UMEÅ UNIVERSITET
Institutionen för Datavetenskap
Rapport obligatorisk uppgift

Applikationsutveckling i Java 7.5 p

5DV135, HT17

Obligatorisk uppgift nr

1

Namn	Moa Hermansson	id14mhn@cs.umu.se mohe0025
Åtkomst	~/id14mhn/edu/5DV135/ou1/	
Inlämningsdatum	2017-11-13	
Handledare	Didrik Lindqvist: didrik@cs.umu.se, Olof Holmlund: olofh@cs.umu.se, William Viktorsson: williamv@cs.umu.se	

Användarhandledning

Programmet är implementerat i **Java 8**. JRE System Library [JavaSE-1.8]
JUnit4 är använt för tester.

Åtkomst:
~/id14mhn/edu/5DV135/ou1/

Main ligger i MyUnitTester.java
Kan köras via följande kommando i terminalen:

```
salt:~/edu/apjava/lab1> java MyUnitTester
```

Eller via en IDE, förslagsvis Eclipse. Då behöver man importera mappen MyUnitTester in i Eclipse och köra MyUnitTester.java.

Vid start av programmet kommer det att dyka upp ett fönster likt fig 1.

Fig. 1: Start av interface

Användaren fyller därefter i namnet på .java filen som hen vill testa i UnitTestern. Sedan klicka på knappen "test"(se fig 1) eller klicka på enterknappen på tangentbordet när textraden är markerad.

Systembeskrivning

Översiktlig beskrivning

Först så startas programmet i klassen MyUnitTester där main ligger. I MyUnitTester skapas en Vy i Vyklassen ViewT. I vyklassen skapas ett fönster och komponenter sätts in i fönstret därav två knappar, en textfält och en ruta för textutskrift.
Två lyssnare kopplas till gränssnittet.

Vid tryck av knapp eller textfält startas en swingWorker i klassen ControllerT. Kontrollen tar given info från användaren vilken test hen vill köra. Därefter använder kontrollen klassen ModelT för att testa givet test. I ModelT testas en klass om den implementerar rätt testklass och om den har rätt konstruktor. Sedan söker den efter metoder som innehåller ordet "test","setup" och "teardown" och sorterar dessa. Efter det så körs de valda metoderna och kontrollen hämtar resultatet från testerna och skriver ut dessa till gränssnittet.

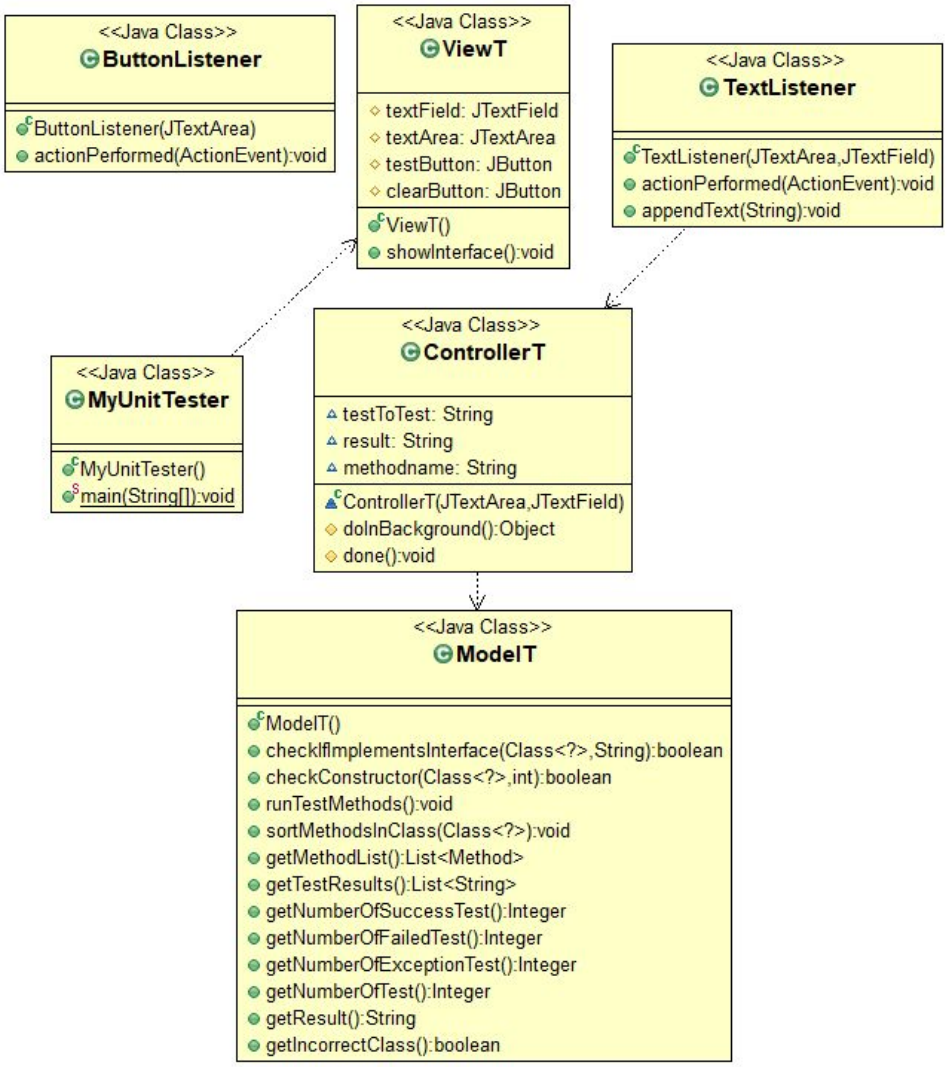

Fig 2. UML-diagram över MyUnitTester

Klassbeskrivning

MyUnitTester

I MyUnitTester klassen finns en main som startar programmet. Där skapas klassen ViewT som implementerar gränssnittet och kallar metoden showInterface för att visa gränssnittet.

ViewT

Är en vyklass som använder Javax.swing för att skapa ett gränssnitt.
Den har fyra element i sig. Högst upp till vänster ligger en "Test"knapp som är kopplad till en listener. Bredvid ligger ett textfält där användaren kan skriva.

Gränssnittet är responsivt. Alltså förändringsbart. Gränssnittet kommer att anpassa sig efter storleken på fönstret. Jämför figur 3 och 1. Den har även en skroller till höger om texten i

textarean blir för lång för fönstret. Den horisontella skrollen är borttagen då det är svårt att skrolla horisontellt för användaren.
I methoden showinterface sätts ramen ihop med gridbaglayouten och paketerar och gör gränssnittet synligt.
ActionListenersmetoderna körs när användaren trycker på någon knapp eller enter i textfältet. Vid clearknapp tömts textarean på text. Vid testknapp eller enter i textfält startar swingworkern i klassen ControllerT.

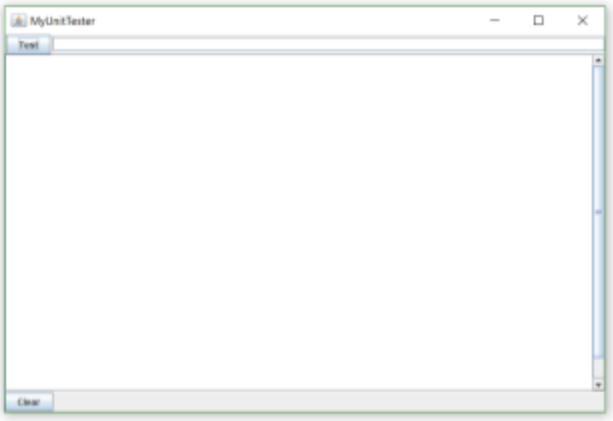

Fig 3: Gränssnittet i en annan storlek

ControllerT

Kontrollen startas i TextListener klassen.
Kontrollen extends SwingWorker och tar hand om uppgifter i bakgrunden till gränssnittet. Detta för att gränssnittet inte ska låsas medan kontrollen utför given uppgift. Kontrollen börjar i metoden doInBakground() och där har i uppgift att ta in en testklass given från användaren och konvertera strängen från textfältet till en klass som finns i samma mapp som myUnitTester. Alltså exempelvis vill användaren testa en klass som heter Test1 då skriver hen in Test1 och så letar kontrollen efter en klass “myUnitTester.Test1”. Om klassen inte finns kastas ett error och användaren får veta att det är en felaktig klass.
Sedan kör kontrollen metoden sortMethodsInClass som finns i klassen ModelT. Den ser till så att ModelT inte har angivit den som en inkorrekt klass. Om det är en felaktig klass så skrivs ett felmeddelande ut till användaren. Den hämtar alla resultat från ModelT och skriver ut det till användaren genom metoden done() som startas när doinBackground() är färdig.

ModelT

ModelT testar så att given klass har rätt interface och rätt kontruktor. Den hämtar deklarerade metoder och sorterar dem. I sortMethodsClass sorterar den metoder efter vad de har för namn. Den söker efter “setup”, ”teardown” och “test”. I runTestMethods provar den att köra metoderna. Först setup sen alla test och sist teardown. Den sparar resultaten och skickar ut till kontrollen.

Testkörningar

ModelTest är ett JUnit test som testar metoder i ModelT. Den testar om ModelT hittar felaktig parameter, om given klass ej har rätt interface, om given klass ej har interface, om struktorn har noll argument, om given klass är korrekt, om klassens structor har en parameter, om given klass är inkorrekt. Samtliga tester klarade.

En klass DummyClasses skrevs och testades i gränssnittet (Se fig 4). I DummyClasses finns det en metod som testar vad som händer om man "thread.sleep" för att se om gränssnittet låser sig medan tråden "sover". Gränssnittet låstes ej utan det gick att clear och klicka in flera test samtidigt.

Test1 som var en given testklass testades också (Se fig 4). Den fick rätt resultat och skriver ut detta korrekt i gränssnittet.

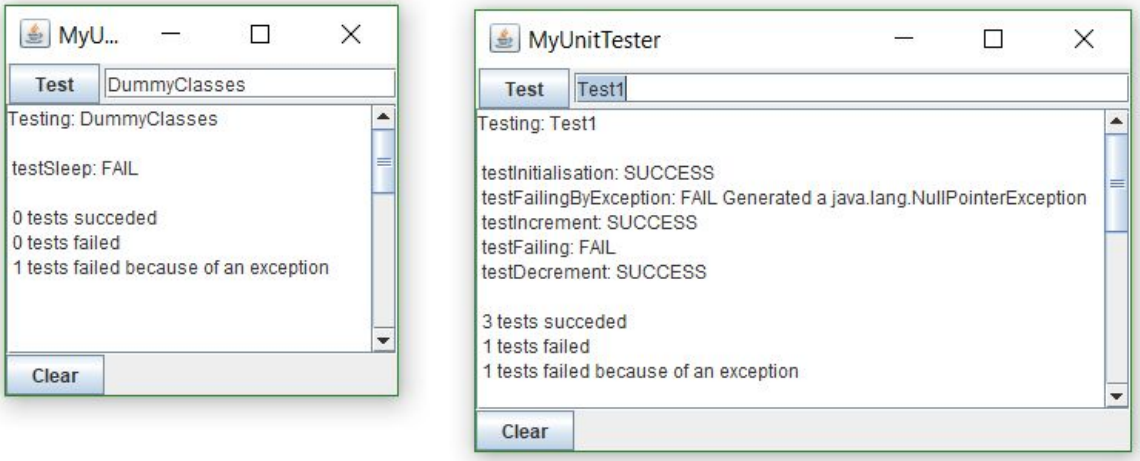

Fig 4: (Vänster) Testning av klassen DummyClasses som pausar tråd. (Höger) Testning av Test1 och dess resultat i gränssnittet.