Utvecklarguide för aDrumDrum

Introduktion

aDrumDrum är en trummaskin till android som fungerar som en step sequencer.

Hur man får tag i koden

Vi finns på GitHub, det är coolt, på adressen: https://github.com/KVHC/adrumdrum

Detta är ingen guide för hur man använder git, så kolla upp det först om du inte vet hur det fungerar.
clone https://github.com/KVHC/adrumdrum.git

Hur man bygger

Utveckla i Eclipse, det blir lättare då.

1. I Eclipse välj File -> Import.
2. Välj "existing android code into workspace" och klicka på next.
3. Som root directory välj ".../adrumdrum/".
4. Eftersom det är Eclipse och Android så bygger den automatiskt.
5. Skapa en AVD(android virtual device) för att testa appen om du inte vill testa med en fysisk enhet

För att bygga med ant, ställ dig i adrumdrum-mappen och kör kommandot "**ant debug**". Du får då ut ut en apk som heter "aDrumDrum-debug.apk" som ligger i bin/. Denna är det bara att skicka till valfri androidenhet, som ska acceptera "Applications from unknown sources".

För att göra en signad release ställer man sig i rotkatalogen och kör kommandot "**ant release**" och skriver sen in det hemliga lösenordet.

Hur man får igång tester

I Eclipse.

1. I Eclipse välj File->Import.
2. Välj "existing android code into workspace".
3. Som root directory välj ".../adrumdrum/test/ADrumDrumTest/" och klicka på next.
4. Högerklicka sedan på ditt nyimporterade projekt och välj Properties.
5. Under Java Build Path, välj fliken Projects.
6. Klicka på Add och lägg till projektet där du har aDrumDrum (alltså själva appens projekt).

Med ant.

Först måste du ha Apache ant installerat. Följ deras instruktioner för att installera och konfigurera ant.

När ant är installerat behöver du lägga till emma.jar och emma_ant.jar i Build Path för testprojektet och för huvudprojektet, detta gör du enklast genom att högerklicka på testprojektet i Eclipse och välja Build Path > Configure Build Path, under fliken libraries välj Add External JARs och lägg till emma.jar och emma_ant.jar som du hittar i mappen adrumdrum/emma_jars/. Upprepa sedan processen för huvudprojektet så att emma är importerat i bägge projekten. För att bygga appen och testprojektet och sedan köra testerna samt generera en coveragerapport med EMMA gör du så här. Ställl dig i mappen ADrumDrumTest och kör kommandot **"ant emma debug install test"**.

(På windows måste du ha ANDROID_HOME environment variable satt till din android sdk path (på windows 7 kan det vara C:\Users\dinanvändare\AppData\Local\Android\android-sdk\).)

Release procedure.

Innan du gör en release bör du se till att följande saker är gjorda:

1. All kod som tillkommit sen föregående release skall ha javadoc-kommentarer.

2. All kod som tillkommit sen föregående release skall vara testad (i möjligaste mån) med unit tester och acceptanstester.

3. Har det tillkommit licensierat material så skall de nya licenserna följas och finnas tillgängliga för användaren att läsa.

4. All kod som tillkommit sen föregående release skall vara genomläst och kollad av åtminstone en utvecklare utöver den som skrev koden.

5. Se till att det inte finns några artefakter i produkten, inga knappar eller liknande som inte fungerar.

6. Se till att både huvudprojeket och testprojektet bygger och kör korrekt.

X. Att du har ett leende på läpparna och inte skäms över vad du nyss gjorde.

Strukturen

För mer information om paketen och de ingående klasserna se arkitekturbeskrivningen. Nedan följer korta beskrivningar av paketen.

kvhc.adrumdrum: Huvudpaketet. Allt som ligger här är aktiviteter, som används för att initiera viktiga saker som ljud, databas och GUI.
Viktiga klasser:

- MainActivity - Huvudprogrammet

kvhc.gui: GUI-paketet. Här ligger alla våra egna knappar, containers och dylikt. Är det något som ska ritas i GUI:t så ligger det här, förutom dialogrutorna.
Viktiga klasser:

- GUIController - Styr det mesta av vad som händer i programmet

kvhc.gui.dialog: Här ligger alla våra egna dialogrutor som vi nyttjar.

kvhc.models: Här ligger de klasser som modellerar tillståndet i programmet.

Viktiga klasser:

- Song - Model för en låt. En Song innehåller flera kanaler.
- Channel - Model för en kanal. En kanal innehåller flera steg.
- Step - Model för ett steg. Ett steg representerar ett slag i en kanal.

kvhc.util: Här ligger alla "Verktygs klasser" som används av programmet
Viktiga klasser:

- Player - Spelar upp en Song
- SoundPoolrenderer

kvhc.util.db:
Här ligger implementationen för SQLite. Varje modell har en SQLiteHelper (som strukturerar upp sin modells tabell och skapar tabellen ifall den inte finns), samt en DataSource som man använder för att faktiskt använda databasen.

DataSource behandlar (eller ska behandla) sin egna modell. Implementationerna av ISongLoader och ISongRenderer

Interfaces

ISongRenderer
Ett interface som visar att en klass kan rendera en Song.
Metoder som krävs för att implementera ISongRenderer:

- renderSong - Rendera en hel Song.
- renderSongAtStep - Rendera ett visst steg i en Song.
- loadSounds - Ladda in ljud till programmet .

ISongLoader
Ett interface för att ladda in en Song till programmet. Om du vill lägga till ett nytt format som går att lägga till så är det det här interfacet som du ska implementera.
Metoder som krävs för att implementera ISongLoader:

- loadSong - Läser in en Song från en lista av objekt.
- getSongList - Läser in flera "Song:s" och retunerar en lista av dem.