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 Gruppeneinteilung finden Sie im Moodle. 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“.**

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**
- Halten Sie sich an den nachfolgenden Übungs-Plan
- Checken Sie alle hier geforderten Artefakte unmittelbar nach deren Erstellen ins SVN 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.

Erstellen Sie dazu unterhalb Ihres Projektverzeichnisses ein Unterverzeichnis ***docs***. Legen Sie für jede Iteration einen Ordner ***itXXX*** (z.b. it001 für die erste Iteration) an und legen sie dort die Dokumente 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 zusätzlich die Aufgabe dafür zu sorgen, dass der Scrum-Prozess im Allgemeinen sowie der hier beschriebene Prozess im Speziellen eingehalten werden.
Klonen Sie das StarterKit von Github:
`https://github.com/gerhardfliess/InternetButton.git`
und legen Sie für Ihre Gruppe ein Repository an.
Machen Sie sich mit dem Starter-Kit vertraut. Geben Sie die URL für Ihr Reposity dem Lektor bekannt damit man den Build am jenkins einrichten kann.

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

Wir werden innerhalb dieser Phase Ihnen auch Ihren Internet Button zuteilen und mit einem Hot-Spot verbinden, damit Sie dann ihr eigenes Gerät haben und die Demos und die Implementierung laufen lassen können.

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 die verschiedenen Funktionen des InternetButtons abzufragen und abhängig davon Ausgaben am InternetButton auszulösen.
Das *StarterKit* enthält einen Hilfsklasse (`ParticleApiWrapperImpl`), welche die Basis- Kommunikation mit dem Button erleichtert. Diese implementiert ein Interface (`ParticleApiWrapper`). In Demo Package können Sie die Funktionen anhand von zwei Beispielen ausprobieren. Um diese sehr einfache Schnittstelle herum soll eine HighLevel Api (`InternetButtonApi`) 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 (`InternetButtonApiImpl`).

1.4.1 Epic 1: Vervollständigen Sie Klasse `InternetButtonApiImpl`

Ziel dieses Epics ist, dass eine HighLevel API für den Internet Button zu Verfügung steht. Für alle Methoden 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 `BehaviourTest` aus dem nächsten Epic an. Der Testfall muss auch ein Ergebnis am Buildsystem liefern.

Dieses Epic ist ein folgende Stories unterteilt:

1. Implementieren Sie die Funktion zum Abspielen eines Tones in der `InternetButtonApiImpl`. Schreiben Sie auch eine kleine Applikation die zeigt, dass die Melodie auch wirklich ertönt (`PlayDemoApp`).[2]
2. Implementieren Sie die Methoden zum Setzen der Leds und zum Abschalten der Leds. Schreiben Sie auch dafür einen Test der ohne den Button aus kommt. Schreiben Sie auch eine kleine Applikation die zeigt, wie die LEDS gesetzt und gelöscht werden (`LedDemoApp`).
3. Implementieren Sie die Methoden zum Abfragen und Zurücksetzen der Button Zähler. Schreiben Sie auch eine kleine Applikation die Zähler ausliest und zurücksetzt (`ButtonCounterDemo`).

[1] Sie können dazu das Interface noch einmal implementieren oder http://mockito.org/ benutzen.
[2] Beim Testen achten Sie bitte auch Ihre Kolleginnen und Kollegen, damit die nicht zu oft die Melodie zu hören bekommen ;-)

1.4.2 Epic 2: Interaktive Rückmeldungen

Ziel dieses Epics ist, dass basierend auf der im 1. Epic umgesetzen APIs unterschiedliche Verhalten des Buttons realisiert werden. Dabei sollten diese Verhalten jeweils in einer eigenen Klasse umgesetzt werden. Es muss für jedes Verhalten auch ein eigenes Demo geben. Das Demo soll die Klasse, die das Verhalten umgesetzt hat zyklisch aufrufen (die implementiert das `Runnable` Interface). Es ist ausreichend wenn die Applikation händisch gestoppt werden muss.

Die erste Story ist verpflichtend, 2 und 3 sind Vorschläge. Wenn Sie ein anderes Verhalten umsetzen möchten können sie dies gerne machen. Definieren Sie es als neue Story checken Sie diese ein. Es müssen zumindest **2** weitere Verhalten in einer eigenen Klasse umgesetzt werden.

1. Die Klasse `CountAndShowLed` zählt die Klicks von einem Button und nimmt die Anzahl als Position der Led die leuchten soll. Erweitern Sie das Verhalten so, das es auch mit mehr als 12 Klicks umgehen kann und dann wieder von vorne beginnt. Erweitern Sie zuerst den Test und vervollständigen Sie dann die Implementierung. Schreiben sie auch eine Applikation die das Verhalten zeigt.
2. Erstellen Sie eine neue Klasse und ändern Sie das Verhalten so ab, dass bei jedem Klick die Farbe immer einen stärkeren rot Anteil bekommt (10er Schritte beim Farbwert). Beim Erreichen, des vollen Rot Anteils (255) soll der Zähler zurück gesetzt werden. Zeigen Sie das wieder anhand eines Tests.
3. Es soll ein neues Verhalten implementiert werden, dass bei allen 10 Klicks eine Melodie abspielt.

1.4.3 Epic 3: Einbeziehen des Beschleunigunssensors

Neben den Werten für Zähler der Buttons kann auch der 3 Achsen Sensor des Internet Buttons angesprochen werden. Dazu müssen die Variablen `xValue`, `yValue` und `zValue` so wie die Zähler abgefragt werden.
Ziel des Epics ist, die Werte wieder in einer High Level API zu Verfügung zu stellen. Die erste Story ist wieder verpflichtend die weiteren ein Vorschlag, die Sensoren müssen aber in zumindest **2** neuen Verhalten benutzt werden.

1. Erweitern Sie die Interfaces so, dass die es einzelne Methoden zur abfrage des x,y, und z wertes des Beschleuniguns Sensors zu Verfügung stehen. Schreiben sie dazu wieder zuerst Testfälle und zeigen Sie die Implementierung in einem kleinen Demo.
2. Erstellen die ein Verhalten ähnlich dem in der Klasse `CountAndShowLed` nur, das der Rot Wert von dem xWert des Beschleunigungs Sensors abhängig ist. Auch hier wieder Testfall und ein kleines Demo.
3. Erstellen die ein Verhalten ähnlich dem in der Klasse `CountAndShowLed` nur, das der x-Wert den rot-Wert, der y-Wert den grün-Wert und der z-Wert den blau-Wert definiert. Auch hier wieder Testfall und ein kleines Demo.

1.4.4 Epic 4: Erweitern der Firmware

Ziel dieses Epics die Firmware des Buttons zu erweitern. Wenn Sie abschätzen können, dass Sie zu diesem Epic kommen, kontaktieren die sie Lektoren, sie bekommen dann Zugriff auf die für die Internet Buttons.

1. Ermöglichen Sie es, dass mehrerer Leds auf einmal gesetzt werden können. Dazu muss eine neue Methode in der Firmware umgesetzt werden und es sollen die entsprechenden HighLevel API Methode angeboten werden. Wieder Testfälle und Demo.
2. Ermöglichen Sie es, dass man eine andere Melodie Spielen kann (Beispiele haben die Lektoren). Dazu muss in der Firmware ein Parameter für die Melodie angegeben werden. Wieder Testfälle und Demo.