DD1339 Introduktion till datalogi 2013/2014

Uppgift nummer: Hemuppgift 4

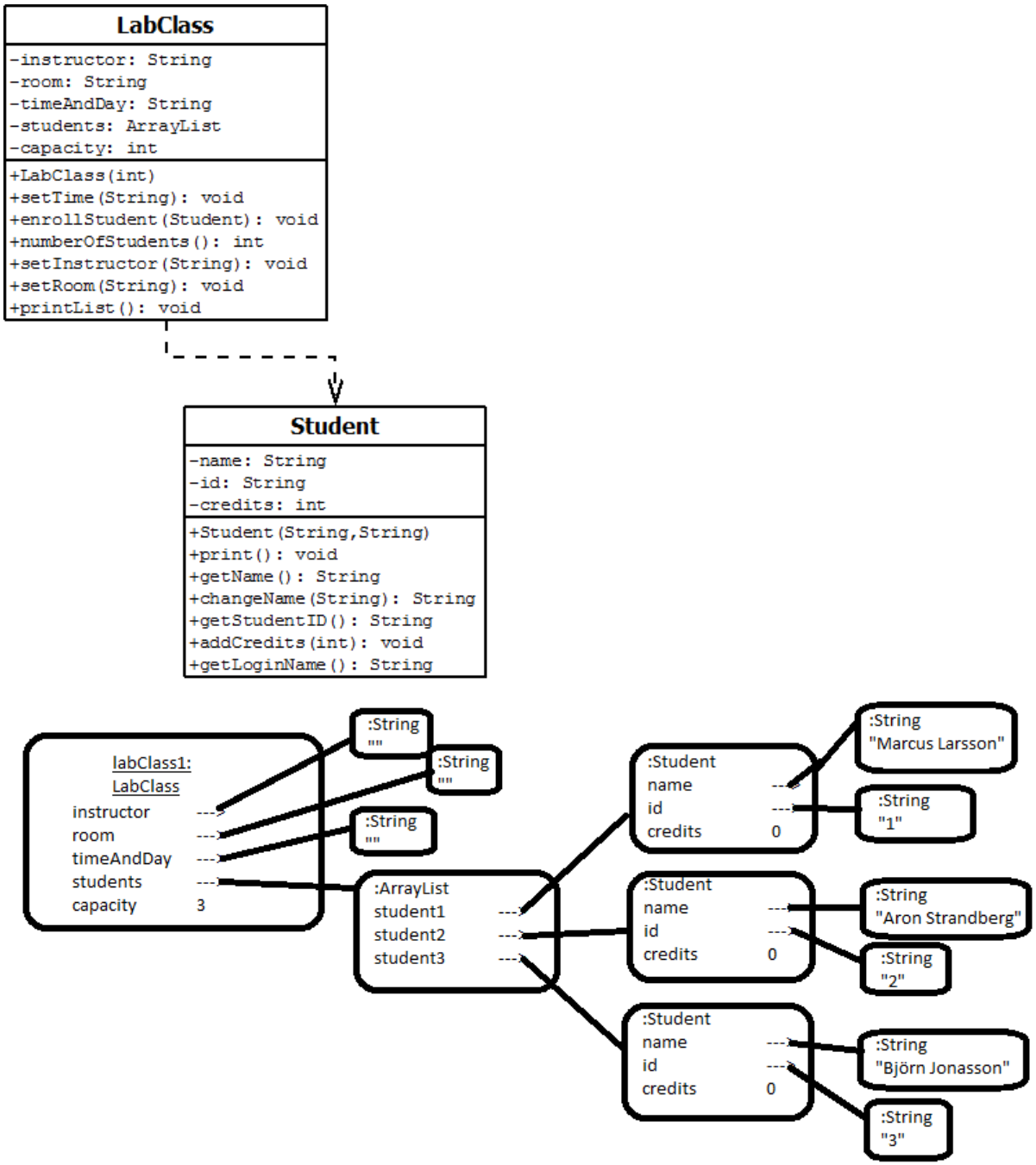
Namn: Marcus Larsson

Grupp nummer: 5

Övningsledare: Marcus Dicander

Betyg: ..... Datum: ..... Rättad av: .....

Exercise 3.1



Skillnaden mellan dessa diagram är att klassdiagrammen innehåller alla fälten och metoder som ett objekt av klassen kan använda sig av. Fälten i klassdiagrammen innehåller inget fast data, utan data sätts till variablerna när programmet körs och det utgör vad som sedan finns i objektdiagrammet.

Jag har valt att avsluta objektträdet vid String. Men en Sting är egentligen en char array med alla bokstäver som strängen innehåller. En ArrayList innehåller också fält som sizeoch count. Men för att få diagrammet tof jag bort dessa "mindre" viktiga i uppgiftens syfte. Där det bara står en siffra utan "" är av den primitiva typen int.

Exercise 3.2

Klassdiagram ändras genom att ändra koden i programmet. När programmet startar är klasserna definerade och förblir så under tiden programmet körs.

Exercise 3.3

Ett objektdiagram existerar endast under tiden programmet körs. Under tiden ett program körs kan också objektets stadier ändras och därmed ändras också objektdiagrammet. Programmet ändrar på objekt diagrammen.

Exercise 3.9

Följande uttryck blir true:

! false (inte falskt = sant)

(34 != 33) && ! false (34 är inte lika med 33 och sant. Blir sant endast om båda uttryck är sanna i &&. I detta fall är båda uttryck sanna.)

Dem andra uttrycken blir falska. || Betyder "eller" och då behöver endast ett av uttrycken vara sanna.

i uttrycket (2 > 2) || ((4 == 4) && (1 < 0)) och (2 > 2) || (4 == 4) && (1 < 0) är det p.g.a. det sista uttrycket som gör att dem blir falska. && operatorn hänger ihop med förgående term, även om det inte sitter en parantes mellan dessa. Därför vlir uttrycket på höger sida av || också falskt.

Exercise 3.10

Ett uttryck som blir sant om antingen både a och b är true eller om båda är false: (a==b) eller (a&&b || !a&&!b)

Exercise 3.11

Ett uttryck som är sant om endast en av a eller b är true: (a!=b)

Exercise 3.12

De Morgans law
Om man vill skriva a&&b utan att använda && får man utgå från denna lag. Den säger följande:
inte a och b = inte a eller inte b. Detta skulle skrivas i java:
!(a&&b) = !a||!b. För att då vända detta till det "positiva" resultatet får vi tillföra ett till "inte" tecken.
!(!a||!b) = a&&b

Exercise 3.21

```
public void increment()
  {
    value+= 1;
```

```
    if(value>=limit){
       value=0;
    }
  }
```

Båda sätten utför samma sak. Jag skulle föredra denna if-sats då den är lättare att läsa. Modolus operatorn bör bara användas när man vill få ut just resten av en division.

Exercise 3.26

En signatur av en konstruktor som svarar mot anropet: new Editor("readme.txt", -1)

Editor(String fileName, int numberToUpdate)

Exercise 2.27

Rectangle window = new Rectangle(10,20);

Exercise 3.30

Exempel på metodanrop av Printer objektet p1.

```
p1.print("reademe.txt", true);
p1.print("blueprint.ppt",false);
int status1 = p1.getStatus(20);
int status2 = p1.getStatus(100);
```

Exercise 3.31

För att göra ändringen så att programmet bara lagrar värden upp till 12 kan jag ändra i 3 metoder. Båda konstruktorerna måste skapa en numberDisplay(12) istället för 24. Och sedan behöver jag hantera fallet då värdet är 0 och sätta det till 12 istället. Det kan göras i updateDisplay(). I denna version implementeras inte am, pm beteckningar. Se källkod nedan på ändrade metoder.

```
/**
    * Constructor for ClockDisplay objects. This constructor
    * creates a new clock set at 12:00.
    */
   public ClockDisplay()
   {
      hours = new NumberDisplay(12);
      minutes = new NumberDisplay(60);
      updateDisplay();
   }

   /**
    * Constructor for ClockDisplay objects. This constructor
    * creates a new clock set at the time specified by the
    * parameters.
```

```
     */
    public ClockDisplay(int hour, int minute)
    {
        hours = new NumberDisplay(12);
        minutes = new NumberDisplay(60);
        setTime(hour, minute);
    }
/**
     * Update the internal string that represents the display.
     */
    private void updateDisplay()
    {
        int hours=this.hours.getValue();
        if(hours==0){
            hours=12;
        }
        displayString = hours + ":" +
                        minutes.getDisplayValue();
    }
```

Exercise 3.32

Implementera 12-timarsklockan genom att bara ändra hur tiden visas och inte själva tiden i sig.

Jag tycker att detta är ett bättre sätt att implementera på. Det ger en enkel möjlighet till att hantera när am eller pm ska visas. Man lagrar fortfarande värden 0 - 24 som man kan använda sig av i beräkningar om det skulle behövas. Fler möjligheter är alltid bra tycker jag. Om det hade varit ett väldigt stort program hade vi också kunnat känna oss mer säkra på att vi inte förstör något annat i denna ändring. Eftersom här ändrar vi bara i hur tiden visas, inte vad tiden är.

Endast klassen ClockDisplay är ändrad. Det är i metoden updateDisplay som jag översatte 24timmarsklockan till 12timmarsklocka. Se kod nedan.

```
    /**
     * Update the internal string that represents the display.
     * Under the hood it's a 24-hour clock, but when displaying the value, this method
     * is translating it to 12-hour clock with am or pm marks.
     */
    private void updateDisplay()
    {
        int hours = 0;
        String ampm = "";
        if(this.hours.getValue()>12){
            hours = this.hours.getValue()-12;
            ampm = "pm";
        } else {
            hours = this.hours.getValue();
            ampm = "am";
        }
        if (this.hours.getValue()==0){
            hours = 12;
```

```
        }
        displayString = hours + ":" +
                        minutes.getDisplayValue()+ampm;
    }
}
```

Exercise 3.34

Efter att ha skapat en server med 3 stycket klienter ser ett objektdiagram ut som nedan. Clienterna har pekar sin variabel server till mailServ1. Serverobjektet använder sig av en lista som variablen items pekar på. Den listan har 3 fält, ett av den är själva listan som lagrar objekt. I detta fallet mailitems som skickas mellan klienterna.