Iteration Zero um das Environment usw. einzurichten und dann in der 1. Iteration startet die Implementierung

1.3.1 Iteration Zero.

Bestimmen Sie einen Scrum-Master. Dieser hat die Aufgabe dafür zu sorgen, dass der Scrum-Prozess im Allgemeinen sowie der hier beschriebene Prozess im Speziellen eingehalten werden. Bestimmen sie auch einen PO der für dem Projektfortschritt zuständig ist und die Gruppe uns gegenüber vertritt. Teilen sie die restliche Gruppe in **2 Teile** die dann immer die gemeinsam eine Unteraufgabe erledigen. Die einen schreiben Tests die anderen die Implantierung. Arbeiten sie wirklich in Gruppen!

Die das starterKit ist im Moddel als zip zu finden. Spielen sie daises in ein öffentliches repository ein und geben sie die Url dazu im Moodle bekannt.

Erstellen Sie einen Release-Plan, welcher darstellt, welche Stories ungefähr in welcher der heute durchgeführten Sprints umgesetzt werden. Checken Sie diesen ein.

Setzen sie in den `device.properties` die Id des Buttons der ihrer Gruppennummer zugeordnet ist und starten sie die Beispiele aus dem Package

`org.c02.swe.iot.cloud.example`

und beobachten sie den live Stream.

Die einzelnen Tokens sind ihrer Gruppe schon zugeordnet.

1.3.2 Iteration 1

Sprint-Planning (max. 5 Minuten)
Führen Sie ein Sprint-Planning-Meeting durch, in dem Sie entscheiden, welche der Stories aus dem Backlog in dieser Iteration umgesetzt werden sollen.
Teilen Sie die Stories in Aufgaben auf und schätzen Sie deren Aufwand gemeinsam.
Legen Sie fest, wer sich um welche Aufgabe kümmert, indem sich die Gruppenmitglieder zu den einzelnen Aufgaben committen. Beachten Sie dabei, dass sie in dieser Übung Pair-Programming verwenden.
Erstellen Sie basierend darauf einen Sprint-Plan und checken Sie ihn ein.

Sprint (45 Minuten)
Arbeiten Sie die ausgewählten Aufgaben in Paaren ab. Verwenden Sie Test-First.

Daily Scrum-Meeting (insgesamt max. 5 Minuten)
Führen Sie im Sprint 2x pro Iteration ein DailyScrum und dokumentieren Sie die drei beantworteten Fragen in einem Dokument, welches Sie einchecken. Anmerkung: In einem richtigen Scrum-Projekt würde man dies nicht dokumentieren. In dieser Übung dient es lediglich dem Nachweis, dass entsprechend der Vorgaben gearbeitet wurde.

Sprint-Review
Anders als in Real-World-Scrum-Projekten wird in dieser Übung kein richtiges Sprint-Review durchgeführt. Aus Zeitgründen werden stattdessen die Lektoren das Ergebnis des Sprints auschecken und begutachten.

Sprint-Retrospektive (max. 5 Minuten)

Führen Sie eine Retrospektive durch und dokumentieren Sie Verbesserungsmaßnahmen in einem Dokument, welches Sie einchecken.

1.3.3 Iteration 2 - n

Die folgenden Iterationen laufen ab, wie Iteration 1. Beachten Sie dabei folgende Punkte:

- Aktualisieren Sie am Beginn jedes Prints ein Burn-Down-Chart, welches sich auf die Iterationen bezieht, und checken Sie es ein.
- Wechseln Sie nach jeder Iteration den Programmier-Partner innerhalb der Gruppe.
- Wenn Sie in einzelnen Retrospektiven keine Verbesserungsmaßnahmen definieren, ist dies auch ok, sofern Sie diesen Umstand dokumentieren.
- Ermitteln Sie am Ende jeder Iteration die Velocity und verwenden Sie diese Kennzahl beim Planen der nächsten Iteration.

1.4 Aufgabenstellung

Sie haben die Aufgabe due LEDs des InternetButtons anzusteuern
Das *StarterKit* enthält einen Hilfsklasse (`ParticleApi`), welche die Basis-Kommunikation mit dem Button erleichtert. Diese implementiert ein Interface (`IParticleApi`). In `org.c02.swe.iot.cloud.example` Package können Sie die Funktionen anhand von zwei Beispielen ausprobieren. Um diese sehr einfache Schnittstelle herum soll eine HighLevel Api (`IButton`) entstehen, über die man dann ohne die genaue Kommunikation mit dem Button verwenden zu müssen Abläufe Umsetzen kann. Für dieses Interface gibt es noch keine fertige Implementierung sondern nur eine leer Hülle (`Button`).

1.4.1 Epic 1: Teilweise Implementierung von `Button`

Ziel dieses Epics ist, dass eine HighLevel API für den Internet Button zu Verfügung steht. Implementieren sie Methoden `setLed` und `allLedsOff` muss ein Test implementiert werden, der überprüft ob beim Aufruf der Methode die richtigen Werte an den ButtonWrapper übergeben werden ohne(!), dass dabei der Button benutzt wird (Verifikation ohne dem Gerät).[1] Schauen Sie sich dazu den `ButtonTest` an. Dieser muss nach dem Epic erfolgreich sein und auch erweitert worden sein.

Dieses Epic ist ein folgende Stories unterteilt:

1. Implementieren Sie die Methoden zum Setzen der Leds
2. Implementieren Sie Abschalten der Leds.
3. Schreiben Sie Für alle einen Test der ohne den Button aus kommt.
4. Erweitern Sie auch die kleine Applikation die zeigt, wie die LEDS gesetzt und gelöscht werden (`LedDemoApp`).

[1] Sie können dazu das Interface noch einmal implementieren oder http://mockito.org/ benutzen.

1.4.2 Epic 2: Einfache Effekte

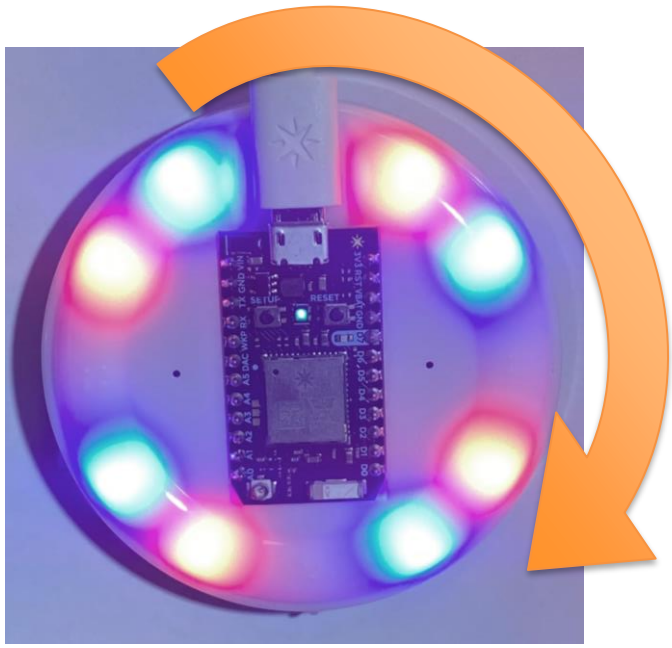
Ziel dieses Epics ist, dass basierend auf der im 1. Epic umgesetzen APIs unterschiedliche Effekte zu realisieren. Dabei werden diese in jeweils in einer eigenen Klasse umgesetzt. Sehen sie sich zuerst den Effekt `WhiteledRunning` an. Es gibt dazu einen Implementierung, eine Demo Anwendung (die jetzt auch funktionieren müsste) und einen Test.

1. Schreiben sie einen Effekt `SpinningLed` bei dem man die Farbe der Led festgelegt werden kann. Es wird immer nur eine LED eingeschaltet. Man kann die Anzahl angeben wie oft die LED im Kreis laufen soll.
2. Schreiben Sie parallel einen Test zu dem Effekt
3. Zeigen sie die Funktion in einer kleinen Anwendung.

1.4.3 Epic 3: Vervollständigen der Button Implementierung

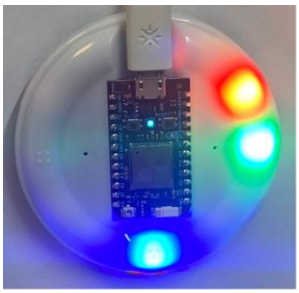
Implementieren sie nun auch die ausständigen Methoden in der Button Klasse und zeigen sie diese anhand eines Effektes.

1. Vervollständigen der Button Implementierung
2. Erweitern des ButtonTests um die neuen Methoden.
3. Erstellen sie einen Effekt `SpinningWheel` der Rot Blau Leds im Uhrzeigersinn rotieren lässt
4. Schreiben sie auch einen Testfall für diesen Effekt
5. Erstellen sie auch eine Demoanwendung für diesen Effekt.

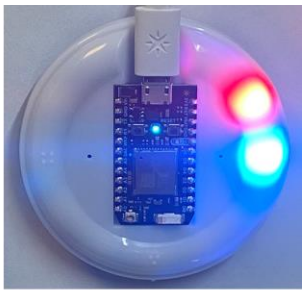


1.4.4 Epic 4: Uhrzeit anzeigen

Ziel dieses Epics eine Uhr anzuzeigen. Dabei können Minuten und Sekunden nur jeweils in 5er Schritten angezeigt werden. Rot steht für den Stundenzeiger, Grün für den Minutenzeiger und Blau für den Sekundenzeiger. Wenn die Zahlen überlappen werden einfach alle die entsprechenden Farben gemeinsam angeschaltet.



14:15:30



14:15:15

03:15:45

03:15:15

1. Vervollständigen Sie die Klasse `ClockUtil`
2. Schreiben Sie einen Test der die Implementierung überprüft
3. Und schreiben Sie eine kleine Demoanwendung die alle Sekunden die Zeit aktualisiert.