

DV017A
Java-programmering
Laboration 4

Jonas Sjöberg
860224
Högskolan i Gävle
`tel12jsg@tudent.hig.se`
<https://github.com/jonasjberg>

Datum: 2015-08-14
Kursansvarig lärare: Atique Ullah

Sammanfattning

Laborationsrapport för *DV017A – Inledande programmering i Java*, distanskurs på Högskolan i Gävle. Blandade uppgifter i grundläggande praktisk programmering i Java. Innehåller labinstruktioner, egna lösningar, källkod, skärmdumpar och kommentarer.

Innehåll

1	Uppgift 1	3
1.1	Instruktioner	3
1.2	Källkod	3
1.2.1	Lab4Uppg01.java	3
1.2.2	UserInputFilter.java	6
1.3	Skärmdump	8
2	Uppgift 2	8
2.1	Instruktioner	8
2.2	Källkod	10
2.2.1	Lab4Uppg02.java	10
2.2.2	Konto.java	11
2.3	Skärmdump	12
3	Uppgift 3	14
3.1	Instruktioner	14
3.2	Källkod	14
3.2.1	Lab4Uppg03.java	14
3.2.2	Student.java	16
3.2.3	Person.java	18
3.3	Skärmdump	20

Figurer

1	Körning av koden till Uppgift 1	9
2	Körning av koden till Uppgift 2	13
3	Körning av koden till Uppgift 3	21

1 Uppgift 1

1.1 Instruktioner

Skriv ett program där man ska skriva in ett antal heltal. Hur många tal ska den som kör programmet själv bestämma, dock max 30. Heltalen ska lagras i en array.

Programmet ska sedan beräkna summan av talen, beräkna talens medelvärde (exakta), ta reda på vilket av talen som är störst och minst. Exempel på hur utskriften kan se ut (inmatning från tangentbordet = **fetstil**):

```
Hur många tal vill du mata in (max 30)? *5*
*4*
*5*
*3*
*7*
*6*
Summa: 25
Medelvärde: 5.0
Minsta värde: 3
Största värde: 7
```

1.2 Källkod

1.2.1 Lab4Uppg01.java

```
1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #4    Uppgift 1
8  */
9
10 import java.util.List;
11 import java.util.Vector;
12 import java.util.logging.Level;
13 import java.util.logging.Logger;
14
15 public class Lab4Uppg01
16 {
17     private final static String  TXT_QUERY          = "Hur många tal vill du mata in (max 30)? ";
18     private final static int     MAX_NUMBER_OF_INPUTS = 30;
19     private final static int     MIN_NUMBER_OF_INPUTS = 0;
20
21     private static List<Integer> numbers = new Vector<Integer>();
22     private static int           sum;
23     private static double        average;
24     private static int           min;
25     private static int           max;
26
27     /* Använder en "Logger" för att hantera debug-meddelanden */
28     private static Logger        logger = Logger.getAnonymousLogger();
29
30     public static void main(String[] args)
31     {
32         /* Debug-meddelanden av/på */
```

```

33     logger.setLevel(Level.OFF);
34
35     getInput(getNumberOfInputs(), numbers);
36     doCalculation(numbers);
37     printResults();
38
39     System.exit(0);
40 }
41
42 /**
43  * Frågar efter ett antal tal att senare utföra beräkningar på.
44  * @return antal tal som ska matas in
45  */
46 private static int getNumberOfInputs()
47 {
48     int n;
49
50     do {
51         n = UserInputFilter.getPositiveInt(TXT_QUERY);
52     } while (n <= MIN_NUMBER_OF_INPUTS || n > MAX_NUMBER_OF_INPUTS);
53
54     return n;
55 }
56
57 /**
58  * Fyller en lista med ett antal heltal med användarens inmatning.
59  * @param numberOfInputs    antal heltal som ska matas in i listan
60  * @param numbers           lista att stoppa talen i
61  */
62 private static void getInput(int numberOfInputs, List<Integer> numbers)
63 {
64     int currentPos = 0;
65
66     while (currentPos++ < numberOfInputs) {
67         int in = UserInputFilter.getPositiveInt("(" + currentPos + "): ");
68         numbers.add(in);
69     }
70 }
71
72 /**
73  * Utför beräkningar på en lista.
74  * @param numbers    lista att utföra beräkningar på
75  */
76 private static void doCalculation(List<Integer> numbers)
77 {
78     logger.entering("doCalculation", "getSum");
79     sum = getSum(numbers);
80     logger.entering("doCalculation", "getAverage");
81     average = getAverage(numbers);
82     logger.entering("doCalculation", "getMin");
83     min = getMin(numbers);
84     logger.entering("doCalculation", "getMax");
85     max = getMax(numbers);
86 }
87
88 /**
89  * Returnerar summan av talen i en lista.
90  * @param numbers    lista att undersöka
91  * @return summan av talen i listan
92  */

```

```

93     private static Integer getSum(List<Integer> numbers)
94     {
95         int sum = 0;
96
97         for (Integer n : numbers) {
98             sum += n;
99             logger.info("sum = " + sum);
100        }
101
102        return sum;
103    }
104
105    /**
106     * Returnerar medelvärde för talen i en lista.
107     * @param numbers lista att undersöka
108     * @return medelvärde av talen i listan
109     */
110    private static double getAverage(List<Integer> numbers)
111    {
112        double average = getSum(numbers) / numbers.size();
113
114        logger.info("average = " + sum);
115        return average;
116    }
117
118    /**
119     * Returnerar det minsta värdet i en lista.
120     * @param numbers lista att undersöka
121     * @return det minsta värdet i listan
122     */
123    private static int getMin(List<Integer> numbers)
124    {
125        int index = 0;
126        int min = Integer.MAX_VALUE;
127
128        do {
129            if (numbers.get(index) < min)
130                min = numbers.get(index);
131
132            index++;
133            logger.info("min = " + min);
134
135        } while (index < numbers.size());
136
137        return min;
138    }
139
140    /**
141     * Returnerar det största värdet i en lista.
142     * @param numbers lista att undersöka
143     * @return det största värdet i listan
144     */
145    private static int getMax(List<Integer> numbers)
146    {
147        int index = 0;
148        int max = Integer.MIN_VALUE;
149
150        do {
151            if (numbers.get(index) > max)
152                max = numbers.get(index);

```

```

153         index++;
154         logger.info("max = " + max);
155     } while (index < numbers.size());
156
157     return max;
158 }
159
160 /**
161  * Skriver ut resultatet av beräkningarna.
162  */
163 private static void printResults()
164 {
165     System.out.println("Summa: " + sum);
166     System.out.println("Medelvärde: " + average);
167     System.out.println("Minsta värde: " + min);
168     System.out.println("Största värde: " + max);
169 }
170 }
171 }
172 }

```

Lab4Uppg01.java

1.2.2 UserInputFilter.java

```

1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #4
8  */
9
10 import java.util.Scanner;
11
12 public class UserInputFilter
13 {
14     /**
15      * Hämta positivt heltal från användaren.
16      * @param msg meddelande vid förfrågan
17      * @return ett positivt heltal
18      */
19     public static int getPositiveInt(String msg)
20     {
21         @SuppressWarnings("resource")
22         Scanner scan = new Scanner(System.in);
23         int vetInt = 0;
24
25         do {
26             queryUser(msg);
27
28             while (!scan.hasNextInt()) {
29                 queryUser(msg);
30                 scan.next();
31             }
32
33             vetInt = scan.nextInt();

```

```

34
35     } while (vetInt <= 0);
36
37     return vetInt;
38 }
39
40 /**
41  * Hämta positivt heltal av typ "long" från användaren.
42  * @param msg    meddelande vid förfrågan
43  * @return       ett positivt heltal
44  */
45 public static long getPositiveLong(String msg)
46 {
47     @SuppressWarnings("resource")
48     Scanner scan = new Scanner(System.in);
49     long john = 0;
50
51     do {
52         queryUser(msg);
53
54         while (!scan.hasNextLong()) {
55             queryUser(msg);
56             scan.next();
57         }
58
59         john = scan.nextLong();
60
61     } while (john <= 0);
62
63     return john;
64 }
65
66 /**
67  * Skriv ut meddelande till användaren.
68  * @param msg    meddelande att skriva ut
69  */
70 private static void queryUser(String msg)
71 {
72     if (msg != null)
73         System.out.print(msg);
74 }
75
76 /**
77  * Ställer en ja/nej fråga och väntar på att användaren svarar 'j' eller 'n'
78  * @param msg    meddelande att skriva ut
79  * @return       sant om svaret är 'j', falskt om svaret är 'n'
80  */
81 public static boolean getYesNoAnswer(String msg)
82 {
83     @SuppressWarnings("resource")
84     Scanner scan = new Scanner(System.in);
85     String token = "";
86
87     do {
88         queryUser(msg);
89         token = scan.nextLine().trim().toLowerCase();
90
91         if (token.equals("j")) {
92             return true;
93         } else if (token.equals("n")) {

```

```

94         return false;
95     }
96     } while (true);
97 }
98
99 /**
100  * Hämtar en textsträng från användaren.
101  * @param msg    meddelande att skriva ut
102  * @return       en textsträng
103  */
104 public static String getString(String msg)
105 {
106     @SuppressWarnings("resource")
107     Scanner scan = new Scanner(System.in);
108     String s = null;
109
110     do {
111         queryUser(msg);
112         s = scan.nextLine();
113     } while (s == null);
114
115     return s;
116 }
117 }

```

UserInputFilter.java

1.3 Skärmdump

Se Figur 1 för skärmdump på körning av koden i Sektion 1.2.1 och Sektion 1.2.2.

2 Uppgift 2

2.1 Instruktioner

Skriv en klass Konto som representerar ett bankkonto. Klassen ska innehålla data om ett bankkonto enligt följande (alltså klassens instansvariabler):

```

saldo (double)
kundnr (int)
kundnamn (String)

```

Klassen ska även ha en klassvariabel (static variabel) som heter:

```

räntesats (double)

```

Varför ska den vara en klassvariabel? Jo, eftersom alla konton har samma räntesats (ränta per år).

De metoder som ska finnas i klassen Konto är följande:

- insättning, metod som ökar saldot med parameterns värde.
- uttag, metod som minskar saldot med parameterns värde.
- hämtaSaldo, metod som returnerar saldo.
- hämtaKundnamn, metod som returnerar kundnamn.
- avläsRänteSats, *klassmetod* som returnerar räntesatsen.


```
Terminal - spock@ProBookII: ~/lab4-bin.symlink
spock@ProBookII:~/lab4-bin.symlink$ java Lab4Uppg01
Hur många tal vill du mata in (max 30)? 0
Hur många tal vill du mata in (max 30)? ----1111
Hur många tal vill du mata in (max 30)? trettio
Hur många tal vill du mata in (max 30)? 3
(1): 1
(2): 2
(3): 3
Summa:      6
Medelvärde:  2.0
Minsta värde: 1
Största värde: 3
spock@ProBookII:~/lab4-bin.symlink$
spock@ProBookII:~/lab4-bin.symlink$
spock@ProBookII:~/lab4-bin.symlink$ java Lab4Uppg01
Hur många tal vill du mata in (max 30)? 10
(1): 1
(2): 2
(3): 3
(4): 4
(5): 5
(6): 60
(7): 70
(8): 80
(9): 90
(10): 100
Summa:      415
Medelvärde:  41.0
Minsta värde: 1
Största värde: 100
spock@ProBookII:~/lab4-bin.symlink$ java Lab4Uppg01
Hur många tal vill du mata in (max 30)? 3
(1): 365
(2): 1337
(3): 6
Summa:      1708
Medelvärde:  569.0
Minsta värde: 6
Största värde: 1337
spock@ProBookII:~/lab4-bin.symlink$
```

Figur 1: Körning av koden till Uppgift 1

Det ska naturligtvis finnas en konstruktor.

Skriv en testklass (testprogram) som skapar två konton. Gör insättning och uttag med valfria belopp på kontona. Skriv sedan ut kontonas saldon och kundens namn. Skriv sist ut vilken räntesats de båda kontona har.

2.2 Källkod

2.2.1 Lab4Uppg02.java

```
1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #4    Uppgift 2
8   */
9
10 import java.util.List;
11 import java.util.Vector;
12
13 /**
14  * Klass 'Lab4Uppg02' testar klassen 'Konto'.
15  */
16 public class Lab4Uppg02
17 {
18     private static Konto      k1;
19     private static Konto      k2;
20     private static final List<Konto> konton = new Vector<Konto>();
21
22     public static void main(String[] args)
23     {
24         k1 = new Konto(3500000, 13, "Gibson Pälsmann");
25         k2 = new Konto(100, 8641, "Foobar Bar");
26
27         konton.add(k1);
28         konton.add(k2);
29
30         info(konton);
31
32         konton.get(0).insattning(2.50);
33         konton.get(1).insattning(4200);
34
35         info(konton);
36
37         konton.get(0).uttag(1000001);
38         konton.get(1).uttag(4500);
39
40         info(konton);
41
42         System.exit(0);
43     }
44
45     /**
46      * Skriver ut information om bankkonton i en lista.
47      * @param konton    listan med konton
48      */
```

```

49     private static void info(List<Konto> konton)
50     {
51         System.out.println("");
52         for (Konto k : konton) {
53             System.out.println("kundnamn: " + k.hamtaKundnamn());
54             System.out.println("saldo:      " + k.hamtaSaldo());
55         }
56
57         System.out.println(("Räntesats:    " + Konto.avlasRantesats()));
58     }
59 }

```

Lab4Uppg02.java

2.2.2 Konto.java

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #4    Uppgift 2
8   */
9
10 /**
11  * Klass 'Konto' representerar ett bankkonto.
12  */
13 public class Konto
14 {
15     /* Fall tillbaka på dessa som defaults. */
16     private static final String DEFAULT_NAME = "okänt";
17
18     /* Alla konton har samma räntesats (ränta per år) */
19     private static double rantesats;
20
21     private double saldo;
22     private int    kundnr;
23     private String kundnamn;
24
25     /**
26      * Konstruktör för ett bankkonto
27      * @param saldo      kontots saldo
28      * @param kundnr     kundens nummer
29      * @param kundnamn   kundens namn
30      */
31     public Konto(double saldo, int kundnr, String kundnamn)
32     {
33         this.saldo = saldo;
34         this.kundnr = kundnr;
35         this.kundnamn = kundnamn;
36     }
37
38     /**
39      * Ökar saldot med värde -- sätt in pengar.
40      * @param summa      saldot ökas med denna mängd
41      */
42     public void insattning(double summa)

```

```

43     {
44         if (summa > 0)
45             saldo += summa;
46     }
47
48     /**
49      * Minskar saldot med värde -- ta ut pengar.
50      * @param summa      saldot minskas med detta värde
51      */
52     public void uttag(double summa)
53     {
54         if (summa > 0)
55             if (summa <= saldo)
56                 saldo -= summa;
57     }
58
59     /**
60      * Returnerar kontots saldo.
61      * @return      kontots saldo
62      */
63     public double hamtaSaldo()
64     {
65         return saldo;
66     }
67
68     /**
69      * Returnerar kundens namn om det är definierat, annars 'DEFAULT_NAME'.
70      * @return      kundens namn
71      */
72     public String hamtaKundnamn()
73     {
74         if (kundnamn != null)
75             return kundnamn;
76         else
77             return DEFAULT_NAME;
78     }
79
80     /**
81      * Returnerar den gemensamma räntesatsen.
82      * @return      räntesatsen
83      */
84     public static double avlasRantesats()
85     {
86         return rantesats;
87     }
88 }

```

Konto.java

2.3 Skärmdump

Se Figur 2 för skärmdump på körning av koden i Sektion 2.2.1 och Sektion 2.2.2.

```
Terminal - spock@ProBookII: ~/lab4-bin.symlink
spock@ProBookII:~/lab4-bin.symlink$ java Lab4Uppg02

kundnamn: Gibson Pälsmann
saldo: 3500000.0
kundnamn: Foobar Bar
saldo: 100.0
Räntesats: 0.0

kundnamn: Gibson Pälsmann
saldo: 3500002.5
kundnamn: Foobar Bar
saldo: 4300.0
Räntesats: 0.0

kundnamn: Gibson Pälsmann
saldo: 2500001.5
kundnamn: Foobar Bar
saldo: 4300.0
Räntesats: 0.0
spock@ProBookII:~/lab4-bin.symlink$
```

Figur 2: Körning av koden till Uppgift 2

3 Uppgift 3

3.1 Instruktioner

I denna uppgift ska du använda dig av klassen `Person` som du skrev i laboration 3, **utan** att göra några ändringar i den. Skriv en till klass `Student` som ärver klassen `Person` (`Student` blir alltså en subklass till `Person`). I klassen `Student` ska du ha dessa instansvariabler:

```
kurs (String)
betyg (char)
skola (String)
```

I klassen `Student` ska du ha följande metoder:

- `ändraKurs`, ändrar kurs enligt parameterns värde.
- `ändraBetyg`, ändrar betyg enligt parameterns värde.
- `ändraSkola`, ändrar skola enligt parameterns värde.
- `hämtaKurs`, returnerar kurs.
- `hämtaBetyg`, returnerar betyg.
- `hämtaSkola`, returnerar skola.
- `toString`, en metod som i en snyggt formaterad sträng returnerar ett `Student`-objekts **samtliga** data, även indirekt "ärvda" data från klassen `Person`. Metoden anropas genom att skriva objektets (referensens) namn.

Och naturligtvis ska du ha med en konstruktor, där du ska initiera alla instansvariabler inkl de som ligger i superklassen `Person`.

Skriv slutligen en testklass där du skapar ett `Person`-objekt och ett `Student`-objekt. Testa först alla metoder som går att anropa på `Person`-objektet och testa sedan alla metoder som går att anropa på `Student`-objektet.

3.2 Källkod

3.2.1 Lab4Uppg03.java

```
1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #4    Uppgift 3
8  */
9
10 /* Eclipse-projektet för Labb #2 länkas in i det aktuella projektet så att
11  * paketet 'main' och klassen 'Person' blir tillgängliga. */
12 import main.Person;
13
14 /**
15  * Klass 'Lab4Uppg03' testar klasserna 'Student' och 'Person'.
16  */
17 public class Lab4Uppg03
18 {
19     /* Plattformsspecifik nyradsmarkör. */
20     private static String NEWLINE = System.getProperty("line.separator");
21 }
```

```

22     private static Person person;
23     private static Student student;
24
25     public static void main(String[] args)
26     {
27         init();
28         testaPerson();
29         testaStudent();
30         System.exit(0);
31     }
32
33     /**
34      * Skapar ett nytt 'Person'-objekt och ett 'Student'-objekt.
35      */
36     private static void init()
37     {
38         person = new Person("Rutherford B. Hayes",      /* Namn          */
39                             221004000,                  /* Personnummer  */
40                             "Ohio State U.S.",          /* Adress        */
41                             71);                        /* Ålder         */
42
43         student = new Student("John Quincy Adams",      /* Namn          */
44                               480711000,                /* Personnummer  */
45                               "Massachusetts U.S.",     /* Adress        */
46                               81,                       /* Ålder         */
47                               "Harvard",                 /* Skola         */
48                               null,                      /* Kurs (ej anmäld) */
49                               'B');                      /* Betyg         */
50     }
51
52     /**
53      * Testar alla metoder hos 'Person'-objektet.
54      */
55     private static void testaPerson()
56     {
57         /* Skriv ut initialt läge. */
58         prompt("Initial data för 'person':");
59         prompt(person.toString());
60
61         /* Anropa alla metoder .. */
62         person.byterAdress("New York City");
63         person.byterNamn("Theodore Roosevelt");
64         person.fyllerAr();
65
66         /* Skriv ut uppdaterat läge. */
67         prompt(NEWLINE + "Uppdaterad data för 'person':");
68         prompt(person.toString());
69     }
70
71     /**
72      * Testar alla metoder hos 'Student'-objektet.
73      */
74     private static void testaStudent()
75     {
76         /* Skriv ut initialt läge. */
77         prompt(NEWLINE + NEWLINE + "Initial data för 'student':");
78         prompt(student.toString());
79
80         /* Anropa alla metoder .. */
81         student.byterAdress("Kansas City");

```

```

82     student.byterNamn("John Lithgow");
83     student.fyllerAr();
84     student.andraSkola("Film School");
85     student.andraKurs("Amazing Acting 101");
86     student.andraBetyg('A');
87
88     /* Skriv ut uppdaterat läge. */
89     prompt(NEWLINE + "Uppdaterad data för 'student':");
90     prompt(student.toString());
91 }
92
93 /**
94  * "Wrapper" runt System.out.println() för mindre skrivande.
95  * @param s      textsträng att skriva ut
96  */
97 public static void prompt(String s)
98 {
99     System.out.println(s);
100 }
101 }

```

Lab4Uppg03.java

3.2.2 Student.java

```

1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #4    Uppgift 3
8  */
9
10 import main.Person;
11
12 /**
13  * Klass 'Student' representerar en student. Subklass till klassen 'Person'.
14  */
15 public class Student extends Person
16 {
17     private static final String INGEN_KURS = "Ej anmäld till någon kurs";
18     private static final String INGEN_SKOLA = "Ej registrerad på någon skola";
19     private static final char  INGET_BETYG = 'G';
20
21     private String skola;
22     private String kurs;
23     private char  betyg;
24
25     /**
26      * Konstruktör för en student.
27      * @param namn      studentens namn
28      * @param ID        studentens ID
29      * @param adress    studentens adress
30      * @param alder     studentens ålder
31      */
32     public Student(String namn, long ID, String adress, int alder, String skola,
33                   String kurs, char betyg)

```



```

34     {
35         super(namn, ID, adress, alder);
36
37         andraSkola(skola);
38         andraKurs(kurs);
39         andraBetyg(betyg);
40     }
41
42     /**
43      * Ändrar kurs enligt parameterens värde.
44      * @param kurs      ny kurs
45      */
46     public void andraKurs(String kurs)
47     {
48         if (kurs != null)
49             this.kurs = kurs;
50         else
51             this.kurs = INGEN_KURS;
52     }
53
54     /**
55      * Ändra betyg enligt parameterens värde.
56      * @param betyg      nytt betyg
57      */
58     public void andraBetyg(char betyg)
59     {
60         betyg = Character.toUpperCase(betyg);
61
62         switch (betyg) {
63             case 'A':
64             case 'B':
65             case 'C':
66             case 'D':
67             case 'E':
68             case 'F':
69                 this.betyg = betyg;
70                 break;
71             default:
72                 this.betyg = INGET_BETYG;
73         }
74     }
75
76     /**
77      * Ändra skola enligt parameterens värde.
78      * @param skola      ny skola
79      */
80     public void andraSkola(String skola)
81     {
82         if (skola != null)
83             this.skola = skola;
84         else
85             this.skola = INGEN_SKOLA;
86     }
87
88     /**
89      * Returnerar kurs studenten är anmäld till.
90      * @return      studentens kurs
91      */
92     public String hamtaKurs()
93     {

```

```

94         if (kurs != null)
95             return kurs;
96         else
97             return INGEN_KURS;
98     }
99
100    /**
101     * Returnerar studentens betyg.
102     * @return      studentens betyg
103     */
104    public char hamtaBetyg()
105    {
106        return betyg;
107    }
108
109    /**
110     * Returnerar studentens skola.
111     * @return      studentens skola
112     */
113    public String hamtaSkola()
114    {
115        if (skola != null)
116            return skola;
117        else
118            return INGEN_SKOLA;
119    }
120
121    /**
122     * Auto-genererad med Eclipse ("source" --> "Generate toString()...") och
123     * sedan modifierad för att inkludera hantering av plattformsspecifika
124     * nyradsmarkeringar.
125     */
126    @Override
127    public String toString()
128    {
129        /* Plattformsspecifik nyradsmarkör. */
130        String NEWLINE = System.getProperty("line.separator");
131        StringBuilder str = new StringBuilder();
132
133        /* Anropa först 'Person'-objektets 'toString()'-metod. */
134        str.append(super.toString());
135
136        /* Utöka sedan med 'Student'-objektets data. */
137        str.append("Skola:      " + skola + NEWLINE);
138        str.append("Kurs:         " + kurs + NEWLINE);
139        str.append("Betyg:        " + betyg + NEWLINE);
140
141        return str.toString();
142    }
143 }

```

Student.java

3.2.3 Person.java

```

1  /**
2  *  DV017A :: Grundläggande programmering i Java
3  *  860224 Jonas Sjöberg

```

```

4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #2    Uppgift 8
8  */
9
10 package main;
11
12 public class Person
13 {
14     private String namn;
15     private long personnummer;
16     private String adress;
17     private int alder;
18
19     /**
20      * Konstruktör för klassen 'Person' som representerar en person.
21      * @param namn      personens namn
22      * @param ID        personens personnummer
23      * @param adress    personens adress
24      * @param alder     personens ålder
25      */
26     public Person(String namn, long ID, String adress, int alder)
27     {
28         this.namn = namn;
29         this.personnummer = ID;
30         this.adress = adress;
31         this.alder = alder;
32     }
33
34     /**
35      * Byter namn på personen.
36      * @param namn      personens nya namn
37      */
38     public void byterNamn(String namn)
39     {
40         this.namn = namn;
41     }
42
43     /**
44      * Ändrar personens adress.
45      * @param adress    personens nya adress
46      */
47     public void byterAdress(String adress)
48     {
49         this.adress = adress;
50     }
51
52     /**
53      * Gör personen ett år äldre.
54      */
55     public void fyllerAr()
56     {
57         alder++;
58     }
59
60     /**
61      * Hämta personens namn
62      * @return          personens namn
63      */

```

```

64     public String hamtaNamn()
65     {
66         return namn;
67     }
68
69     /**
70      * Hämta personens personnummer
71      * @return      personens personnummer
72      */
73     public long hamtaPersnr()
74     {
75         return personnummer;
76     }
77
78     /**
79      * Hämta personens ålder
80      * @return      personens ålder
81      */
82     public int hamtaAlder()
83     {
84         return alder;
85     }
86
87     /**
88      * Hämta personens adress
89      * @return      personens adress
90      */
91     public String hamtaAdress()
92     {
93         return adress;
94     }
95
96     /**
97      * Returnerar objektets data i "human-readable" format
98      */
99     public String toString()
100    {
101        // Använder '\n' newlines trots att det inte är särskilt portabelt ..
102        return ("Namn:      " + this.namn
103              + "\nPersonnummer: " + this.personnummer
104              + "\nAdress:      " + this.adress
105              + "\nÅlder:      " + this.alder + "\n");
106    }
107 }

```

Person.java

3.3 Skärmdump

Se Figur 3 för skärmdump på körning av koden i Sektion 3.2.1, Sektion 3.2.2 och Sektion 3.2.3.

```
Terminal - spock@ProBookII: ~/lab4-bin.symlink
spock@ProBookII:~/lab4-bin.symlink$ java Lab4Uppg03
Initial data för 'person':

Namn:      Rutherford B. Hayes
Personnummer: 221004000
Adress:     Ohio State U.S.
Ålder:      71

Uppdaterad data för 'person':

Namn:      Theodore Roosevelt
Personnummer: 221004000
Adress:     New York City
Ålder:      72

Initial data för 'student':

Namn:      John Quincy Adams
Personnummer: 480711000
Adress:     Massachusetts U.S.
Ålder:      81
Skola:      Harvard
Kurs:       Ej anmäld till någon kurs
Betyg:      B

Uppdaterad data för 'student':

Namn:      John Lithgow
Personnummer: 480711000
Adress:     Kansas City
Ålder:      82
Skola:      Film School
Kurs:       Amazing Acting 101
Betyg:      A

spock@ProBookII:~/lab4-bin.symlink$
```

Figur 3: Körning av koden till Uppgift 3