Laboration 0: Begrepp och bit-manipulering i Java

1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Begrepp som används i Inbyggda system.
- Att träna på bit-manipulering i Java

1.1 Utrustning

- Javamiljö, tex Eclipse

2 Beskrivning av laboration

Laborationen består av följande moment:

1. Implementera rutiner för konvertering mellan bitar och tal
2. Implementera rutiner för att visa bitoperationer

2.1 Syfte och mål

Målet är att åskådliggöra begrepp som senare återkommer som hårdvara och i de inbyggda systemen.

2.2 Examination

2.2.1 Inlämning av rapport och programkod

Ingen inlämning av kod eller rapport! (enbart redovisning)

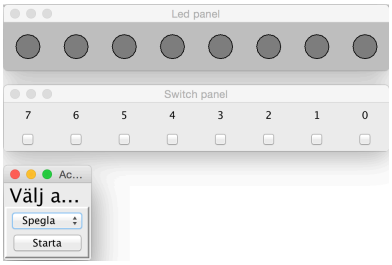
2.2.2 Praktisk och muntlig redovisning

Den praktiska delen av laborationen examineras efter att laborationens sista moment har avslutats. Följande ska redovisas:

- Demonstration av systemet, fullt fungerande enligt specifikation.

3 Systembeskrivning

Skapa ett projekt och placera paketen laboration0 och laboration0_system i src-katalogen. I Main.java hittar ni main-metoden och lite till. Exekvera main-metoden. Tre fönster blir synliga på skärmen. Men systemet saknar funktion vilken ni snart ska bidra med.



3.1 Led panel

Det översta fönstret, ”LedPanel”, visar 8 släckta lysdioder. I Controller-klassen finns metoden:

public void setLeds(int state)

vilken mappar bitmönstret i state till ledpanelen. 1 innebär att motsvarande lysdiod tänds (blir röd).

3.2 Switch panel

Fönstret i mitten, ”SwitchPanel”, visar 8 checkboxar vilka kan vara markerade eller ej. I Controller-klassen finns metoden:

public int getSwitchState()

vilken returnerar status på switcharna mappade till en int. 1:a betyder att switchen är på och 0:a att den är av.

3.3 Action panel

Det understa fönstret ger användaren möjlighet att välja någon form av ”action”, dvs att välja vilken kod som ska exekveras. Successivt under laborationen kommer det bli allt fler ”actions” att välja mellan.

4 Moment 1: Implementera rutiner för konvertering mellan bitar och tal

4.1 Beskrivning och förberedelse

I detta moment ska ni implementera 2 metoder som konverterar mellan:

- Ett tal som skall konverteras till en array av 8 boolska variabler
- En array av 8 boolska variabler som skall konverteras till ett tal av typen int

Interfacet BitConverter beskriver dessa metoder:

```
public interface BitConverter {
    public boolean[] intToArray(int state);
    public int arrayToInt(boolean[] switchState);
}
```

I filen Main.java ser du klassen BitHandler, vilken implementerar BitConverter. Men metoderna är inte fullständiga.
En instans av BitHandler är tredje argumentet då en Controller instansieras (se klassen Main). Och Controller-instansen behöver metoderna i BitHandler för att systemet ska fungera.

4.2 intToArray

Färdigställ metoden intToArray i klassen BitHandler. Metoden ska mappa innehållet i state till boolean-arrayen result. Sedan ska result returneras.
1:a i state ska medföra true och 0:a i state ska medföra false. De sista 8 bitarna i state ska användas.

Exempel

state: ...10111010 den sista 0:an är least significant bit

result:

0	1	2	3	4	5	6	7
F	T	F	T	T	T	F	T

position 0 motsvarar least significant bit

Ni kan kontrollera implementeringen av intToArray med klassen IntToArray, vilken startar systemet och skickar ett visst bitmönster till controller.setLeds(state). Du kan själv prova med lite olika bitmönster och konstatera att motsvarande lysdioder tänds.

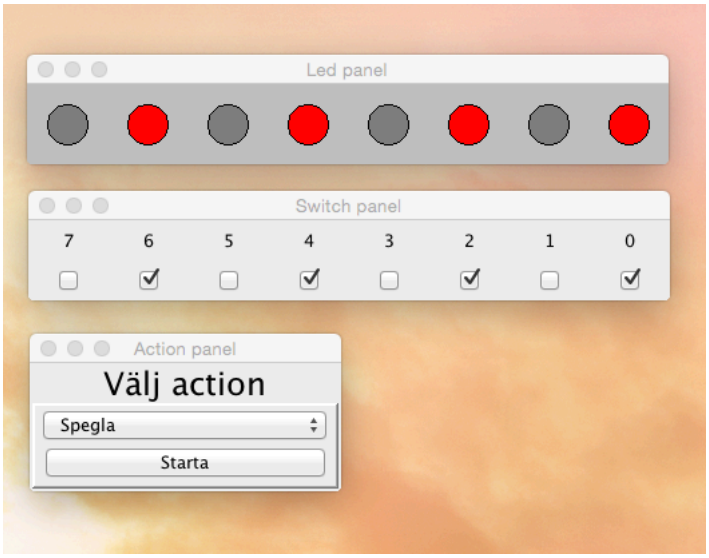
4.3 arrayToInt

Färdigställ metoden arrayToInt i klassen BitHandler. Metoden ska mappa innehållet i en boolean-array till en int.

Exempel

	0	1	2	3	4	5	6	7	
result:	F	T	F	T	T	T	F	T	position 0 motsvarar least significant bit
state:	...10111010								den sista 0:an är least significant bit

När du är färdig med metoden kan du exekvera main-metoden i klassen Main. Nu ska markerade checkboxar ge korrekta tända lysdioder då du valt ”Spegla” (enda valet) och klickar på Start-knappen.



Figur 1 Färdiga paneler

5 Moment 2: Implementera rutiner för att visa bitoperationer

5.1 Beskrivning

I detta moment ska ni använda ett antal bit-operationer och implementera kod, så att dessa åskådliggörs på ett bra sätt.

I koden skall ni skriva ett antal implementationer av interfacet Action

```
public interface Action {
    public String getActionTitle();
    public void action(Controller controller);
}
```

och lägga in dessa implementationer i en lista i main-metoden (se kommentarer i koden). All kod skrivs i filen Main.java.

5.2 Att implementera Action

Klassen ShowSwitchPanel (se filen Main.java) är ett exempel på en klass vilken implementerar Action.

- Strängen som returneras av metoden getActionTitle() visas i komboboxen i ActionPanel.
- Metoden action(Controller controller) kommer att anropas av systemet om ”Spegla” är vald och Start-knappen klickas. I klassen ShowSwitchPanel så sker följande i action-metoden:

  Anropet controller.setLeds(…); sker. Argument ska vara en int vilken beskriver vilka lysdioder som ska vara tända respektive släckta. Denna int erhålls genom anrop till metoden controller.getSwitchState(). Den enda rad som action-metoden behöver innehålla är:

  controller.setLeds(controller.getSwitchState());

Nu ska ni få ett exempel på en implementering av Action:

Skriv klassen RandomLeds i Main.java:

```
class RandomLeds implements Action {
}
```

Ni får en rödmarkering under klassens namn. Det beror på att metoderna i Action måste implementeras i klassen.

Implementera getActionTitle() så här:

```
    public String getActionTitle() {
        return "Slumpmässigt";
    }
```

Implementera action(Controller controller) så här:

```
    public void action(Controller controller) {
        Random rand = new Random();
        int state;
        for(int i=0; i<5; i++) {            // Upprepa fem gånger
            state = rand.nextInt(256);      // Slumpvärde 0-255
            System.out.println(state);      // Utskrift av slumpvärde
            controller.setLeds(state);      // Ledpanelens lysdioder sätts
            Controller.pause(1000);         // Paus en sekund
        }
    }
```

Nu ska du lägga till en instans av RandomLeds i main-metoden i klassen Main:

```
public static void main(String[] args) {
    new Controller(NBR_OF_BITS,
                   new Action[]{new ShowSwitchPanel(), new RandomLeds()},
                   new BitHandler());
}
```

Exekvera programmet på nytt. Nu ska texten ”Slumpmässigt” finnas i komboboxen. Markera den och klicka på Starta.

5.3 Att göra

Se till att följande operationer implementeras (en i varje implementation av Action):

- Implementera ”Cylon’s Eyes”, baserat på kod ni hittar i Pardue’s bok (C programming for Microcontrollers”, kap 2., exempel ”Blinky”). Byt dock ut i*2 respektive i/2 mot motsvarande bitoperationer!
- ”rinnande punkter v”. Data skiftas åt vänster och ersätts med 0, dvs data försvinner ut åt vänster i 8 steg.
- ”rinnande punkter h”. Data skiftas åt höger och ersätts med 0, dvs data försvinner ut åt höger i 8 steg.
- ”roterande punkter v”. Data ROTERAS åt vänster. Det som försvinner ut åt vänster stoppas in från höger, dvs data ”snurrar runt” i 8 steg.
- ”roterande punkter h”. Data ROTERAS åt höger. Det som försvinner ut åt höger stoppas in från vänster, dvs data ”snurrar runt” i 8 steg.
- En bit roteras åt vänster. Checkboxar som är aktiva (”på”) skall innebära att motsvarande lysdiod lyser, då biten ”passerar”. Om checkboxen är inaktiv (”off”), skall lysdioden förbli släckt, då biten ”passerar”.

Testa! Prova olika inställningar och verifiera att programmet fungerar som det är tänkt och att ni förstår vad som händer i varje operation.

När detta är klart kan ni redovisa för labhandledare!