Praktikum IoT Display

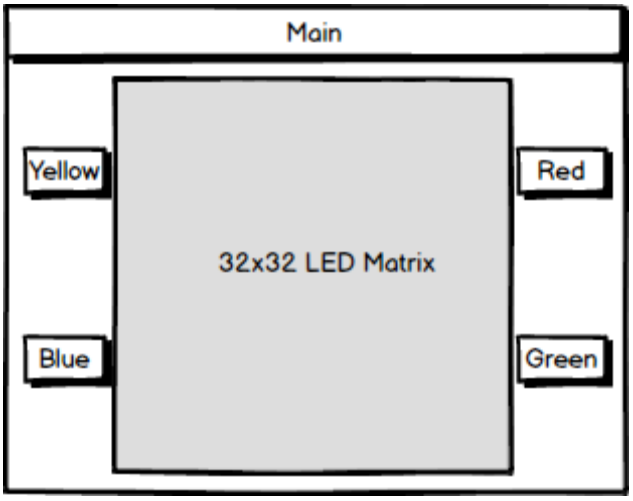
1 Ausgangslage und Rahmenbedingungen

1.1 Einleitung

Ihre Firma hat sich entschieden, im Bereich „Internet of Things“ (IoT) tiefer gehende Abklärungen durchzuführen. Dazu sollen Sie die Software für ein LED-Display entwickeln, das Taster hat und über einen Internet-Zugang verfügt. Ein Taster ist ein Bedienelement, das durch Drücken betätigt wird und nach dem Loslassen selbständig in die Ausgangslage zurückkehrt.

1.2 Hardware

Die Hardware für diese erste Technologie Abklärung besteht aus einem Raspberry Pi, auf dem Java 8 installiert ist, einem Display aus 32 x 32 RGB LEDs (wahlfrei ansteuerbar) und 5 Tastern. Davon ist 1 Taster oben montiert und je 2 Taster sind auf beiden Seiten montiert. Die Taster haben folgende Namen: Oben „Main", links oben "Yellow", links unten "Blue", rechts oben „Red", rechts unten "Green" (siehe Wireframe-Sketch unten). Es besteht ein Internet-Zugang, der aber eventuell nicht über die Standard Java Klassen verfügbar ist.



Es ist schon jetzt klar, dass die Hardware grösseren Veränderungen unterworfen sein wird. Ein paar Möglichkeiten für weitere Versuche sind:

- Grössere Auflösung der LED Matrix, eine 64x64 Variante ist bereits fest eingeplant
- LCD Bildschirm mit (viel) grösserer Auflösung und Beleuchtung
- Touch-Screen mit „Soft"- Tasten
- Weitere, zusätzliche Sensoren, z.B. für Helligkeit oder Temperatur
- Internet-Zugang über eine Relay Stationen oder USB, also nicht direkt über die eingebaute Hardware
- Mikrofon mit Sprachanalyse

1.3 Firmware

Von der Hardware-Entwicklungsabteilung bekommen Sie später einen funktionierenden Prototyp mit einer Firmware, die den Zugriff auf die Hardware und Systemcode ermöglicht. Das Interface der Firmware ist in der Package „**`ch.zhaw.swen1.iotdisplay.platform`**“ enthalten. Das Design ist

absichtlich nur schwach objekt-orientiert, da dies bei Firmware-Programmen meistens der Fall ist. Im selben Package befindet sich auch das „**`Executable`**“-Interface. Dieses Interface müssen Anwendungsprogramme implementieren. Seine einzige Methode „start(...)“ wird nur einmal beim Starten der Anwendung aufgerufen. Dabei wird eine Instanz des „**`Platform`**“-Interface übergeben. Dort kann sich die Anwendung als Listener für Events vom Timer und von den Tastern registrieren. Die „**`Platform`**“ stellt zusätzlich noch eine SocketFactory zur Verfügung, über die Client-Sockets erzeugt werden können.

1.4 Hilfsklassen

Im Package „**`ch.zhaw.swen1.iotdisplay.util`**“ befinden sich Klassen (inklusive Quelltext), die Ihnen die Arbeit erleichtern können: Eine Klasse, um einen Server des Network Time Protocols (NTP) anzusprechen, einen 5x7 Pixel Font und ein paar Hilfsmethoden, um auf dem Display Zeichen mithilfe des Fonts zu zeichnen. Diese Klassen sind z.T. schlecht dokumentiert und wurden offensichtlich aus Code-Fragmenten aus dem Internet zusammengesetzt. Für diese Versuche reichen sie aus, sollten dann später aber refaktorisiert werden.

1.5 Simulationsumgebung

Bis die Hardware bereit ist, steht eine Simulations-Umgebung zur Verfügung. Sie implementiert das „**`Platform`**“-Interface basierend auf Swing und ermöglicht die Ausführung von einem Executable. Zusätzlich finden Sie noch eine einfache Anwendung, die ein Rechteck auf das Display zeichnet. Dort finden Sie auch die Methode „**`main`**“, um die Simulations-Umgebung zu starten.
Die Simulations-Umgebung ermöglicht es Ihnen, Ihr Executable entweder mit einem 32x32 oder mit einem 64x64 Pixel Bildschirm zu starten.

2 Anforderungen

Im Rahmen dieses Praktikums entwickeln Sie eine Anwendung für das IoT Display, die die folgenden 2 Funktionen ausführt:

1. Die aktuelle Zeit (Stunden und Minuten) und das aktuelle Datum (Tag und Monat) wird angezeigt. Optional können noch die Sekunden irgendwie visualisiert werden.
2. Videospiel „Pong“.

Jede Funktion füllt das gesamte Display und soll damit auch die Taster exklusiv benutzen. Zwischen den beiden Funktionen wird mit dem Taster „Main“ umgeschaltet.

Die aktuelle Zeit soll über einen Zeitserver (NTP) beim Starten der Anwendung geholt und dann periodisch jede volle Stunde erneut synchronisiert werden. Wenn eine Synchronisation nicht gelingt, soll in 15 s ein neuer Versuch durchgeführt werden. Solange die Synchronisation nicht funktioniert, soll die Anzeige der Zeit und des Datums blinken. Vor der ersten Synchronisation soll der 01.01(.2000) 00:00 angezeigt werden.

Das Spiel „Pong“ (Regeln siehe Wikipedia) soll so implementiert werden, dass mit jedem Wechsel in diese Funktion das Spiel neu startet. Die Taster an den beiden Seiten dienen dazu, die Schläger zu bewegen. Wenn der Ball oben oder unten das Spielfeld verlässt, bekommt der Spieler einen Punkt, der den Ball nicht mehr spielen konnte. Wenn aber ein Spieler den Ball seitlich durchlässt, bekommt der andere Spieler einen Punkt. Nach jedem Punkt pausiert das Spiel für 3s, bis dann der nächste Ball von der Seite desjenigen Spielers kommt, der den Punkt erhalten hat. Der Schläger soll den Ball je nach der Position bezüglich dem Schlägermittelpunkt seitlich etwas ablenken.

3 Aufgaben

3.1 Einleitung

Da die Anwendungsfälle eher einfach sind, sollten Sie diese Artefakte für sich als Vorbereitung erarbeiten. Wir werden diese dann in der letzten Vorlesung gemeinsam konsolidieren und Musterlösungen zu allen Aufgaben besprechen.

Es ist notwendig, dass Sie die Interfaces und Klassen des „**`Platform`**“-Packages studieren und verstehen.

Vergessen Sie nicht, Ihre Design-Entscheide mittels Entwurfsmuster zu begründen -;)
Beachten Sie, dass das Praktikum eine Einzelarbeit ist und dass bei diesem Praktikum die technischen Vorgaben erfüllt werden müssen, um die volle Punktzahl zu erreichen.

3.2 Domänenmodell Pong (0.5 P)

Entwerfen Sie das Domänenmodell des Spiels Pong.

3.3 Zustandsdiagramm der gesamten Benutzerführung (1 P)

Zeichnen Sie ein UML-Zustandsdiagramm, das die gesamte Benutzerführung zeigt, wie sie oben beschrieben ist, inklusive der Berücksichtigung relevanter Domänenlogik Ereignisse wie z.B. eine neue Sekunde oder dass im Spiel Pong ein Punkt erzielt worden ist.

Dieses Diagramm ist etwas komplexer, da hier konkret modelliert wird, was bei jedem Tastendruck und bei speziellen Zuständen der Domänenlogik geschieht.

3.4 Entwurf übergeordnetes Design (0.5 P)

Da klar ist, dass verschieden grosse Displays zur Verfügung stehen und weitere Funktionalität (z.B. Wecker) hinzukommt, entscheiden Sie sich dafür, konsequent eine geschichtete Softwarearchitektur (Layering) anzuwenden und somit UI und Domänenlogik strikt zu trennen.

Im Sinne eines Framework-Designs sollen Sie Interfaces und/oder abstrakte Basisklassen sowohl für das UI wie auch für die Domänenschicht entwerfen. Überlegen Sie sich, welche Verantwortlichkeiten im Detail diese Basisklassen übernehmen und wie genau die Methodenköpfe mit Parameterlisten aussehen.

Dokumentieren Sie Ihren Entwurf für diese Basisklassen mit einem Design-Klassendiagramm. Fügen Sie dort auch die relevanten Klassen aus dem **„`Platform`"**-Package ein sowie die Klasse, die „`Executable`" implementiert.

3.5 Entwurf Pong (1 Punkte)

Entwerfen Sie die Domänenlogik für das Spiel Pong, basierend auf Ihrer Lösung für Aufgabe 3.2 und 3.3. Visualisieren Sie Ihren Entwurf mit einem UML-Klassendiagramm und einem Interaktionsdiagramm, das die Behandlung des Timer-Events zeigt für den Fall, dass der Ball auf einen Schläger trifft.

3.6 Abgabe

Zeitpunkt: Bis 20:00 vor der letzten Vorlesung auf OLAT als ein PDF hochladen.

4 Bonusaufgabe: Programmierung (2 Punkte)

4.1 Aufgabe

Programmieren Sie die Anwendung. Dazu gehören alle oben erwähnten Anforderungen. Es steht Ihnen natürlich frei, noch weitere Funktionalität wie Spiele oder Animationen zu realisieren.

Hinweise

- Beachten Sie noch, dass Sie alle Ressourcen wie Graphiken oder Icons über den Class-Loader von Java laden.
- Als Zeitserver muss zwingend die Url im Beispiel „**`SntpFetchExample`**" verwendet werden.
- Alle Sockets müssen über die Klasse „**`SocketFactory`**" erstellt werden, die Sie von der „**`Platform`**" beziehen können.
- Ihr Code muss mit Ihrem Design übereinstimmen, sonst kann es Abzüge geben.

Achtung:
Verwenden Sie das zur Verfügung gestellte Beispiel-Programm bzw. Klasse **„PraktikumIoTDisplay".** Belassen Sie den Datei/Klassennamen, aber ändern Sie die Package-Bezeichnung, indem Sie „example" mit Ihrem **eigenen Kürze**l ersetzen. Die Klasse **„PraktikumIoTDisplay"** enthält das Beispielprogramm, das Sie natürlich mit Ihrem eigenen Code füllen sollen, und enthält die Main-Methode, mit der die Simulationsumgebung gestartet wird.

4.2 Zusatzmaterial

Das gesamte Zusatzmaterial befindet sich in der ZIP-Datei „Praktikum_IoTDisplay-FS19.zip". Darin befinden sich die folgenden Dateien:

Platform.jar	Beinhaltet alle Plattform-Interfaces und die gesamte Simulationsumgebung. Fügen Sie diese Datei als Dependency Ihrem Projekt hinzu.
PlatformDoc.zip	Die Javadoc der Plattform-Interfaces.
Example.zip	Die Datei „PraktikumIoTDisplay.java"
Util.zip	Quelltexte von Utility Klassen für BitMapFont Handling und das SNTP-Protokoll. Die Dateien, die Sie verändern möchten, sollten Sie in Ihre Packages kopieren. Das Beispielprogramm greift auf diese Klassen zu, weshalb es empfehlenswert ist, diesen Code ebenfalls zu entpacken.

4.3 Abnahmetest

Alle eingereichten Lösungen werden einem Abnahmetest unterworfen und ein paar wenige davon werden während der letzten Vorlesung demonstriert. Die Punktevergabe dieser Teilaufgabe richtet sich dann nach der erreichten Funktionalität und Stabilität, aber auch an der Qualität des Quelltextes.

4.4 Abgabe

Zeitpunkt: Bis 20:00 vor der letzten Vorlesung als ZIP-Datei auf OLAT hochladen. .
Form: Nur der Quelltext, der von Ihnen geschrieben wurde, in der korrekten Package-Struktur.