

Laborationsrapport OU1

Objektorienterad programmeringsmetodik 5DV133

Jonas Sjödin

josj0105@cs.umu.se

2017-04-02



Kursansvarig

Anders Broberg

Handledare

Adam Dahlgren

Jakob Vesterlind

Sebastian Sandberg

Didrik Lindqvist

Daniel Harr

1. Inledning

I denna rapport redovisar jag den första obligatoriska uppgiften på kursen Objektorienterad programmeringsmetodik (5DV133) i språket java vid Umeå Universitet. Uppgiften gick ut på att tre huvudklasser skulle skapas med givna funktionsnamn och funktionsbeskrivningar vid namn `NumberDisplay`, `Clock` och `AlarmClock` där `AlarmClock` ärver av `Clock`. Genom att b.l.a. sätta tiden, få tiden att ticka på, sätta ett larm och sätta på/stänga av ett larm ska dessa klasser kunna simulera en klocka. För att testa om klasserna fungerar som de ska har tre tester utformats. Dessa tester kommer också att behandlas i rapporten. Denna rapport kommer även att behandla hur programmet och testerna körs.

2. Användarhandledning

2.1 Filer

2.1.1 `NumberDisplay.java`

Denna klass har tre privata klassglobala variabler. `minLimit`, `maxLimit` och `value` som alla är integers.

`public NumberDisplay(int minLimit, int maxLimit)`

Denna publika metod är en konstruktor som tar två argument, `minLimit` och `maxLimit` och sätter deras värde till de globala variablerna med samma namn. Om `minLimit` inte är mindre än `maxLimit` hanteras ett undantag.

`public int getValue()`

Denna publika metod returnerar den globala variabeln `value` som innehåller klassens värde.

`public void setValue(int newValue)`

Denna publika metod tar ett argument, integern `newValue`, och skriver över `value` med `newValue` om `newValue` inte är större än `maxLimit` eller mindre än `minLimit` för då hanteras ett undantag. Om `newValue` är lika med `maxLimit` sätts `value` till `minLimit`.

`public String getDisplayValue()`

Denna publika metod returnerar `value` som en `String` och med lika många tecken som `maxLimit`. Om `value` har färre tecken än `maxLimit` så fylls det på med nollor framför tills de har lika många tecken.

`public void increment()`

Denna publika metod ökar på `value` med 1. Om `value` är lika med `maxLimit` så sätts `value` till `minLimit`.

public boolean didWrapAround()

Denna publika metod kollar om **value** precis har återställts till **minValue** genom att kolla om **value** är lika med **minValue**. Om likheten stämmer returneras **true** annars **false**.

2.1.2 Clock.java

Denna klass har en privat klassglobal String displayString som är själva stringen som representerar varje klockslag. Klassen har även två privata klassglobala NumberDisplays, hours och minutes som gör det som namnet antyder.

public Clock()

Denna publika metod är en konstruktor som sätter klockans timtid till 0 och minuttid till 0, d.v.s. sätter den klockan till '00:00'.

public Clock(int hour, int minute)

Denna publika metod är en konstruktor som sätter klockans timtid till **hour** och minuttid till **minute**, d.v.s. sätter den till '**hour:minute**'.

public void timeTick()

Denna publika metod simulerar ett klockslag och kör metoden minutes.increment som finns i **NumberDisplay**. Om **minutes.increment** får klockans minuter att nå 60 så körs **hours.increment**.

public void setTime(int hour, int minute)

Denna publika metod sätter clockans tid till **hour** och **minute**. Om **hour** är större än 24 eller mindre än 0 eller om **minute** är större än 60 eller mindre än 0 så genereras ett undantag.

public String getTime()

Denna publika metod returnerar den klassglobala variabeln displayValue.

private void updateDisplay()

Denna privata metod sätter den klassglobala variabeln displayString till timmarna plus kolon plus minuterna. T.ex. om timmarna är 2 och minuterna 43 så blir displayString '02:43'.

2.1.3 AlarmClock.java

Denna klass ärver från Clock. Klassen innehåller en privat klassglobal variabel vid namn alarmStatus som är en bool som är true om larmet är på och false om det är av. Klassen har även en privat klassglobal Clock vid namn alarm som är en klocka.

AlarmClock(int hour, int minute)

Denna metod är en konstruktor som kör clocks konstruktor och sätter klockans tid till hour och minute.

```
public void setAlarm(int hour, int minute)
```

Denna metod sätter ett larm vid tiden hour och minute. Den slår även på larmet.

```
public boolean isTriggered()
```

Denna metod kollar om larmet ska ringa vid denna tidpunkt. Om den ska det så skriver metoden ut "alarm" och returnerar true. Annars returnerar den false.

```
public void turnOn()
```

Denna metod sätter på larmet.

```
public void turnoff()
```

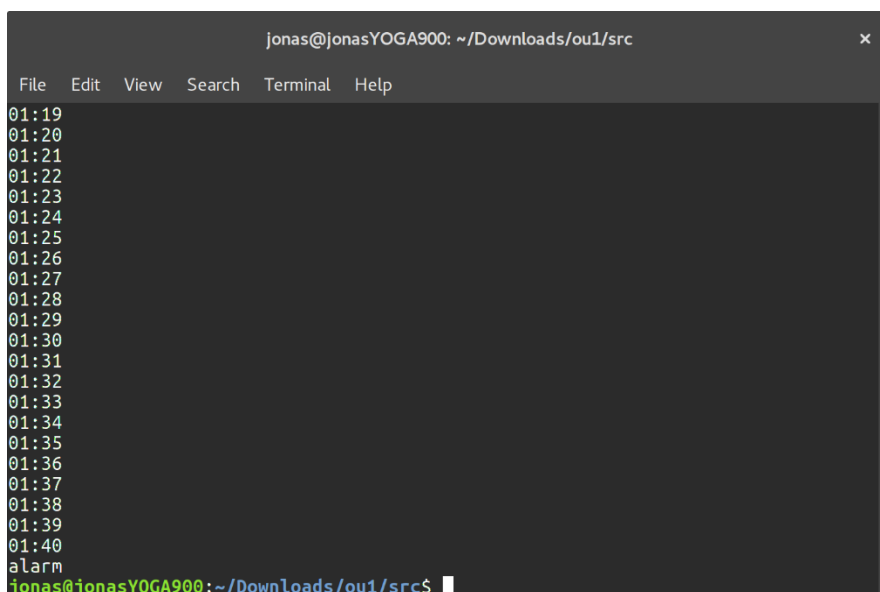
Denna metod sätter på larmet.

2.2 Körning av programmet

För att kunna enkelt simulera en körning av programmet har jag skapat en main-fil som simulerar en körning av programmet. För att köra denna fil från terminalen skriver man följande:

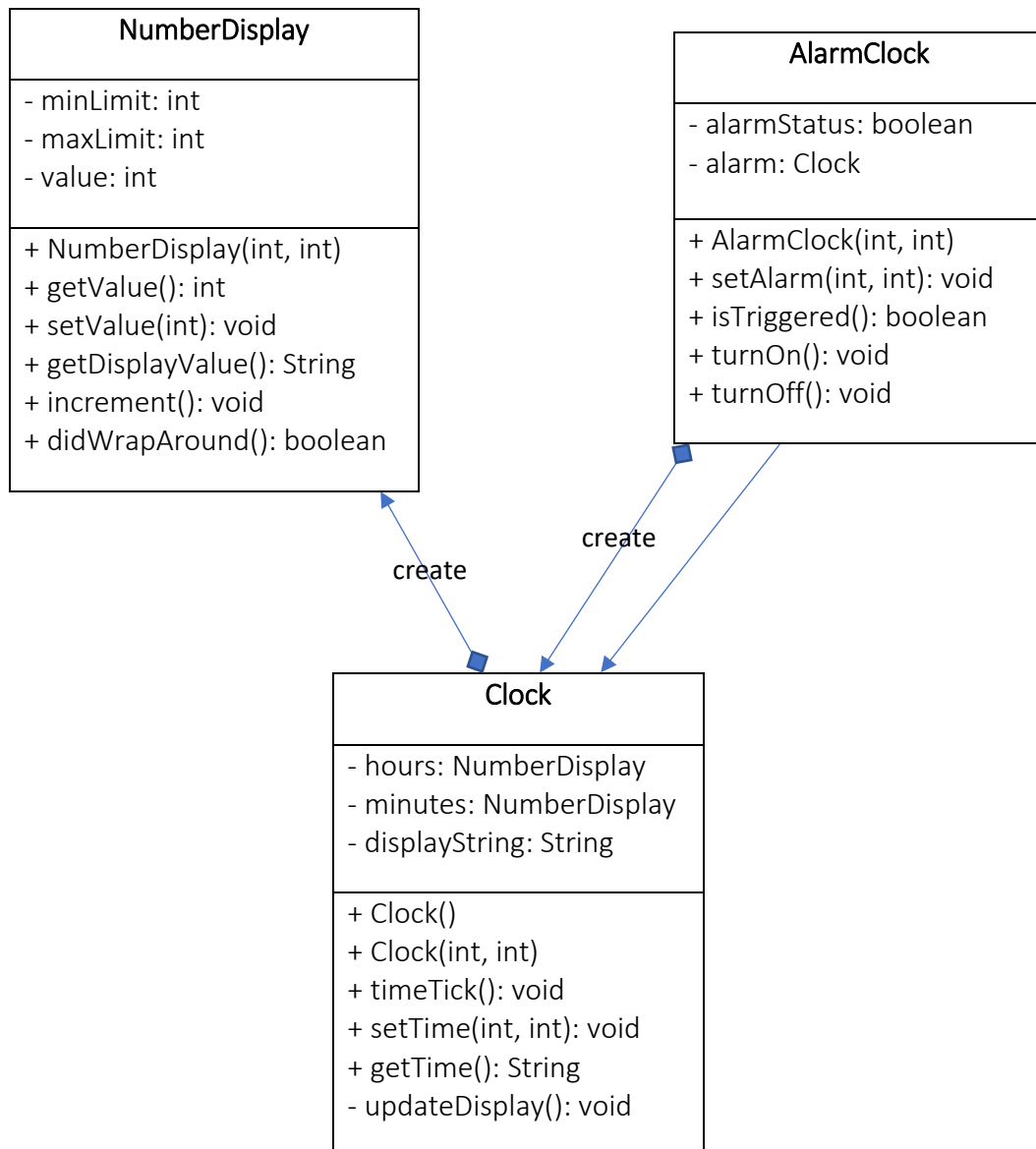
Först navigerar man till mappen i terminalen med kommandot "**cd java/ou1/src**" fast med den väg där dina filer ligger. När man sedan kommit fram till rätt mapp ska man skapa en mapp vid namn **output** så slipper man ha de kompilerade filerna i samma mapp som sina .java filer. Sedan skriver man "**javac -d output Main.java Clock.java AlarmClock.java NumberDisplay.java**" för att kompilera koden. Sedan skriver man "**java -cp output Main**" för att köra de kompilerade filerna. Då kommer programmet att köras och skriva ut tidpunkt i terminalen tills klockan ringer och den skriver ut "alarm".

För att kompilera junit-testerna måste man först navigera till mappen där junit är lokaliserad. T.ex. så går man först till root mappen och skriver i terminalen "**javac -cp /usr/share/java/junit4.jar File.java FileTest.java**" för att kompilera filen. Sedan skriver man "**java -cp ./usr/share/java/junit4.jar org.junit.runner.JUnitCore FileTest**".



```
jonas@jonasYOGA900: ~/Downloads/ou1/src
File Edit View Search Terminal Help
01:19
01:20
01:21
01:22
01:23
01:24
01:25
01:26
01:27
01:28
01:29
01:30
01:31
01:32
01:33
01:34
01:35
01:36
01:37
01:38
01:39
01:40
alarm
jonas@jonasYOGA900:~/Downloads/ou1/src$
```

3. Klassdiagram



4. Tester

4.1 NumberDisplayTest.java

```
public void createNumberDisplayMaxIsGreaterThanMin()
```

Detta test skapar en NumberDisplay med parametrarna 1, 2. Då maxLimit är större än minLimit borde inget undantag borde kastas.

```
public void createNumberDisplayMinIsGreaterThanMax()
```

Detta test skapar en NumberDisplay med parametrarna 2, 1. Då minLimit är större än maxLimit borde ett undantag kastas.

```
public void createNumberDisplayMinIsEqualToMax()
```

Detta test skapar en NumberDisplay med parametrarna 1, 1. Då minLimit är lika med maxLimit borde ett undantag kastas.

```
public void setValueWithinRange()
```

Detta test skapar en NumberDisplay med parametrarna 0, 2. Sedan sätts värdet 1. Inget undantag borde kastas.

Följande fyra test gör ungefär samma som förra testet fast undantag borde kastas. Kolla namn.

```
public void setValueLesserThanMinValue()
```

```
public void setValueGreaterThanMaxValue()
```

```
public void setValueEqualToMax()
```

Detta test kollar överstående plus kollar om värdet återställs till minLimit.

```
public void setValueEqualToMin()
```

```
public void shouldGetValue()
```

Detta test skapar en NumberDisplay med parametrarna 0, 2. Den sätter värdet till 1. Sedan hämtas värdet och om värdet inte är lika med 1 kastas ett undantag.

```
public void getDisplayValue10characters()
```

Detta test skapar en NumberDisplay med parametrarna 0, 1000000000. Sedan sätts värdet 123. Sedan hämtas värdet. Om värdet inte är "0000000123" så kastas ett undantag.

```
public void shouldIncreaseValueByOne()
```

Detta test skapar en NumberDisplay med parametrarna 0, 4 och sätter värdet till 1. Sedan körs metoden increment som borde öka värdet med ett. Sedan hämtas värdet. Om värdet är inte är lika med 2 så kastas ett undantag.

```
public void shouldResetValueToMin()
```

Detta test skapar en NumberDisplay med parametrarna 0, 4 och sätter värdet till 3. Sedan körs metoden increment som borde öka värdet med ett och återställa värdet till 0. Sedan hämtas värdet. Om värdet är inte lika med 0 så kastas ett undantag.

```
public void didWrapAround()
```

Detta test skapar en NumberDisplay med parametrarna 0, 4 och sätter värdet till 3. Sedan körs metoden increment som borde öka värdet med ett och återställa värdet till 0. Sedan kollar det om värdet har återställts till 0 med metoden didWrapAround. Om den gjort det så kastas ett undantag.

4.2 ClockTest.java

```
public void createClockNoParams()
```

Detta test skapar en Clock utan parametrar och hämtar tiden. Om tiden är "00:00" så har testet lyckats, annars kastas ett undantag.

```
public void createClockWithParams()
```

Detta test skapar en Clock med parametrarna 1, 20 och hämtar tiden. Om tiden är "01:20" så har testet lyckats, annars kastas ett undantag.

De kommande fyra testerna gör exakt det som namnen antyder:

```
public void createClockWithHourTooSmall()
```

```
public void createClockWithHourTooGreat()
```

```
public void createClockWithMinuteTooSmall()
```

```
public void createClockWithMinuteTooGreat()
```

```
public void timeTickMinute()
```

Detta test skapar en Clock utan parametrar och låter den ticka en gång. Om klockan nu är "00:01" har testet lyckats, annars kastas ett undantag.

```
public void timeTickLastMinute()
```

Detta test skapar en Clock med parametrarna 12, 59 och låter den ticka en gång. Om klockan nu är "13:00" har testet lyckats, annars kastas ett undantag.

4.3 AlarmClockTest.java

public void createAlarmClockWithoutParams ()

Detta test skapar en alarmClock utan parametrar och hämtar tiden. Om tiden är "00:00" så har testet lyckats, annars kastas ett undantag.

public void createAlarmClockWithParams ()

Detta test skapar en alarmClock med 5 och 12 som parametrar och hämtar tiden. Om tiden är "05:12" så har testet lyckats, annars kastas ett undantag.

public void setAlarmAndCheckTrigger ()

Detta test skapar en alarmClock och sätter tiden till "12:24". Testet sätter även ett alarm till "12:24". Sedan kollar testet om alarmet ringer. Om det ringer så har testet lyckats, annars kastas ett undantag.

public void setAlarmAndTurnOnCheckTrigger ()

Detta test skapar en alarmClock och sätter tiden till "12:24". Testet sätter även ett alarm till "12:24". Sedan stängs alarmet av och sätts sedan på. Sedan kollar testet om alarmet ringer. Om det ringer så har testet lyckats, annars kastas ett undantag.

public void setAlarmAndAlarmStatusFalse ()

Detta test skapar en alarmClock och sätter tiden till "12:24". Testet sätter även ett alarm till "12:24". Sedan stängs alarmet av. Sedan kollar testet om alarmet ringer. Om det inte ringer så har testet lyckats, annars kastas ett undantag.