

DV017A  
Java-programmering  
Laboration 3

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

Datum: 2015-08-06  
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

<b>1</b>	<b>Uppgift 1</b>	<b>4</b>
1.1	Instruktioner . . . . .	4
1.2	Svar . . . . .	4
<b>2</b>	<b>Uppgift 2</b>	<b>4</b>
2.1	Instruktioner . . . . .	4
2.2	Kommentar . . . . .	5
2.3	Källkod . . . . .	5
2.3.1	Lab3Uppg02.java . . . . .	5
2.3.2	Flygriktning.java . . . . .	6
2.3.3	FlygPlan.java . . . . .	6
2.4	Skärmdump . . . . .	9
<b>3</b>	<b>Uppgift 3</b>	<b>9</b>
3.1	Instruktioner . . . . .	9
3.2	Källkod . . . . .	11
3.2.1	Lab3Uppg03.java . . . . .	11
3.2.2	Cirkel.java . . . . .	12
3.3	Skärmdump . . . . .	14
<b>4</b>	<b>Uppgift 4</b>	<b>14</b>
4.1	Instruktioner . . . . .	14
4.2	Kommentar . . . . .	16
4.3	Källkod . . . . .	16
4.3.1	Lab3Uppg04.java . . . . .	16
4.3.2	Personbil.java . . . . .	17
4.3.3	Personbil.java . . . . .	19
4.4	Skärmdump . . . . .	23
<b>5</b>	<b>Uppgift 5</b>	<b>23</b>
5.1	Instruktioner . . . . .	23
5.2	Källkod . . . . .	26
5.2.1	Lab3Uppg05.java . . . . .	26
5.2.2	Bilagare.java . . . . .	27
5.3	Skärmdump . . . . .	29
<b>6</b>	<b>Uppgift 6</b>	<b>29</b>
6.1	Instruktioner . . . . .	29
6.2	Källkod . . . . .	30
6.2.1	Lab3Uppg06.java . . . . .	30
6.2.2	Berakning.java . . . . .	31
6.3	Skärmdump . . . . .	32

## Figurer

1	Körning av koden till Uppgift 2 . . . . .	10
2	Körning av koden i Sektion 3.2.1 tillhörandes Uppgift 3 . . . . .	15
3	Körning av koden till Uppgift 4 . . . . .	24

4	Körning av <b>PersonbilGUI</b> från Uppgift 4 . . . . .	25
5	Körning av <b>PersonbilGUI</b> från Uppgift 4 med andra bilar . . . . .	25
6	Körning av koden till Uppgift 5 . . . . .	29
7	Körning av koden till Uppgift 6 . . . . .	33

# 1 Uppgift 1

## 1.1 Instruktioner

Hur visar man att en referensvariabel inte refererar (pekar) på något objekt?  
Utgå från följande lilla kodsnitt:

```
//Här refererar (pekar) kalle till ett nytt Person-objekt
Person kalle = new Person ("Kalle Persson");
//Hur skriver du här om du vill att referensen kalle *inte*
//ska peka på något objekt ????????
```

## 1.2 Svar

Om referensen `kalle` inte ska peka på något objekt så måste referensvariabeln `kalle` sättas till att peka på *ingenting*, eller `null`. Referensvariabeln sätts till `null` genom att man skriver:

```
kalle = null;
```

# 2 Uppgift 2

## 2.1 Instruktioner

Skriv en klass `FlygPlan` som representerar ett flygplan. Klassens instansvariabler är:

```
höjd (int)
flygriktning (int)
hastighet (int)
modellbeteckning (String)
```

Lämplig datatyp står inom parentes. När det gäller instansvariabeln `flygriktning` så ska den ha värdet 0 om planet står stilla, värdet 1 vid nordlig riktning, 2 ostlig, 3 sydlig och 4 västlig.

Metoder ska finnas för följande operationer:

- ändra planets höjd
- returnera planets höjd
- ändra planets flygriktning
- returnera planets flygriktning
- ändra planets hastighet
- returnera planets hastighet
- ändra planets modellbeteckning
- returnera planets modellbeteckning
- skriv ut alla data om flygplanet, metoden ska ha returtypen `void`

Glöm ej konstruktorn.

Skriv sedan ett testprogram som testar klassens metoder. Skapa minst 2 stycken `FlygPlans`-objekt och testa metoderna på dessa objekt.

## 2.2 Kommentar

Jag har valt att använda datatypen `enum` för att lösa uppgiften. I instruktionerna efterfrågas numeriska värden för flygriktningen och min lösning uppfyller det kravet genom att metoden `getordinal()` från typen `enum` används för att hämta de implicita värden som tillskrivs de olika flygriktningarna i `Flygriktning` efter den ordningen de deklarerats. Min lösning för att *sätta* värdet hos en variabel av typen `Flygriktning` med en `int` är bristfällig, bättre vore att basera lösningen på de mer avancerade funktionerna som finns "inbyggda" i typen `enum`.

Användandet av `enum` baseras på information från The Java Tutorials – Enum Types.<sup>1</sup>

## 2.3 Källkod

### 2.3.1 Lab3Uppg02.java

---

```
1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #3    Uppgift 2
8  */
9
10 package Lab3Uppg02;
11
12 /**
13  * Testar klassen 'FlygPlan' och enum-typen 'Flygriktning'
14  */
15 public class Lab3Uppg02
16 {
17     private static FlygPlan planEtt, planTva;
18
19     public static void main(String[] args)
20     {
21         planEtt = new FlygPlan(900, Flygriktning.EAST, 3300, "JAS 39 Gripen");
22         planTva = new FlygPlan(500, Flygriktning.WEST, 470, "Mikoyan MiG-29");
23
24         skrivUtData("Data vid programstart:");
25         planEtt.printInfo();
26         planTva.printInfo();
27
28         planEtt.setAltitude(654321);
29         planEtt.setHeading(Flygriktning.NORTH);
30         planEtt.setSpeed(22000);
31         planTva.setAltitude(0);
32         planTva.setIdentifier("UFO!");
33
34         /* Använd metod för att ändra flygriktning med en int: */
35         planTva.setHeading(Flygriktning.fromOrdinal(0));
36
37         skrivUtData("Uppdaterad data:");
38         planEtt.printInfo();
39         planTva.printInfo();
40     }
41
42     /**
```

---

<sup>1</sup><https://docs.oracle.com/javase/tutorial/java/java00/enum.html>

```

43     * Skriver ut texten 'text' och gör en "understrykning" med '='-tecken.
44     * @param text      text att skriva ut och "stryka under"
45     */
46     private static void skrivUtData(String text)
47     {
48         if (text != null) {
49             String underline = "";
50
51             for (int i = 0; i < text.length(); i++) {
52                 underline = underline.concat("=");
53             }
54
55             System.out.println(text);
56             System.out.println(underline);
57             System.out.println("");
58         }
59     }
60 }

```

---

Lab3Uppg02.java

### 2.3.2 Flygriktning.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 2
8   */
9
10 package Lab3Uppg02;
11
12 public enum Flygriktning {
13     /* Ett numeriskt värde av Flygriktning kan hämtas med metoden 'getOrdinal()'
14     * STOPPED ger då värdet 0, NORTH ger 1, EAST ger 2, osv.. */
15     STOPPED, NORTH, EAST, SOUTH, WEST;
16
17     /* Metoden 'values()' returnerar en array som innehåller enumens värden
18     * i den ordning de deklarerades. (STOPPED = 0, NORTH = 1, osv.. ) */
19     private static Flygriktning[] allValues = values();
20
21     /* Metod för att kunna sätta en enum med en int.
22     * Förutsätter att ordningen för "enums" värden inte ändras. */
23     public static Flygriktning fromOrdinal(int n)
24     {
25         return allValues[n];
26     }
27 }

```

---

Flygriktning.java

### 2.3.3 FlygPlan.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java

```

```

3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #3    Uppgift 2
8  */
9
10 package Lab3Uppg02;
11
12 /**
13  * Klass 'FlygPlan' representerar ett flygplan.
14  */
15 public class FlygPlan
16 {
17     private int      altitude;
18     private Flygriktning heading;
19     private int      speed;
20     private String    identifier;
21
22     /**
23      * Konstruktor för ett Flygplan
24      * @param altitude    flygplanets höjd
25      * @param heading     flygplanets riktning
26      * @param speed       flygplanets hastighet
27      * @param identifier  flygplanets modellbeteckning
28      */
29     public FlygPlan(int altitude, Flygriktning heading, int speed,
30                     String identifier)
31     {
32         this.altitude = altitude;
33         this.heading = heading;
34         this.speed = speed;
35         this.identifier = identifier;
36
37         if (heading == Flygriktning.STOPPED) {
38             speed = 0;
39         }
40     }
41
42     /**
43      * Alternativ konstruktor hanterar att 'heading' sätt till ett värde av
44      * typen 'int'.
45      */
46     public FlygPlan(int altitude, int heading, int speed, String identifier)
47     {
48         this(altitude, Flygriktning.fromOrdinal(heading), speed, identifier);
49     }
50
51     /**
52      * Returnerar planets höjd.
53      * @return planets höjd
54      */
55     public int getAltitude()
56     {
57         return altitude;
58     }
59
60     /**
61      * Ändrar planets höjd.
62      * @param altitude ny höjd

```

```

63     */
64     public void setAltitude(int altitude)
65     {
66         this.altitude = altitude;
67     }
68
69     /**
70      * Returnerar planets flygriktning.
71      * @return planets flygriktning
72      */
73     public Flygriktning getHeading()
74     {
75         return heading;
76     }
77
78     /**
79      * Ändrar planets flygriktning.
80      * @param heading    ny flygriktning
81      */
82     public void setHeading(Flygriktning heading)
83     {
84         this.heading = heading;
85     }
86
87     /**
88      * Returnerar planets hastighet.
89      * @return planets hastighet
90      */
91     public int getSpeed()
92     {
93         return speed;
94     }
95
96     /**
97      * Ändrar planets hastighet.
98      * @param speed      ny hastighet
99      */
100    public void setSpeed(int speed)
101    {
102        if (speed > 0) { /* Anta att speed inte är en vektor. */
103            this.speed = speed;
104        }
105    }
106
107    /**
108     * Returnerar planets modellbeteckning.
109     * @return planets modellbeteckning
110     */
111    public String getIdentifier()
112    {
113        if (identifier != null) {
114            return identifier;
115        } else {
116            return "Planet saknar modellbeteckning!";
117        }
118    }
119
120    /**
121     * Ändrar planets modellbeteckning.
122     * @param identifier    ny modellbeteckning

```



```

123     */
124     public void setIdentifier(String identifier)
125     {
126         this.identifier = identifier;
127     }
128
129     /**
130      * Skriver ut all data om flygplanet.
131      */
132     public void printInfo()
133     {
134         System.out.println("Flygplansdata");
135         System.out.println("-----");
136         System.out.println("Höjd:           " + getAltitude());
137         System.out.println("Flygriktning:    " + getHeading().ordinal());
138         System.out.println("Hastighet:       " + getSpeed());
139         System.out.println("Modellbeteckning: " + getIdentifier());
140         System.out.println("");
141     }
142 }

```

---

FlygPlan.java

## 2.4 Skärmdump

Se Figur 1 för skärmdump på körning av koden i Sektion 2.3.1.

## 3 Uppgift 3

### 3.1 Instruktioner

Skriv en klass Cirkel som representerar en cirkel. I klassen ska följande två instansvariabler ingå:

radie (int), färg (String)

I parentes står lämplig datatyp.

I klassen ska finnas en metod för var och en av följande operationer:

- ändra färgen på cirkeln
- returnera cirkelns färg
- ändra cirkelns radie
- returnera cirkelns radie
- beräkna och returnera cirkelns omkrets
- beräkna och returnera cirkelns area

När man ska initiera ett cirkel-objekt ska man kunna välja på följande två alternativ:

- välja *\*både\** färg och radie på cirkeln
- välja endast radien på cirkeln, men cirkelns färg blir alltid gul

Detta betyder att du behöver två stycken konstruktorer (som överlagrar varandra) i din klass.

```
Terminal - spock@ProBookII: ~/lab3-bin.symlink
spock@ProBookII:~/lab3-bin.symlink$ java Lab3Uppg02.Lab3Uppg02
Data vid programstart:
=====
Flygplansdata
-----
Höjd: 900
Flygriktning: 2
Hastighet: 3300
Modellbeteckning: JAS 39 Gripen

Flygplansdata
-----
Höjd: 500
Flygriktning: 4
Hastighet: 470
Modellbeteckning: Mikoyan MiG-29

Uppdaterad data:
=====
Flygplansdata
-----
Höjd: 654321
Flygriktning: 1
Hastighet: 22000
Modellbeteckning: JAS 39 Gripen

Flygplansdata
-----
Höjd: 0
Flygriktning: 0
Hastighet: 470
Modellbeteckning: UF0!

spock@ProBookII:~/lab3-bin.symlink$
```

Figur 1: Körning av koden till Uppgift 2

Slutligen skriv ett testprogram som testar så att klassens metoder fungerar som de ska. Skapa åtminstone två stycken cirkel-objekt, som använder sig av olika konstruktörer. Alla decimaltals-utskrifter på skärmen ska avrundas till \*en\* decimal.

Tips: För att vid utskrift avrunda ett decimaltal använd klassen DecimalFormat, finns exempel i boken hur du använder denna. Den ligger i paketet java.text, du måste alltså importera denna klass i ditt testprogram.

## 3.2 Källkod

### 3.2.1 Lab3Uppg03.java

---

```
1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 3
8   */
9
10 package Lab3Uppg03;
11
12 import java.text.DecimalFormat;
13
14 /**
15  * Testar metoder hos klassen 'Cirkel'.
16  */
17 public class Lab3Uppg03
18 {
19     private static Cirkel c1, c2;
20
21     public static void main(String[] args)
22     {
23         int radius = 13;
24         String color = "röd";
25
26         c1 = new Cirkel(radius);
27         c2 = new Cirkel(color, radius);
28
29         showObjectState();
30
31         c1.setColor("svart");
32         c1.setRadius(1337);
33
34         c2.setColor("vit");
35         c2.setRadius(-1);
36
37         showObjectState();
38     }
39
40     /**
41      * Skriver ut all data för cirkelarna 'c1' och 'c2'.
42      */
43     private static void showObjectState()
44     {
45         promptInfo("c1", "radien", c1.getRadius());
```

```

46     promptInfo("c1", "färgen", c1.getColor());
47     promptInfo("c1", "omkretsen", c1.getCircumference());
48     promptInfo("c1", "arean", c1.getArea());
49     System.out.println("");
50     promptInfo("c2", "radien", c2.getRadius());
51     promptInfo("c2", "färgen", c2.getColor());
52     promptInfo("c2", "omkretsen", c2.getCircumference());
53     promptInfo("c2", "arean", c2.getArea());
54     System.out.println("");
55 }
56
57 /**
58  * Skriver ut värdet 'v' för attributet 'a' hos cirkeln 'c'
59  * @param c      cirkel
60  * @param a      attribut
61  * @param v      värde
62  */
63 private static void promptInfo(String c, String a, int v)
64 {
65     System.out.println("Cirkeln \"" + c + "\" har " + a + ": " + v);
66 }
67
68 private static void promptInfo(String c, String a, String v)
69 {
70     System.out.println("Cirkeln \"" + c + "\" har " + a + ": " + v);
71 }
72
73 private static void promptInfo(String c, String a, double v)
74 {
75     System.out.println("Cirkeln \"" + c + "\" har " + a + ": " + round(v));
76 }
77
78 /**
79  * Avrunda decimaltal till en decimal med DecimalFormat
80  * @param n      tal att avrunda
81  * @return       talet 'n' som textsträng, avrundat till en decimal
82  */
83 private static String round(double n)
84 {
85     DecimalFormat df = new DecimalFormat("#.##");
86
87     String formatted = df.format(n);
88     return formatted;
89 }
90 }

```

---

Lab3Uppg03.java

### 3.2.2 Cirkel.java

---

```

1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #3    Uppgift 3
8  */

```

```

9
10 package Lab3Uppg03;
11
12 /**
13  * Klass 'Cirkel' representerar en cirkel.
14  */
15 public class Cirkel
16 {
17     private static final String DEFAULT_COLOR = "gul";
18
19     private int    radius;
20     private String color;
21
22     /**
23      * Konstruktor för klassen Cirkel.
24      * @param color      cirkelns färg
25      * @param radius      cirkelns radius
26      */
27     public Cirkel(String color, int radius)
28     {
29         this.color = color;
30         this.radius = radius;
31     }
32
33     /**
34      * Konstruktor för klassen Cirkel.
35      * Skapar en cirkel med färgen 'DEFAULT_COLOR'.
36      * @param radius      cirkelns radie
37      */
38     public Cirkel(int radius)
39     {
40         this(DEFAULT_COLOR, radius);
41     }
42
43     /**
44      * Returnerar cirkelns radie.
45      * @return      cirkelns radie
46      */
47     public int getRadius()
48     {
49         return radius;
50     }
51
52     /**
53      * Ändrar cirkelns radie.
54      * @param radius      ny radie
55      */
56     public void setRadius(int radius)
57     {
58         if (radius > 0) {
59             this.radius = radius;
60         } else {
61             System.out.println("Radien måste vara större än noll!");
62         }
63     }
64
65     /**
66      * Returnerar cirkelns färg.
67      * @return      cirkelns färg
68      */

```

```

69     public String getColor()
70     {
71         return color;
72     }
73
74     /**
75      * Ändrar cirkelns färg.
76      * @param color    ny färg
77      */
78     public void setColor(String color)
79     {
80         this.color = color;
81     }
82
83     /**
84      * Beräknar och returnerar cirkelns omkrets.
85      * @return    cirkelns omkrets
86      */
87     public double getCircumference()
88     {
89         double omkrets = 2 * Math.PI * getRadius();
90         return omkrets;
91     }
92
93     /**
94      * Beräknar och returnerar cirkelns area.
95      * @return    cirkelns area
96      */
97     public double getArea()
98     {
99         /* Den lokal variabeln 'radius' håller värdet från 'getRadius()'
100          * Användningen av en lokal variabel för mellanlagring motiveras med
101          * ökad läslighet av kod, snarare än "optimering" (färre metodanrop).
102          */
103         int radius = getRadius();
104         double area = Math.PI * radius * radius;
105         return area;
106     }
107 }

```

---

Cirkel.java

### 3.3 Skärmdump

Se Figur 2 för skärmdump på körning av koden i Sektion 3.2.1.

## 4 Uppgift 4

### 4.1 Instruktioner

Skriv en klass Personbil som representerar en personbil. Klassen ska innehålla följande 4 instansvariabler. Välj själv passande datatyp:

bilmodell, årsmodell, registreringsnr och bilfärg

Det ska finnas metoder för var och en av följande operationer:

- returnera bilens modell

```
Terminal - spock@ProBookII: ~/lab3-bin.symlink
spock@ProBookII:~/lab3-bin.symlink$ java Lab3Uppg03.Lab3Uppg03
Cirkeln "c1" har radien: 13
Cirkeln "c1" har färgen: gul
Cirkeln "c1" har omkretsen: 81.7
Cirkeln "c1" har arean: 530.9

Cirkeln "c2" har radien: 13
Cirkeln "c2" har färgen: röd
Cirkeln "c2" har omkretsen: 81.7
Cirkeln "c2" har arean: 530.9

Radien måste vara större än noll!
Cirkeln "c1" har radien: 1337
Cirkeln "c1" har färgen: svart
Cirkeln "c1" har omkretsen: 8400.6
Cirkeln "c1" har arean: 5615813.6

Cirkeln "c2" har radien: 13
Cirkeln "c2" har färgen: vit
Cirkeln "c2" har omkretsen: 81.7
Cirkeln "c2" har arean: 530.9

spock@ProBookII:~/lab3-bin.symlink$
1  public static void main(String[] args)
2      {
3          // Skapa två plan
4          PlanEtt planEtt = new PlanEtt(100, Flygriktning.EAST, 3300, "JAS 39 Gripen");
5          PlanTva planTva = new PlanTva(500, Flygriktning.WEST, 470, "Mikoyan MiG-29");
6
7          // Skriv ut information om planerna
8          planEtt.printInfo();
9          planTva.printInfo();
10
11          // Använd setterna för att ändra flygriktning med enbart
12          planEtt.setHeading(Flygriktning.NORTH);
13          planTva.setHeading(Flygriktning.fromOrdinal(0));
14
15          // Skriv ut uppdaterad data
16          skrivUtData("Uppdaterad data:");
17          planEtt.printInfo();
18          planTva.printInfo();
19      }
20  }
```

Figur 2: Körning av koden i Sektion 3.2.1 tillhörandes Uppgift 3

- returnera bilens årsmodell
- returnera bilens regnr
- ändra bilens färg
- returnera bilens färg
- skriva ut bilens samtliga data (metodens returtyp ska vara void)

Skriv sedan ett testprogram där du skapar 2 st personbils-objekt och sedan testat samtliga metoder. Objektens instansvariabler ska via konstruktorn initieras med följande värden:

Bil1 : bilmodell Saab, årsmodell -90, regnr CCC222 och färg röd.

Bil2 : bilmodell Volvo, årsmodell -99, regnr ABC988 och färg svart.

## 4.2 Kommentar

Programmet PersonbilGUI skapades med verktyget WindowBuilder i Eclipse Mars. Programmet visar hur typen Color kan användas i ett GUI. Metoden toString() används för att skriva ut färgen. Den visas då i RGB-format. För att programmet ska skriva ut färgen som ett enkelt ord kan en möjlig lösning nyttja en enum som "mappade" objekt av typen Color med textsträngar av typen String. Typen kunde representera en färg som sedan kan returneras i "fler former", t.ex. Color och textsträng. Referenser för Color och Swing:

Color – The Java Language Specification, Java SE 7 Edition <sup>2</sup>

Swing Tutorial – The Java Tutorials <sup>3</sup>

## 4.3 Källkod

### 4.3.1 Lab3Uppg04.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 4
8   */
9
10 package Lab3Uppg04;
11
12 import java.awt.Color;
13
14 /**
15  * Testar klassen 'Personbil'
16  */
17 public class Lab3Uppg04
18 {
19     private static Personbil bil1, bil2;
20
21     public static void main(String[] args)
22     {
23         bil1 = new Personbil("Saab", 1990, "CCC222", Color.RED);
24         bil2 = new Personbil("Volvo", 1999, "ABC988 ", Color.BLACK);
25     }

```

<sup>2</sup><http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html>

<sup>3</sup><http://docs.oracle.com/javase/tutorial/uiswing/>



```

26         bil1.printAllData();
27         bil2.printAllData();
28     }
29 }

```

---

Lab3Uppg04.java

### 4.3.2 Personbil.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 4
8   */
9
10 package Lab3Uppg04;
11
12 import java.awt.Color;
13 import java.util.Calendar;
14
15 /**
16  * Klass 'Personbil' representerar en personbil.
17  */
18 public class Personbil
19 {
20     private String model;
21     private int    yearManufactured;
22     private String registrationNumber;
23     private Color  color;
24
25     /**
26      * Konstruktör för klassen 'Personbil' som representerar en personbil.
27      * @param model
28      * @param yearMf
29      * @param regNmbr
30      * @param color
31      */
32     public Personbil(String model, int yearMf, String regNmbr, Color color)
33     {
34         this.model = model;
35         yearManufactured = yearMf;
36         registrationNumber = regNmbr;
37         this.color = color;
38     }
39
40     /**
41      * Returnerar bilens modell.
42      * @return    bilens modell
43      */
44     public String getModel()
45     {
46         return model;
47     }
48
49     /**

```

```

50     * Ändrar bilens årsmodell.
51     * @param registrationNumber    nytt årsmodell
52     */
53     public void setModel(String model)
54     {
55         if (model == null)
56             return;
57
58         this.model = model;
59     }
60
61     /**
62     * Returnerar bilens tillverkningsår.
63     * @return    bilens tillverkningsår
64     */
65     public int getYearManufactured()
66     {
67         return yearManufactured;
68     }
69
70     /**
71     * Ändrar bilens tillverklingsnår.
72     * @param registrationNumber    nytt tillverklingsnår
73     */
74     public void setYearManufactured(int yearManufactured)
75     {
76         int year = Calendar.getInstance().get(Calendar.YEAR);
77
78         if (yearManufactured > 0 && yearManufactured <= year)
79             this.yearManufactured = yearManufactured;
80     }
81
82     /**
83     * Returnerar bilens registreringsnummer.
84     * @return    bilens registreringsnummer
85     */
86     public String getRegistrationNumber()
87     {
88         return registrationNumber;
89     }
90
91     /**
92     * Ändrar bilens registreringsnummer.
93     * @param registrationNumber    nytt registreringsnummer
94     */
95     public void setRegistrationNumber(String registrationNumber)
96     {
97         if (registrationNumber == null)
98             return;
99
100         this.registrationNumber = registrationNumber;
101     }
102
103     /**
104     * Returnerar bilens färg.
105     * @return    bilens färg
106     */
107     public Color getColor()
108     {
109         return color;

```

```

110     }
111
112     /**
113      * Ändrar bilens färg.
114      * @param color    ny färg
115      */
116     public void setColor(Color color)
117     {
118         if (color == null)
119             return;
120
121         this.color = color;
122     }
123
124     /**
125      * Skriver ut bilens samtliga data.
126      */
127     public void printAllData()
128     {
129         System.out.println("Modell:           " + getModel());
130         System.out.println("Tillverklingsår: " + getYearManufactured());
131         System.out.println("Registreringsnr: " + getRegistrationNumber());
132         System.out.println("Färg:           " + getColor().toString());
133         System.out.println("-----");
134     }
135 }

```

---

Personbil.java

### 4.3.3 Personbil.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 4
8   */
9
10 package Lab3Uppg04;
11
12 import java.awt.Color;
13 import java.awt.EventQueue;
14 import java.awt.Font;
15 import java.awt.GridBagConstraints;
16 import java.awt.GridBagLayout;
17 import java.awt.Insets;
18
19 import javax.swing.JFrame;
20 import javax.swing.JPanel;
21 import javax.swing.JTextArea;
22 import javax.swing.border.EmptyBorder;
23 import java.awt.FlowLayout;
24
25 /**
26  * Vidareutvecklat test av klassen 'Personbil'
27  */

```

```

28 public class PersonbilGUI extends JFrame
29 {
30     /* Används av "serialization runtime" .. */
31     private static final long serialVersionUID = 1L;
32
33     /* Variabler relaterat till utseende */
34     static final Color backgroundColor = Color.RED;
35     static final Font  stdFont      = new Font("monospaced", 0, 12);
36     static final Font  stdFontBold  = new Font("monospaced", 1, 12);
37
38     /* Bakomliggande "ritytan" */
39     private JPanel contentPane;
40
41     private static Personbil bil1;
42     private static Personbil bil2;
43
44     public static void main(String[] args)
45     {
46         EventQueue.invokeLater(new Runnable() {
47             public void run()
48             {
49                 try {
50                     PersonbilGUI frame = new PersonbilGUI();
51                     frame.setVisible(true);
52                 } catch (Exception e) {
53                     e.printStackTrace();
54                 }
55             }
56         });
57     }
58
59     /**
60      * Anropar metoder för att initiera och rita GUI:t
61      */
62     public PersonbilGUI()
63     {
64         initialize();
65         createFrame();
66
67         /* Bil #1 "Data" textfält */
68         createCar1Text();
69
70         /* Bil #1 Färgat område */
71         createCar1ColoredArea();
72
73         /* Bil #2 "Data" textfält */
74         createCar2Text();
75
76         /* Bil #2 Färgat område */
77         createCar2ColoredArea();
78     }
79
80     /**
81      * Skapar objekt och konfigurerar JFrame
82      */
83     private void initialize()
84     {
85         bil1 = new Personbil("Saab", 1990, "CCC222", Color.RED);
86         bil2 = new Personbil("Volvo", 1999, "ABC988 ", Color.BLACK);
87         //bil1 = new Personbil("Cadillac", 1903, "ABC123", Color.MAGENTA);

```

```

88     //bil2 = new Personbil("Austin", 1906, "DEF456", Color.GREEN);
89
90     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
91 }
92
93 /**
94  * Skapar en "frame" som delar upp ritytan i en "grid"-layout
95  */
96 private void createFrame()
97 {
98     setBounds(100, 100, 450, 300);
99     contentPane = new JPanel();
100    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
101    contentPane.setBackground(Color.WHITE);
102    setContentPane(contentPane);
103
104    GridBagLayout gbl_contentPane = new GridBagLayout();
105    gbl_contentPane.columnWidths = new int[] { 0 };
106    gbl_contentPane.rowHeights = new int[] { 0, 0, 0, 0, 0 };
107    gbl_contentPane.columnWeights = new double[] { 1.0 };
108    gbl_contentPane.rowWeights = new double[] { 1.0, 0.0, 1.0, 0.0,
109        Double.MIN_VALUE };
110    contentPane.setLayout(gbl_contentPane);
111 }
112
113 /**
114  * Ritar färgband som representerar färgen hos Bil #1
115  */
116 private void createCar1ColoredArea()
117 {
118     JPanel panelBil1Color = new JPanel();
119     FlowLayout flowLayout = (FlowLayout) panelBil1Color.getLayout();
120     flowLayout.setHgap(0);
121     flowLayout.setVgap(15);
122     panelBil1Color.setBackground(bil1.getColor());
123
124     GridBagConstraints gbc_panelBil1Color = new GridBagConstraints();
125     gbc_panelBil1Color.insets = new Insets(0, 0, 5, 5);
126     gbc_panelBil1Color.fill = GridBagConstraints.BOTH;
127     gbc_panelBil1Color.gridx = 0;
128     gbc_panelBil1Color.gridy = 1;
129
130     contentPane.add(panelBil1Color, gbc_panelBil1Color);
131 }
132
133 /**
134  * Ritar färgband som representerar färgen hos Bil #2
135  */
136 private void createCar2ColoredArea()
137 {
138     JPanel panelBil2Color = new JPanel();
139     FlowLayout flowLayout = (FlowLayout) panelBil2Color.getLayout();
140     flowLayout.setHgap(0);
141     flowLayout.setVgap(15);
142     panelBil2Color.setBackground(bil2.getColor());
143
144     GridBagConstraints gbc_panelBil2Color = new GridBagConstraints();
145     gbc_panelBil2Color.insets = new Insets(0, 0, 0, 5);
146     gbc_panelBil2Color.fill = GridBagConstraints.BOTH;
147     gbc_panelBil2Color.gridx = 0;

```

```

148         gbc_panelBil2Color.gridy = 3;
149
150         contentPane.add(panelBil2Color, gbc_panelBil2Color);
151     }
152
153     /**
154      * Ritar text med data för Bil #1
155      */
156     private void createCar1Text()
157     {
158         JPanel panelBil1Data = new JPanel();
159         panelBil1Data.setBackground(Color.WHITE);
160
161         GridBagConstraints gbc_panelBil1Data = new GridBagConstraints();
162         gbc_panelBil1Data.anchor = GridBagConstraints.BASELINE;
163         gbc_panelBil1Data.insets = new Insets(0, 0, 5, 5);
164         gbc_panelBil1Data.fill = GridBagConstraints.HORIZONTAL;
165         gbc_panelBil1Data.gridx = 0;
166         gbc_panelBil1Data.gridy = 0;
167         contentPane.add(panelBil1Data, gbc_panelBil1Data);
168
169         JTextArea textBil1 = new JTextArea();
170         textBil1.setFont(stdFont);
171         textBil1.setText("Bil #1 Data");
172         addLine(textBil1, "Bilmodell", bil1.getModel());
173         addLine(textBil1, "Årsmodell",
174             String.valueOf(bil1.getYearManufactured()));
175         addLine(textBil1, "Registreringsnummer",
176             String.valueOf(bil1.getRegistrationNumber()));
177         addLine(textBil1, "Färg", bil1.getColor().toString());
178
179         panelBil1Data.add(textBil1);
180     }
181
182     /**
183      * Ritar text med data för Bil #2
184      */
185     private void createCar2Text()
186     {
187         JPanel panelBil2Data = new JPanel();
188         panelBil2Data.setBackground(Color.WHITE);
189
190         GridBagConstraints gbc_panelBil2Data = new GridBagConstraints();
191         gbc_panelBil2Data.anchor = GridBagConstraints.BASELINE;
192         gbc_panelBil2Data.insets = new Insets(0, 0, 5, 5);
193         gbc_panelBil2Data.fill = GridBagConstraints.HORIZONTAL;
194         gbc_panelBil2Data.gridx = 0;
195         gbc_panelBil2Data.gridy = 2;
196         contentPane.add(panelBil2Data, gbc_panelBil2Data);
197
198         JTextArea textBil2 = new JTextArea();
199         textBil2.setFont(stdFont);
200         textBil2.setText("Bil #2 Data");
201         addLine(textBil2, "Bilmodell", bil2.getModel());
202         addLine(textBil2, "Årsmodell",
203             String.valueOf(bil2.getYearManufactured()));
204         addLine(textBil2, "Registreringsnummer",
205             String.valueOf(bil2.getRegistrationNumber()));
206         addLine(textBil2, "Färg", bil2.getColor().toString());
207

```

```

208     panelBil2Data.add(textBil2);
209 }
210
211 /**
212  * Lägger till en rad text med två fält, str1 och str2, i JTextArea
213  * textArea.
214  * @param textArea    text läggs till här
215  * @param str1        textfält 1
216  * @param str2        textfält 2
217  */
218 private static void addLine(JTextArea textArea, String str1, String str2)
219 {
220     final String format = "%1$-20s : %2$-15s";
221
222     String line = String.format(format, str1, str2);
223     textArea.append("\n" + line);
224 }
225
226 }

```

---

PersonbilGUI.java

## 4.4 Skärmdump

Se Figur 3 för skärmdump på körning av koden i Sektion 4.3.1.  
 Figur 4 och Figur 5 visar körning av koden i Sektion 4.3.3.

## 5 Uppgift 5

### 5.1 Instruktioner

Skriv en klass Bilagare som representerar en bilägare. Klassen ska ha följande tre instansvariabler:

namn (String), adress (String), bil (Personbil)

Instansvariablernas typer står inom parentes. I denna uppgift ska du även använda dig av klassen Personbil som du skapade i uppgift 4. Eftersom en bilägare äger en bil så måste vi ha med instansvariabeln bil, som är en referens till ett objekt av klassen Personbil.

I klassen Bilagare ska det finnas metoder för var och en av följande operationer:

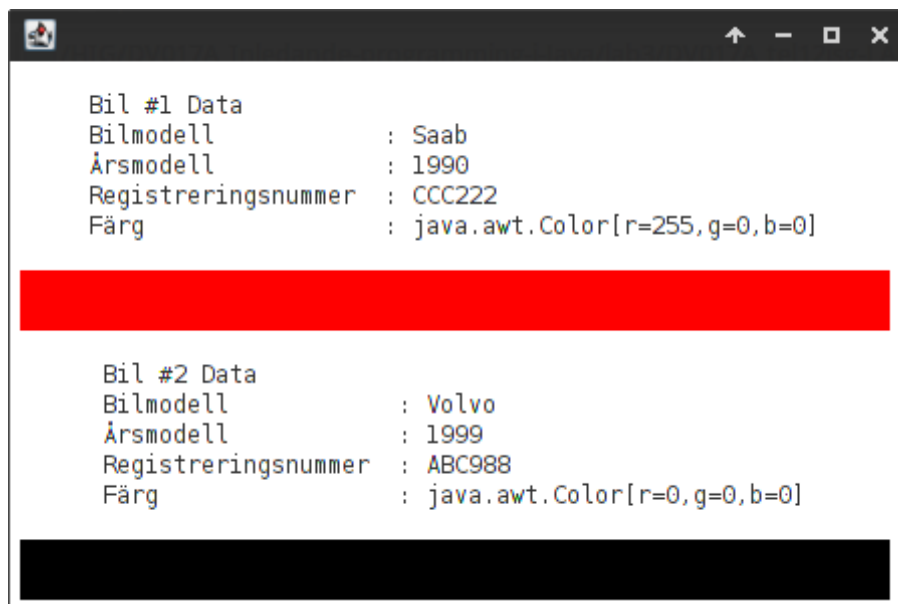
- returnera bilägarers namn
- returnera bilägarers adress
- bilägaren säljer sin bil (sätt instansvariabeln bil att ha värdet null)
- bilägaren köper ny bil (skicka som parameter med referensen till detta bil-objekt)
- skriv ut alla data om bilägarers bil, om han inte äger någon bil så ska valfritt meddelande (t.ex "Äger ingen bil för närvarande") istället skrivas ut. Metoden ska ha returtypen void.
- skriv ut bilägarers namn och adress.

Det ska naturligtvis finnas en konstruktor så att du kan initiera de tre instansvariablerna.

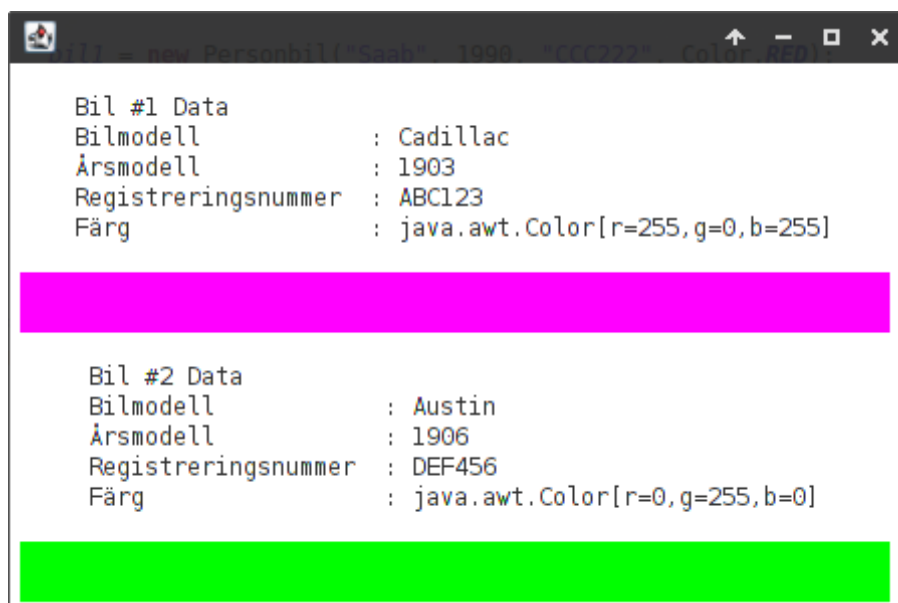
```
Terminal - spock@ProBookII: ~/lab3-bin.symlink
spock@ProBookII:~/lab3-bin.symlink$ java Lab3Uppg04.Lab3Uppg04
Modell:      Saab
Tillverklingsår: 1990
Registreringsnr: CCC222
Färg:      java.awt.Color[r=255,g=0,b=0]
-----
Modell:      Volvo
Tillverklingsår: 1999
Registreringsnr: ABC988
Färg:      java.awt.Color[r=0,g=0,b=0]
-----
spock@ProBookII:~/lab3-bin.symlink$ java Lab3Uppg02.Lab3Uppg02
-----
15 public class Lab3Uppg02
16 {
17     private static FlygPlan planEtt, planTva;
18
19     public static void main(String[] args)
20     {
21         planEtt = new FlygPlan(900, Flygriktning.EAST, 3300, "JAS 39 Gripen");
22         planTva = new FlygPlan(500, Flygriktning.WEST, 470, "Mikoyan MiG-29");
23
24         skrivUtData("Data vid programstart:");
25         planEtt.printInfo();
26         planTva.printInfo();
27
28         planEtt.setAltitude(654321);
29         planEtt.setHeading(Flygriktning.NORTH);
30         planEtt.setSpeed(22000);
31         planTva.setAltitude(0);
32         planTva.setIdentifier("UFO!");
33
34         // Använd metoden för att ändra flygriktning med en index
35         planTva.setHeading(Flygriktning.fromOrdinal(0));
36
37         skrivUtData("Uppdaterad data:");
38         planEtt.printInfo();
39         planTva.printInfo();
40     }
41 }
```

Figur 3: Körning av koden till Uppgift 4





Figur 4: Körning av PersonbilGUI från Uppgift 4



Figur 5: Körning av PersonbilGUI från Uppgift 4 med andra bilar

Skriv ett testprogram som utför följande:

- Skapa två Personbils-objekt. Initiera objekten med valfria värden.
- Skapa ett Bilägare-objekt. Initiera objektet med valfria värden, ägarens bil måste dock vara någon av de två personbils-objekten.
- Skriv ut bilägarens namn och adress.
- Skriv ut data om bilägarens bil.
- Bilägaren säljer sin bil.
- Skriv ut data om bilägarens bil.
- Bilägaren köper en bil. Låt honom köpa någon av de två skapade personbils-objekten.
- Skriv ut data om bilägarens bil.

## 5.2 Källkod

### 5.2.1 Lab3Uppg05.java

---

```
1 /**
2  * DV017A :: Grundläggande programmering i Java
3  * 860224 Jonas Sjöberg
4  * Högskolan i Gävle
5  * tel12jsg@student.hig.se
6  *
7  * Labb #3    Uppgift 5
8  */
9
10 package Lab3Uppg05;
11
12 import java.awt.Color;
13 import Lab3Uppg04.Personbil;
14
15 /**
16  * Testprogram för klassen 'Bilagare'.
17  */
18 public class Lab3Uppg05
19 {
20     private static Personbil bil1;
21     private static Personbil bil2;
22     private static Bilagare agare1;
23
24     public static void main(String[] args)
25     {
26         init();
27         run();
28     }
29
30     /**
31      * Initierar objekten.
32      */
33     private static void init()
34     {
35         bil1 = new Personbil("Ferrari 250 GTO", 1962, "1337GT", Color.RED);
36         bil2 = new Personbil("Bugatti Royale Kellner Coupe", 1931, "EWS13", Color.BLUE);
37         agare1 = new Bilagare("Bilägarkatten Gibson", "Vägenvägen 72", bil1);
38     }
39
40     /**
```

```

41     * Kör olika tester.
42     */
43     private static void run()
44     {
45         agare1.printName();
46         agare1.printAdress();
47
48         System.out.println("");
49         System.out.println("* Data om bilägarens bil:");
50         agare1.printCarData();
51
52         System.out.println("");
53         System.out.println("* Bilägaren säljer sin bil ..");
54         agare1.sellCar();
55
56         System.out.println("");
57         System.out.println("* Data om bilägarens bil:");
58         agare1.printCarData();
59
60         System.out.println("");
61         System.out.println("* Bilägaren köper en bil ..");
62         agare1.buyCar(bil2);
63         agare1.printCarData();
64
65     }
66 }

```

---

Lab3Uppg05.java

### 5.2.2 Bilagare.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 5
8   */
9
10 package Lab3Uppg05;
11
12 import Lab3Uppg04.Personbil;
13
14 /**
15  * Klass 'Bilagare' representerar en bilägare.
16  */
17 public class Bilagare
18 {
19     private static final String TXT_DOES_NOT_OWN_CAR = "Äger ingen bil för närvarande";
20     private static final String TXT_PREFIX_NAME      = "Bilägarens namn: ";
21     private static final String TXT_PREFIX_ADRESS    = "Bilägarens adress: ";
22     private String              name;
23     private String              adress;
24     private Personbil           bil;
25
26     /**
27      * Konstruktör för klassen 'Bilagare' som representerar en bilägare.

```

```

28      * @param name      bilägarens namn
29      * @param adress     bilägarens adress
30      * @param bil        bilägarens bil
31      */
32      public Bilagare(String name, String adress, Personbil bil)
33      {
34          this.name = name;
35          this.adress = adress;
36          this.bil = bil;
37      }
38
39      /**
40       * Skriver ut bilägarens namn.
41       */
42      public void printName()
43      {
44          System.out.println(TXT_PREFIX_NAME + name);
45      }
46
47      /**
48       * Skriver ut bilägarens adress.
49       */
50      public void printAdress()
51      {
52          System.out.println(TXT_PREFIX_ADRESS + adress);
53      }
54
55      /**
56       * Bilägaren säljer bilen.
57       */
58      public void sellCar()
59      {
60          bil = null;
61      }
62
63      /**
64       * Bilägaren köper en ny bil.
65       * @param bil    den bil som köps.
66       */
67      public void buyCar(Personbil bil)
68      {
69          this.bil = bil;
70      }
71
72      /**
73       * Skriver ut data om bilägarens bil.
74       */
75      public void printCarData()
76      {
77          if (bil == null)
78              System.out.println(TXT_DOES_NOT_OWN_CAR);
79          else
80              bil.printAllData();
81      }
82
83  }

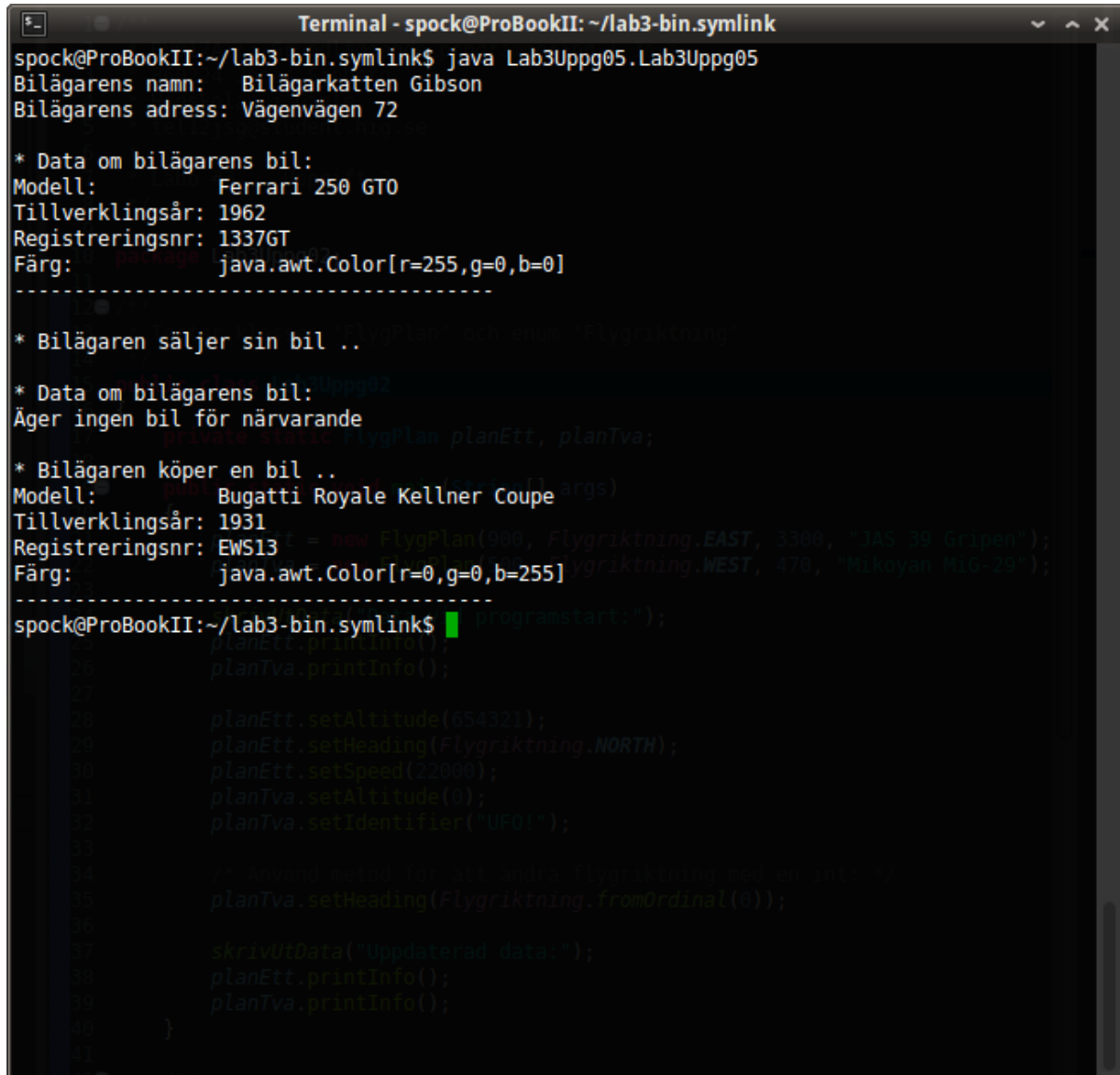
```

---

Bilagare.java

## 5.3 Skärmdump

Se Figur 6 för skärmdump på körning av koden i Sektion 5.2.1.



```
Terminal - spock@ProBookII: ~/lab3-bin.symlink
spock@ProBookII:~/lab3-bin.symlink$ java Lab3Uppg05.Lab3Uppg05
Bilägarens namn:  Bilägarkatten Gibson
Bilägarens adress: Vägenvägen 72

* Data om bilägarens bil:
Modell:      Ferrari 250 GTO
Tillverklingsår: 1962
Registreringsnr: 1337GT
Färg:      java.awt.Color[r=255,g=0,b=0]
-----

* Bilägaren säljer sin bil ..

* Data om bilägarens bil:
Äger ingen bil för närvarande

* Bilägaren köper en bil ..
Modell:      Bugatti Royale Kellner Coupe
Tillverklingsår: 1931
Registreringsnr: EWS13
Färg:      java.awt.Color[r=0,g=0,b=255]
-----

spock@ProBookII:~/lab3-bin.symlink$
14      programstart:");
15      planEtt.printlnInfo();
16      planTva.printlnInfo();
17
18      planEtt.setAltitude(654321);
19      planEtt.setHeading(Flygriktning.NORTH);
20      planEtt.setSpeed(22000);
21      planTva.setAltitude(0);
22      planTva.setIdentifier("UF0!");
23
24      // ändra hastighet för att ändra flygriktning med en info()
25      planTva.setHeading(Flygriktning.fromOrdinal(0));
26
27      skrivUtData("Uppdaterad data:");
28      planEtt.printlnInfo();
29      planTva.printlnInfo();
30  }
31
```

Figur 6: Körning av koden till Uppgift 5

## 6 Uppgift 6

### 6.1 Instruktioner

Skriv en klass som heter Berakning. Klassen ska användas för olika beräkningar med tre stycken valfria heltal. Klassen innehåller inga instansvariabler.

De operationer (beräkningar) som klassen ska kunna utföra är följande:

- Beräkna och returnera summan av tre stycken heltal. Talen kommer in via parametrar.

- Beräkna och returnera produkten av tre stycken heltal. Talen kommer in via parametrar.
- Beräkna och returnera det minsta av tre stycken heltal. Talen kommer in via parametrar.
- Beräkna och returnera det största av tre stycken heltal. Talen kommer in via parametrar.

Skriv sedan ett program som beräknar och skriver ut summan resp produkten av talen 4, 8 och 78. Och som sedan skriver ut det minsta och största talet av talen 100, 23 och 27. Du ska använda dig av metoderna i klassen Berakning för att utföra dessa beräkningar!

## 6.2 Källkod

### 6.2.1 Lab3Uppg06.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 6
8   */
9
10 package Lab3Uppg06;
11
12 /**
13  * Testprogram för klassen 'Berakning'.
14  */
15 public class Lab3Uppg06
16 {
17     public static void main(String[] args)
18     {
19         System.out.println("");
20         printSumAndProduct();
21         printMinAndMax();
22     }
23
24     /**
25      * Beräknar och skriver ut summan respektive produkten av talen.
26      */
27     private static void printSumAndProduct()
28     {
29         int a = 4;
30         int b = 8;
31         int c = 78;
32
33         int sum = Berakning.getSum(a, b, c);
34         int product = Berakning.getProduct(a, b, c);
35
36         System.out.print("För talen    ");
37         prettyPrintNumbers(a, b, c);
38         System.out.print("\nbli'r summan    ");
39         prettyPrintNumbers(sum);
40         System.out.print("\noch produkten ");
41         prettyPrintNumbers(product);
42         System.out.println("\n");

```

```

43     }
44
45     /**
46      * Skriver ut det minsta och största talet.
47      */
48     private static void printMinAndMax()
49     {
50         int a = 100;
51         int b = 23;
52         int c = 27;
53
54         int min = Berakning.getMin(a, b, c);
55         int max = Berakning.getMax(a, b, c);
56
57         System.out.print("Bland talen          ");
58         prettyPrintNumbers(a, b, c);
59         System.out.print("\nså är det minsta talet ");
60         prettyPrintNumbers(min);
61         System.out.print("\noch det största talet ");
62         prettyPrintNumbers(max);
63         System.out.println("\n");
64     }
65
66     /**
67      * Skriver ut ett godtyckligt antal tal, kommaseparerade med
68      * undantag för det sista talet där ett "och" skrivs ut.
69      * @param numbers   tal att skriva ut
70      */
71     private static void prettyPrintNumbers(int... numbers)
72     {
73         for (int i = 0; i < numbers.length; i++) {
74             System.out.print "\"" + numbers[i] + "\"");
75
76             if (numbers.length == 1)
77                 return;
78
79             if (i == numbers.length - 2)
80                 System.out.print(" och ");
81             else if (i != numbers.length - 1)
82                 System.out.print(", ");
83         }
84     }
85 }

```

---

Lab3Uppg06.java

## 6.2.2 Berakning.java

---

```

1  /**
2   * DV017A :: Grundläggande programmering i Java
3   * 860224 Jonas Sjöberg
4   * Högskolan i Gävle
5   * tel12jsg@student.hig.se
6   *
7   * Labb #3    Uppgift 6
8   */
9
10 package Lab3Uppg06;

```

```
11
12 public class Berakning
13 {
14     public static int getSum(int a, int b, int c)
15     {
16         return a + b + c;
17     }
18
19     public static int getProduct(int a, int b, int c)
20     {
21         return a * b * c;
22     }
23
24     public static int getMin(int a, int b, int c)
25     {
26         return Math.min(a, Math.min(b, c));
27     }
28
29     public static int getMax(int a, int b, int c)
30     {
31         return Math.max(a, Math.max(b, c));
32     }
33 }
```

---

Berakning.java

### 6.3 Skärmdump

Se Figur 7 för skärmdump på körning av koden i Sektion 6.2.1.



