Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

60 Min.

Programmieren 1

Test 1, HS 2011/2012

R. Diehl, H. Diethelm, M. Pouly, P. Sollberger, Version 7.0

Name: Rohrer Vorname: Felix
(Bitte mit Druckbuchstaben schreiben)

Rahmenbedingungen:

1. Zeit: **90 Minuten**
2. Im Test können maximal **90 Punkte** erreicht werden. Jeder Aufgabe ist eine maximal erreichbare Punktezahl zugeordnet.
3. Schreiben Sie Ihren Namen und Vornamen auf dieses Blatt. Blätter ohne Namensangabe werden nicht bewertet.
4. Es handelt sich um einen schriftlichen Test ohne Einsatz des Computers oder elektronischer Hilfsmittel. Sie dürfen keine Unterlagen verwenden.
5. Sollte die Aufgabenstellung Unklarheiten aufweisen, dürfen Sie eigene Annahmen treffen. Führen Sie diese in der Lösung auf.
6. Schreiben Sie möglichst verständlich und gut leserlich. Missverständliche Lösungen werden nicht berücksichtigt.
7. Benutzen Sie den Freiraum unter den Aufgaben für Ihre Lösung.

Für die Korrektur (nicht ausfüllen!)

1	2	3	4	5	6	7	8	9	10	Punkte
5	8	4/6	5	8/10	7	8/10	19,5/20	9	10	83.5

Aufgabe 1: Begriffe (5 Punkte)

Auf der nächsten Seite finden Sie den Code der Javaklasse `Coffee`.
Kreuzen Sie an, welcher Ausdruck (in Klammer finden Sie einen Hinweis auf die Zeilennummer, wo dieser Ausdruck verwendet wird) zu dem gesuchten Begriff passt.
Hinweis: Es ist immer nur eine Antwort möglich. (0.5 Punkte pro Antwort)

a.) Konstruktor mit Parameter

☐	`public Coffee()`	(Zeile 7)
☒	`public Coffee(int price)`	(Zeile 14)
☐	`public class Coffee`	(Zeile 1)

b.) Instanzvariable

☐	`price`	(Zeile 16)
☐	`money`	(Zeile 38)
☒	`balance`	(Zeile 18)

c.) Accessor

☐	`public Coffee()`	(Zeile 7)
☒	`public int getBalance()`	(Zeile 29)
☐	`public void makeCoffee(int money)`	(Zeile 34)

d.) Formaler Parameter

☒	`price`	(Zeile 16)
☐	`balance`	(Zeile 41)
☐	`beanTracer`	(Zeile 23)

e.) Mutator

☒	`public void makeCoffee(int money)`	(Zeile 34)
☐	`public boolean enoughBeans()`	(Zeile 21)
☐	`public int getBalance()`	(Zeile 29)

f.) Signatur

☐	`public boolean enoughBeans()`	(Zeile 21)
☐	`void makeCoffee(int money)`	(Zeile 34)
☒	`getBalance()`	(Zeile 29)

g.) Zugriffsmodifizierer

☐	`int`	(Zeile 29)
☐	`return`	(Zeile 26)
☒	`public`	(Zeile 1)

h.) Rückgabetyp

☐	`int`	(Zeile 3)
☒	`boolean`	(Zeile 21)
☐	`false`	(Zeile 24)

i.) Methodenkopf

☒	`public boolean enoughBeans()`	(Zeile 21)
☐	`public class Coffee`	(Zeile 1)
☐	`public Coffee(int price)`	(Zeile 14)

j.) Aktueller Parameter

- ☐ `price` (Zeile 14)
- ☐ `balance` (Zeile 31)
- ☒ `"Fill in beans first."` (Zeile 37)

```
1:      public class Coffee
2:      {
3:              private int pricePerCoffee;
4:              private int beanTracer;
5:              private int balance;
6:
7:              public Coffee()
8:              {
9:                      pricePerCoffee = 2;
10:                     beanTracer = 5;
11:                     balance = 0;
12:             }
13:
14:             public Coffee(int price)
15:             {
16:                     pricePerCoffee = price;
17:                     beanTracer = 5;
18:                     balance = 0;
19:             }
20:
21:             public boolean enoughBeans()
22:             {
23:                     if(beanTracer <= 0) {
24:                             return false;
25:                     }
26:                     return true;
27:             }
28:
29:             public int getBalance()
30:             {
31:                     return balance;
32:             }
33:
34:             public void makeCoffee(int money)
35:             {
36:                     if(beanTracer <= 0) {
37:                             System.out.println("Fill in beans first.");
38:                     } else if(money != pricePerCoffee){
39:                             System.out.println("Wrong amount of money.");
40:                     } else {
41:                             balance = balance + money;
42:                             beanTracer = beanTracer - 1;
43:                             System.out.println("Enjoy your coffee!");
44:                     }
45:             }
46:
47:     }
```

5P

Aufgabe 2: Ausdrücke (8 Punkte)

a) Die folgenden Methoden haben jeweils einen oder zwei formale Parameter, welche dann im Test einer Bedingung verwendet werden. Gilt die Bedingung, so wird ein Text auf dem Bildschirm ausgegeben. Beschreiben Sie für jede Methode, welche Eigenschaft die aktuellen Parameter (Eingabewerte) erfüllen müssen, damit der entsprechende Text auf dem Bildschirm erscheint. Führen Sie dazu den begonnenen Satz zu Ende. (1 Punkt pro Antwort)

```
public void test1(int x)
{
        if(x > 2 && x < 7) {
            System.out.println("Good morning.");
        }
}
```

Antwort: Der Text erscheint bei folgenden ganzzahligen Werten (Aufzählung): ...3,4,5,6.....

```
public void test2(int x)
{
        if(x % 2 == 0) {
            System.out.println("Enjoy your breakfast.");
        }
}
```

Antwort: Der Text erscheint bei allen ....geraden..... Zahlen.

```
public void test3(boolean a, boolean b)
{
        if(a ^ b) {
            System.out.println("Would you like some more coffee?");
        }
}
```

XOR

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

Antwort: Der Text erscheint, wenn (die Wahrheitswerte) ..nur eine der beiden true ist. sind.

Wahrheitswerte

```
public void test4(int x, boolean b)
{
        if(x >= 3 && !b) {
            System.out.println("Have a good day.");
        }
}
```

Antwort: Der erste Parameter ist eine ganze Zahl, der zweite ist ein Wahrheitswert.

Der Text erscheint, wenn die Zahl ....grösser gleich 3..... ist und

der Wahrheitswert ...false..... ist.

b) Gegeben sind folgende Variablen:

```
int        i = 5;
String     s = " drinks cost ";
float      f = 2.0f;
```

Schreiben Sie einen Java Ausdruck, der ausschliesslich aus den gegebenen Variablen, den Java Operatoren + und * und dem expliziten Casting `(int)` besteht. Der Ausdruck muss folgendes retournieren. (2 Punkte pro Antwort)

Gewünschter Wert:	**Java Ausdruck:**
`"5 drinks cost 10.0"`	System.out.println(i+s + i*f);
`"10 drinks cost 20.0"`	System.out.println((int)(i*f) + s + i*f*f);

8P

4

Aufgabe 3: Casting (6 Punkte)

a.) In der folgenden Tabelle sind Berechnungen und das erwünschte Resultat vorgegeben. Schreiben Sie den Ausdruck in Java Notation, damit die Berechnung das gewünschte Resultat liefert. (3 Punkte)

1

Gegeben sind folgende Variablen:

```
int    i = 2000;
float  f = 2.0f;
```

Berechnung	gewünschter Wert	Java Ausdruck
$\frac{i}{800}$	2.5	i/800f ✓ i/800f oder (float) i/800
$\frac{7}{f}$	3	(int) (7/f) ✓
$4 \cdot i^3$	32'000'000'000	4*i*i*i ✗ 4L*i*i*i oder 4* (long)*i*i*i

b.) Wird eine Gleitkommazahl an eine Variable eines ganzzahligen Datentyps zugewiesen, so wird/werden:
(1 Punkt)

1

- [x] die Nachkommastellen abgeschnitten
- [ ] die Zahl gerundet
- [ ] der Dezimalpunkt nach rechts verschoben

2

c.) Welche Zuweisungen von Variabeln eines Datentyps sind zulässig (implizites Casting)?
(2 Punkte)

Hinweis: Typ1 → Typ2 bedeutet implizites Casting von Typ1 nach Typ2

- [ ] char → byte
- [x] short → float
- [ ] float → long
- [x] float → double

byte 1
short 2
int 4
long 8

float 4
double 8

boolean 1
char 2

Aufgabe 4: Methoden (5 Punkte) 5P

a.) Man unterscheidet zwischen dem Kopf und dem Rumpf einer Methode. Erklären Sie unmissverständlich die Rolle des Methodenkopfes:
(1 Punkte):

Definiert WAS gemacht wird. 1P

b.) Kreuzen Sie an, ob die Aussagen richtig oder falsch sind. (3 Punkte)

Hinweis: Für jedes falsche Kreuz gibt es einen halben Punkt Abzug! Insgesamt bekommen Sie aber mindestens Null Punkte für diese Teilaufgabe.

1. Methoden dienen auch zum Abstrahieren.	☒ richtig	☐ falsch
2. Methoden unterstützen die Wiederverwendung.	☒ richtig	☐ falsch
3. Objekte interagieren mit Hilfe von Methodenaufrufen.	☒ richtig	☐ falsch
4. Getter-Methoden verändern den Zustand eines Objektes.	☐ richtig	☒ falsch
5. Methoden realisieren das Verhalten von Objekten.	☒ richtig	☐ falsch
6. Rückgabewert, Methodenname und Parameter legen die Signatur einer Methode fest	☐ richtig	☒ falsch

3P

c.) Wieso kompiliert nachfolgende Methode nicht?
(1 Punkt)

```
public int moveHorizontal(int distance)
{
    erase();                // Form an alter Position löschen
    xPosition += distance; // neue Position berechnen
    return distance;        // neue Position zurückgeben
    draw();                 // Form an neuer Position zeichnen
}
```

Bei return wird die Methode umgehend beendet! Die nachfolgenden Codezeilen werden nie ausgeführt. 1P

8

Seite 8/18

Aufgabe 5: Bedingungen (10 Punkte)

a) Die Variable a ist als `char` definiert. Formulieren Sie die Bedingung „a besitzt den Wert a“ in Java. (1 Punkt)

1

```
if ( a == 'a' ) {
...
}
```

(✓)

a == 'a'

b) Die Variable b ist als `float` und die Variable c ist als `long` definiert. Formulieren Sie die Bedingung „c ist kleiner als b“ in Java. (1 Punkt)

0

```
if ( (float) c < b ) {
...
}
```

c < b

f wäre richtig

c) In einer do-while Iteration gibt man die zwei Werte für die Geschwindigkeit und Zeit ein. Die Iteration soll wiederholt werden, wenn die Geschwindigkeit (`int v`) NICHT im Bereich 1 m/s bis 333 m/s (inkl. der Bereichsgrenzen) liegt sowie die Zeit (`int t`) negativ, null oder mehr als 999 Sekunden ist. Formulieren Sie eine entsprechende Bedingung. (4 Punkte)

3

Tipp: Die do-while Iteration wird wiederholt, wenn die Bedingung `true` ist.

```
(v<1 || v>333) && (t<1 || t>999)
```

```
(v<1 || v>333) || (t<1 || t>999)
```

d) Ein Geldbetrag soll je nach Rückerstattungsstatus verschieden eingefärbt werden. Dazu stehen die Eingabefelder "Krankenkasse" und "Beihilfe" zur Verfügung, die den Wert "ja" oder "nein" annehmen können. Hat das Feld "Krankenkasse" den Wert "ja", dann soll der Betrag "gelb" sein, wenn gleichzeitig das Feld "Beihilfe" auch "ja" hat soll der Betrag "grün" werden. Haben beide Felder den Wert "nein", wird der Betrag "rot".

4

Formulieren Sie drei Bedingungen für den Rückerstattungsstatus "rot", "gelb" und "grün". Die Variablen für die Eingabefelder heissen `krankenkasse` und `beihilfe` und sind vom Typ `Boolean`. (4 Punkte)

rot: `!krankenkasse && !beihilfe`

gelb: `krankenkasse && !beihilfe`

grün: `krankenkasse && beihilfe`

Aufgabe 6: Selektion mit if (7 Punkte) 7 P

Gegeben ist die Klasse `Account` mit drei Attributen:

```
// aktueller Kontostand
private int balance;

// im aktuellen Monat abgehobener Betrag
private int payments;

// beschreibt wieviel pro Monat bezogen werden kann (positiver Wert)
private int limit;
```

Damit eine Auszahlung am Geldautomaten vorgenommen werden kann, gelten folgende Bedingungen:

- Die Auszahlung darf die Monatslimite, abzüglich der schon erfolgen Auszahlungen, nicht überschreiten.
- Ist die Auszahlung grösser als 500, müssen nach Ausführung mindestens 50% der Monatslimite auf dem Konto verbleiben, ist sie kleiner, darf der Betrag ohne Einschränkung abgehoben werden.
- Ein Negativkontostand ist nicht erlaubt.

Implementieren Sie die Methode `debitCheck` der Klasse `Account`, die basierend auf dem abzubuchenden aktuellen Betrag (der Parameter ist ein positiver Wert) und den Attributen `balance`, `payments` und `limit` überprüft, ob die Auszahlung erfolgen darf (Rückgabewert `true`) oder nicht (Rückgabewert `false`). (7 Punkte)

```
public boolean debitCheck(int amount)
{
  boolean debit = false;
  if((amount <= (limit-payments)) && ((balance - amount) >= 0))
  {
    if(amount <=500)
    { debit = true;
    } else {
      if((balance - amount) > (limit /2))
      {
        debit = true;
      }

    }
  }
  return debit;
}
```

Aufgabe 7: Selektion mit switch (10 Punkte)

a) Gegeben ist der folgende Programmausschnitt, der die Kosten eines Kaffees berechnet:

```
// Coffee kind: 1=Caffe Latte 2=Latte Macchiato
// Coffee size: 1=Small 2=Medium 3=Large
public int calcCost(int size, int kind)
{
   int cost = 0;
   switch (kind*10+size) {
      case 13:
         cost += 20;
      case 12:
         cost += 25;
      case 11:
         cost += 30;
         break;
      case 22:
         cost += 30;
      case 21:
         cost += 35;
         break;
      case 23:
         cost = 70;
         break;
      default:
         System.out.println("parameter out of range");
   }
   return cost;
}
```

Kreuzen Sie an, ob die Aussagen richtig oder falsch sind. Für jedes richtige Kreuz erhalten Sie einen Pluspunkt; für jedes falsche gibt's einen Minuspunkt! Insgesamt bekommen Sie aber mindestens Null Punkte für diese Teilaufgabe. (5 Punkte)

Der Compiler meldet einen Fehler	☒ richtig	☐ falsch
Die zulässigen Datentypen des Arguments für das switch-Konstrukt sind `byte`, `short`, `int` und `long`.	☐ richtig	☒ falsch
Der günstigste Kaffee ist ein „Small Caffe Latte“	☒ richtig	☐ falsch
Der teuerste Kaffee ist ein „Large Latte Macchiato“	☐ richtig	☒ falsch
Ein „Medium Latte Macchiato“ ist günstiger als ein „Medium Caffe Latte“	☐ richtig	☒ falsch

5

b) Vervollständigen Sie die Methode `checkNumber` mit Hilfe der `switch`-Anweisung. Als aktuellen Parameter übergibt man der Methode eine Zahl von 2 bis 14, als Resultat liefert die Methode eine Aussage über die Zahl. Es wird nur eine Aussage ausgegeben mit der folgenden Priorität (5 Punkte):

out1 (1) "Nummer ist eine Primzahl" // 2, 3, 5, 7, 11, 13
out2 (2) "Nummer ist eine Quadratzahl" // 4, 9,
out3 (3) "Nummer ist eine gerade Zahl" // 6, 8, 10, 12, 14
out4 (4) "Nummer ist ausserhalb des Bereiches" //

Tipp: Sie müssen nicht die vollständige `System.out.println` Anweisung hinschreiben. Sie können auch nur auf die Nummer der Aussage verweisen.

```
public void checkNumber(int number)
{
  switch (number) {
    case 2: case 3: case 5: case 7: case 11: case 13:
      //out1 ausgeben
      break;

    case 4: case 9:
      //out2 ausgeben
      break;

    case 6: case 8: case 10: case 12: case 14:
      //out3 ausgeben
      break;

    case default:
      //out4 ausgeben




  }
}
```

Aufgabe 8: Implementation einer Klasse (20 Punkte)

Hinweis: Bei der Korrektur dieser Aufgabe werden auch die *Syntax* und das Beachten der *Programmierrichtlinien* (Namensgebung, Klammernsetzung, Einrückung) bewertet. (2 Punkte) 1,5

Implementieren Sie wie folgt schrittweise die Klasse `Bulb` (= Glühlampe)**:**

a) *Klassenkopf* (1 Punkt)

```
public class Bulb
{

}
```

b) Zwei *Attribute*: (1 Punkt)
- Farbe der Glühlampe (als Zeichenkette)
- Helligkeit der Glühlampe (als ganze Zahl)

```
private String color;
private int brightness;
```

c) Einen *Konstruktor* mit 1 Parameter zum Initialisieren der Farbe. Die Helligkeit wird defaultmässig auf Null gesetzt. (1 Punkt)

```
public Bulb(String newColor)
{
    color = newColor;
    brightness = 0;
}
```

d) Einen *Konstruktor* ohne Parameter, welcher die Farbe auf weiss und die Helligkeit auf Null setzt. (1 Punkt)

```
public Bulb()
{
    color = "white";
    brightness = 0;
}
```

e) Eine *getter-Methode* für die Farbe der Lampe. (1 Punkt)

```
public String getColor()
{
    return color;
}
```

f) Eine *setter-Methode* zum Festlegen der Helligkeit der Lampe (0..10). Bei einem unzulässigen Wert soll der Zustand unverändert bleiben. (2 Punkte)

```
public void setBrightness (int newBrightness)
{ if((newBrightness >= 0) && (newBrightness <= 10)) {
    brightness = newBrightness;
  }
}
```

g) Eine *Mutator-Methode* zum Einschalten der Lampe (auf volle Helligkeitsstufe 10). (1 Punkt)

```
public void switchOn()
{
  brightness = 10;

}
```

h) Eine Methode `printState()`, welche sinngemäss folgende Ausgabe auf die Konsole macht: (4 Punkte)

```
white |x.........|      z.B. weisse Lampe mit Helligkeitsstufe 0
white |........x..|     z.B. weisse Lampe mit Helligkeitsstufe 8
white |.........x|      z.B. weisse Lampe mit Helligkeitsstufe 10
```

```
public void printState()
{
  String res = color + " |";
  for (int x = 0; x < brightness; x++) {
     res += ".";
  }

  res += "x";

  for(int x = brightness +1; x <=10 ; x++) {
     res += ".";
  }

  res += "|";

  System.out.println(res);
```

}

Die Klasse `Bulb` anwenden bzw. eine weitere Klasse **`TrafficLight`** implementieren:

i) *Klassenkopf* (1 Punkt)

```
public class TrafficLight
{

}
```

j) Vier *Attribute*: (2 Punkte)
- drei Lampen (vgl. Verkehrsampel und Klasse `Bulb`)
- Betriebsart (automatic vs. manual) der Verkehrsampel (als Wahrheitswert)

```
private Bulp bulpGreen;
private Bulp bulpOrange;
private Bulp bulpRed;
private boolean operatingManual; // true = manual; false = automatic
```

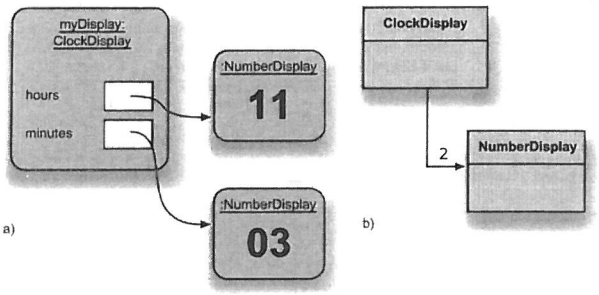
k) Einen *Konstruktor* ohne Parameter. Nach dem Erzeugen soll eine Verkehrsampel defaultmässig manuell laufen und voll auf Rot stehen. (3 Punkte)

```
public TrafficLight()
{
    bulpGreen = new Bulp("green");
    bulpOrange = new Bulp("orange");
    bulpRed = new Bulp("red");
    operatingManual = true;
    bulpRed.switchOn(); //Rot einschalten, Grün & Orange sind per Default off.
}
```

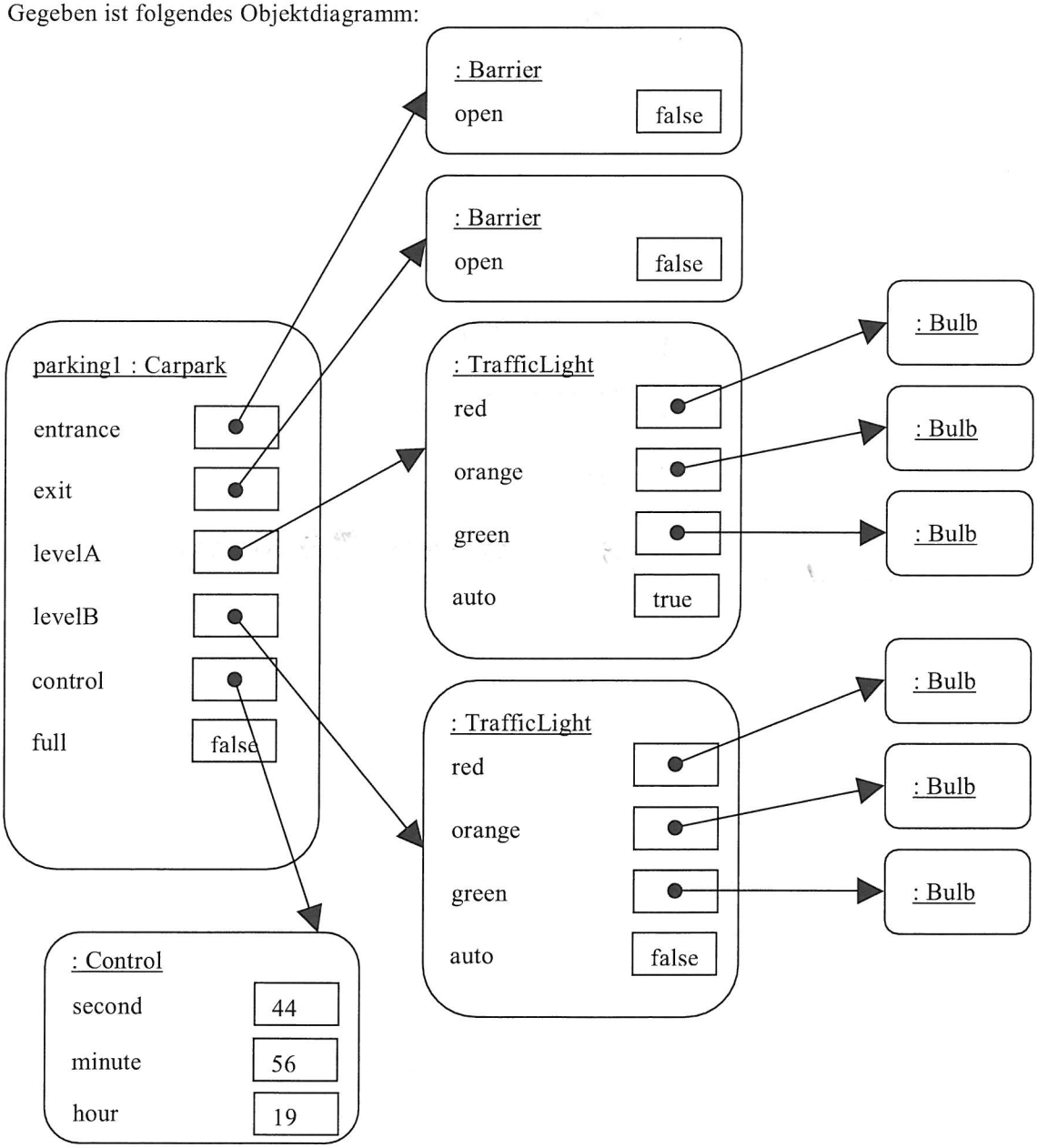
13,5P

Aufgabe 9: Objekt- und Klassendiagramme (9 Punkte)

Zu Ihrer Erinnerung aus dem Buch:



Gegeben ist folgendes Objektdiagramm:



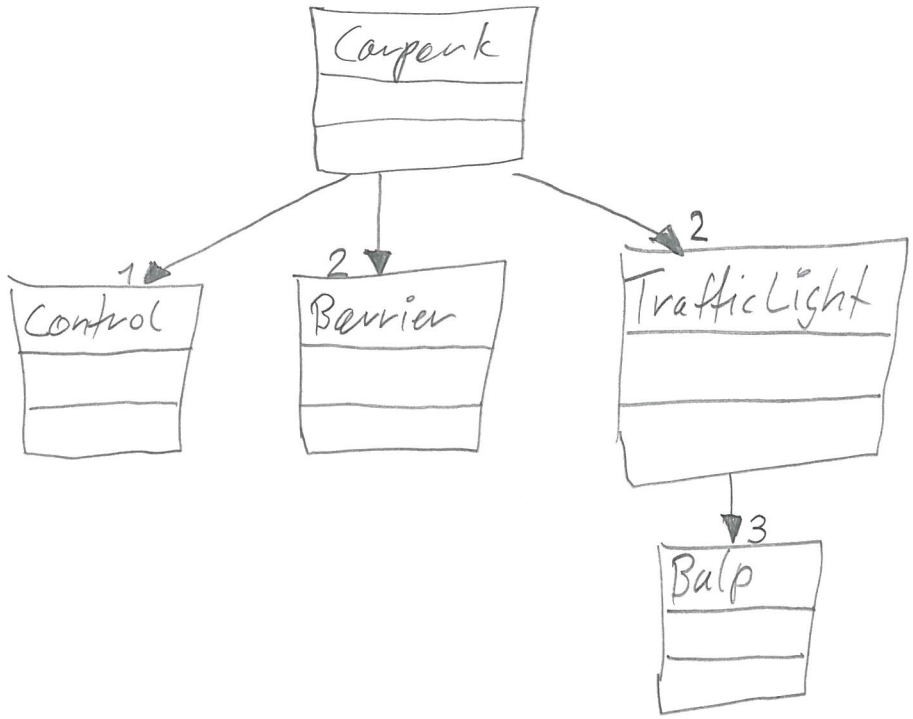
a) Welche der folgenden sechs Antworten ist richtig? (3 Punkte)

Kreuzen Sie an, ob die Aussagen richtig oder falsch sind. (Für jedes richtige Kreuz erhalten Sie einen halben Punkt; für jedes falsche gibt's einen halben Punkt Abzug! Insgesamt bekommen Sie aber mindestens null Punkte für diese Aufgabe.)

1. `second` ist eine Instanzvariable und enthält eine ganze Zahl. ☒ richtig ☐ falsch ✓
2. `red` ist eine Instanzvariable und enthält ein Objekt der Klasse `Bulb`. Objektreferenz ☐ richtig ☒ falsch ✓
3. In der Anwendung gibt es 11 Objekte. 12 ☐ richtig ☒ falsch ✓
4. Im Objektdiagramm gibt es 11 Objektreferenzen. ☒ richtig ☐ falsch ✓
5. Das Objekt `parking1` hat 6 Instanzvariablen. ☒ richtig ☐ falsch ✓
6. Ein Objekt der Klasse `Barrier` hat keine Referenzvariable. ☒ richtig ☐ falsch ✓

3P

b) Zeichnen Sie das *Klassendiagramm* auf, das zum Objektdiagramm von vorne passt. (In die Kästchen hinein müssen Sie nur die Klassennamen schreiben.) (6 Punkte)



✓

6P

Aufgabe 10: Korrektur von Programmcode (10 Punkte)

Im Code der Klasse `TicketMachine` haben sich verschiedenartige Fehler eingeschlichen.

Compiler-Fehler:

1. *cannot return a value from method whose result type is void* (2 Punkte)
2. *possible loss of precision* (2 Punkte)

Semantische Fehler: (D.h. das Programm führt etwas "Unsinniges" aus.)

3. fehlerhafter Ausdruck (2 Punkte)
4. falsche Berechnung (2 Punkte)
5. unpräzise Überprüfung (2 Punkte)

Korrigieren Sie alle 5 Fehler unmissverständlich direkt im Code:

```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    public TicketMachine(int ticketCost)
    {
        price = ticketCost;
        balance = 0;
        total = 0;
    }

    public int getPrice()
    {
        return price;
    }

    public int getBalance()
    {
        return balance;
    }

    public void insertMoney(int amount)
    {
        if(amount > 0) {
            balance = balance + amount;
        }
    }

    public void printTicket()
    {
        if(balance >=price) {
            System.out.println("SBB Ticket");
            System.out.println("Horw-Luzern");
            total = total + price;
            balance = price;
        }
        // Fortsetzung siehe nächste Seite
```

```
        else {
            System.out.println("Insert at least: " + "price - balance");
        }
    }

    public void refundBalance()
    {
        int amountToRefund;
        amountToRefund = balance;
        balance = 0;
        return amountToRefund;
    }
}
```

(price - balance) ✓

2P

2P

2P

---- Ende des Tests ----