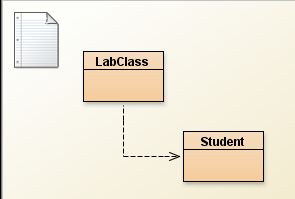
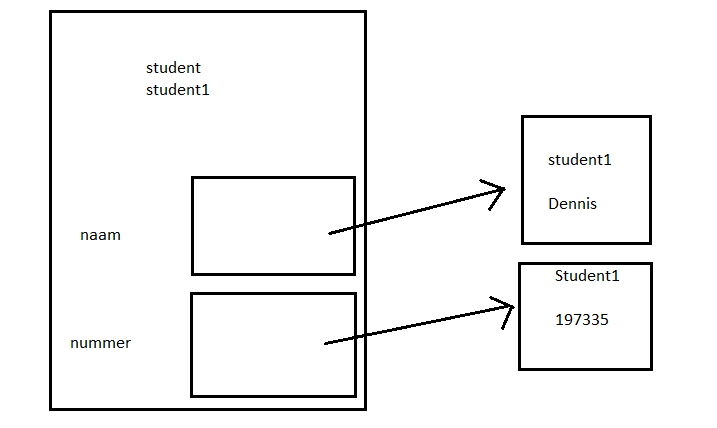
Bestudeer en ***maak de opgaven*** 3.1 t/m 3.9, 3.13 t/m 3.30, 3.32 t/m 3.46.

3.1

Klassendiagram



objectdiagram



A: Objectendiagram: Het objectdiagram toont de objecten en hun onderlinge relaties op een bepaald moment tijdens de uitvoering van een toepassing. Het geeft informatie over objecten terwijl het programma wordt uitgevoerd. Het geeft een dynamisch beeld van een programma.

B: Klassendiagram: Het klassendiagram bevat de klassen van een toepassing en de onderlinge relaties. Het bevat informatie over de broncode. Het is een statisch beeld van een programma.

3.2

Als er meer velden bij komen

3.3

Als er meer methodes bij komen

3.4

private instructor Tutor;

3.5

Ok

3.6

hij geeft geen fout en doet dan niks. In ieder geval een foutmelding laten weergeven.

3.7

Dan kan de waarde 0 niet worden gebruikt.

3.8

dan kan het ook onder 0 en hoger dan de limiet worden

3.9

false

true

false

false

true.

3.13

nee, als je boven je limiet blijft ie hetzelfde en veranderd er niets.

3.14

bij de 2e maakt hij van de waarde een string zonder dat er tekens voor komen.

3.15

Die berekent het rest getal van een deling.

3.16

2

3.17

-8/3 geeft ie -2 als antwoord.

3.18

Van 0 t/m 4

3.19

Alle getallen denk ik.

3.20

Increment is verhogen. Dus de methode increment verhoogt telkens met 1

En als die boven de 60 (limit) komt begint hij weer op 0

3.21

public void increment()

{

value++

if ( value =>limit) value = 1;

}

3.22

Ok

3.23

00:00

dat is de start tijd, omdat er nog niks ingevoerd is.

3.24

60x, zoveel seconden zitten in een minuut.

Met de setTime functie

3.26

public Editor(String varA, int varB)

3.27

Private Rectangle window;

Window = new Rectangle(50, 40);

3.28

Hij maakt eerst de uren en de minuten display aan met de max tijden.

Daarna zet hij de juiste tijden met de waarden die hij binnenkrijgt van de parameters.

3.29

Omdat deze de setTime functie aanroept, die daarna op zijn beurt wel de updateDisplay aanroept.

3.30

P1.print(“hoi.doc”, FALSE);

P1.getStatus(500);

3.32

Int hours = 2;

Int minutes = 23;

If(minutes >= 60){

Minutes = 0;  
 hours += 1;

If(hours > 12){

Hours = 0;

}

}

Int hourDisplay;

If(hours > 12){  
 hourDisplay = hours-12;

hourDisplay = hourDisplay + “pm”;

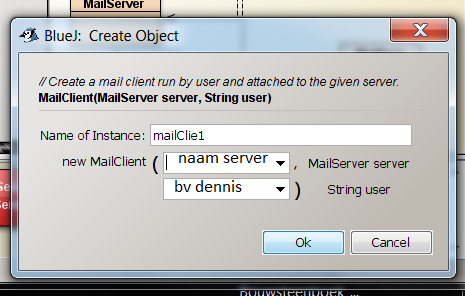
}

de laatste ik vind 24 uur makkelijker.

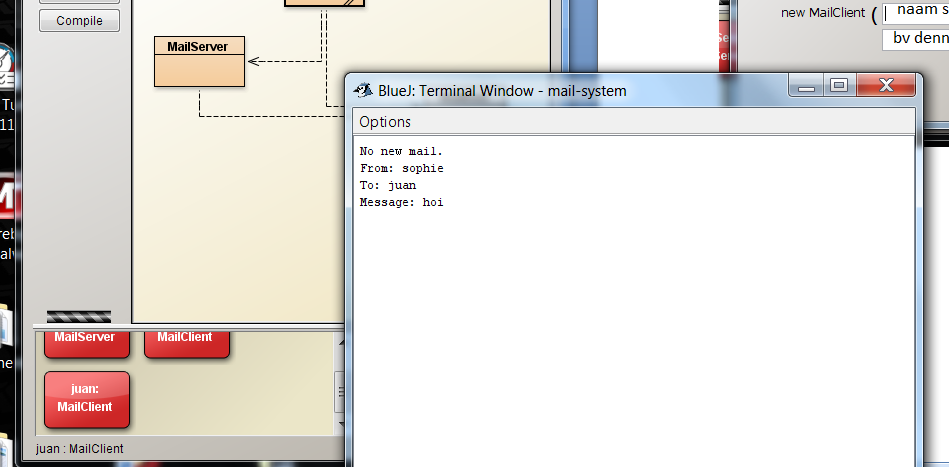
3.33

Ok

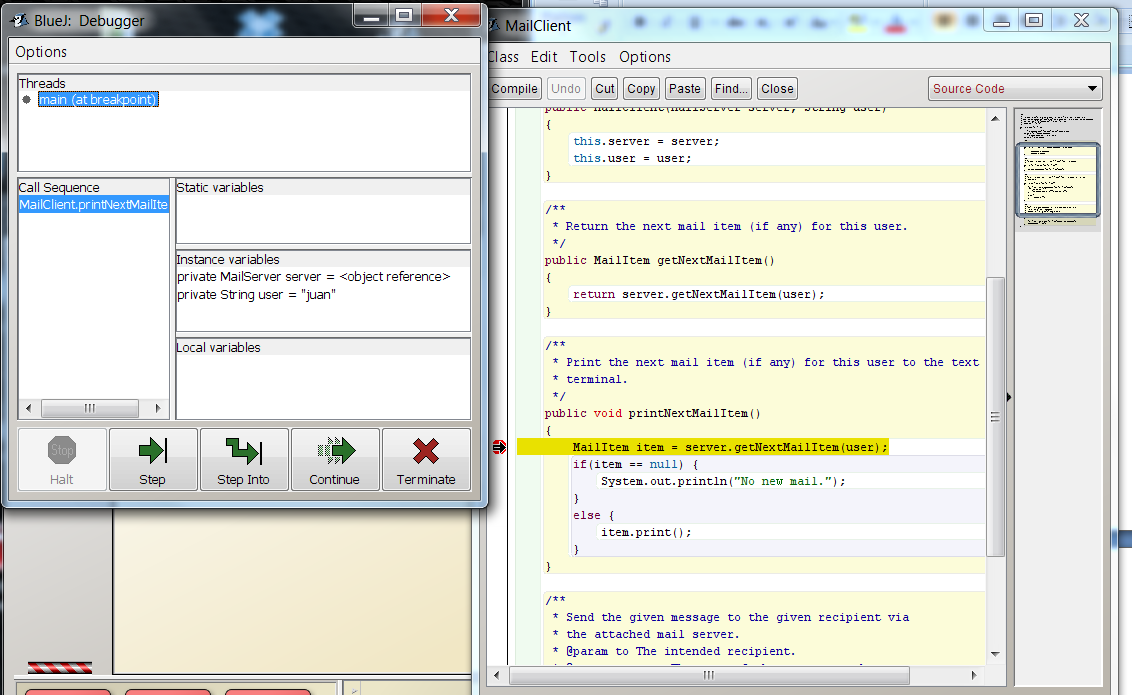
3.34



3.35



3.36



3.37

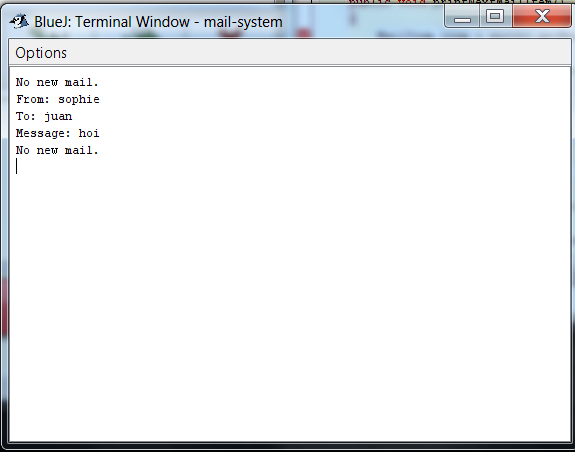
zwarte pijl komt in beeld.

3.38

de opvolgende regel, ja is uitgekomen.

3.39

nu komt er een terminal window met de laatste berichten.



3.40

Hij stapt de regels code die hij gaat uitvoeren door en gaat de betreffende dingen doen.

3.42

Als je een nieuwe gebruiker maakt, bepaal je de server en dan naam van het object. (Client)

Als je met deze object dan een mailitem wil gaan sturen, maakt hij een nieuwe mail item object aan met de MailItem klasse. In deze klasse worden dan de variabelen gezet van het bericht dat je wil gaan sturen inclusief de ontvanger. Vervolgens worden deze gegevens van dat object verstuurd met de server.post actie.

3.43 3.44 ok

3.45

/\*\*

\* A class to model a simple mail item. The item has sender and recipient

\* addresses and a message string.

\*

\* @author David J. Barnes and Michael Kölling

\* @version 2011.07.31

\*/

public class MailItem

{

// The sender of the item.

private String from;

// The intended recipient.

private String to;

// The text of the message.

private String message;

//subject line

private String subject;

/\*\*

\* Create a mail item from sender to the given recipient,

\* containing the given message.

\* @param from The sender of this item.

\* @param to The intended recipient of this item.

\* @param message The text of the message to be sent.

\*/

public MailItem(String from, String to, String message , String subject)

{

this.from = from;

this.to = to;

this.message = message;

this.subject = subject;

}

/\*\*

\* @return The sender of this message.

\*/

public String getFrom()

{

return from;

}

/\*\*

\* @return The intended recipient of this message.

\*/

public String getTo()

{

return to;

}

/\*\*

\* @return The text of the message.

\*/

public String getMessage()

{

return message;

}

/\*\*

\* Print this mail message to the text terminal.

\*/

public void print()

{

System.out.println("From: " + from);

System.out.println("To: " + to);

System.out.println("Message: " + message);

System.out.println("Message: " + subject);

}

}

3.46

/\*\*

\* Write a description of class Screen here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Screen{

private int xres;

private int yres;

public Screen(int xRes, int yRes)

{

this.xres = xRes;

this.yres = yRes;

int pixels;

pixels = numberOfPixels();

if(pixels > 2000000){

clear(false);

}

}

public int numberOfPixels()

{

return 5;

}

public void clear(boolean invert)

{

return;

}

}