University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Informatik I – Eprog HS13

Übung 4

1 Aufgabe: Codeverständnis

1.1 Lernziele

1. Syntax von Schleifen und Verzweigungen verstehen.
2. Codelesen und Codeverständnis trainieren.

1.2 Aufgabenstellung

a) Code Snippets

Welche der nachfolgenden dreizehn Code-Fragmente sind syntaktisch korrekt und wie lautet in diesem Fall die Ausgabe? Begründen Sie Ihre Antwort. Benutzen Sie dazu folgende Hilfsmittel:

- Walter Savitch, Java: An Introduction to Problem Solving & Programming
- Java API

```
1 int sum = 0;
2 for(i=0;i<7;i++){
3 sum += i;
4 }
5 System.out.println(sum);
```

Listing 1: Snippet 1

```
1 int sum = 0;
2 for(int i=0;i<6;i++){
3 sum += i;
4 }
5 System.out.println(sum);
```

Listing 2: Snippet 2

```
int sum = 0;
for(int i=6;i>0;i--){
sum += i;
}
System.out.println(sum);
```

Listing 3: Snippet 3

```
int i = 0;
for(i=5;i>0;i--){
System.out.print(i);
}
```

Listing 4: Snippet 4

```
int j = 0;
for(int i=0;i<5;j++){
System.out.print(i);
}
```

Listing 5: Snippet 5

```
for(int i=0;i<10;i++){
   i++;
   System.out.print(i);
}
```

Listing 6: Snippet 6

```
int sum = 0;
for(int i=0;i<5;i++)
sum++;
sum++;
System.out.println(sum);
```

Listing 7: Snippet 7

```
int sum = 0;
for(int i=0;i<5;i++)
sum += i;
sum += i;
System.out.println(sum);
```

Listing 8: Snippet 8

```
int b = 10;
String s = if ( b % 2 == 0 ) {
"Even";
} else {
```

```
"Odd";
}
System.out.println(s);
```

Listing 9: Snippet 9

```
int a = 4;
if( a < 10 ){
a = 3;
} else if ( a < 5 ) {
a = 2;
} else {
a = 1;
}
System.out.println(a);
```

Listing 10: Snippet 10

```
Integer a = null;
if( (a = 10) > 0 ){
System.out.println(a);
}
```

Listing 11: Snippet 11

```
int sum = 1;
int i;
for(i=0;(i<10 && sum < 100);i++){
   sum *= 2;
}
System.out.println(i);
System.out.println(sum);
```

Listing 12: Snippet 12

```
int count = 0;
for(int i=0;i<5;i++)
   for(int j=0;j<i;j++)
      for(int k=0;k<j;k++)
         count++;
System.out.println(count);
```

Listing 13: Snippet 13

2 Aufgabe: Primitive Datentypen, Binärwerte und logische Operationen

2.1 Lernziele

1. Verstehen wie Java primitive Datentypen intern darstellt
2. Zwei Java Logik Operatoren kennen lernen
3. Umrechnung in Binärdarstellung üben

2.2 Binärdarstellung ganzer Zahlen

Java verwendet intern für `byte`, `short`, `int` und `long` eine Zweierkomplement Darstellung. Wie lautet entsprechend die binäre Darstellung des folgenden `int` Wertes?

```
1 int a = - 1025;
```

Listing 14: Snippet 14

Bei einem cast, von einem grösseren zu einem kleineren ganzzahligen Datentyp, werden die überzähligen höchstwertigen Bits von Java abgeschnitten.

Nutzen Sie diese Tatsache, ihr Wissen über die Grösse der Java-Datentypen, sowie Ihre Kenntnisse aus der Vorlesung, um die Ergebnisse der folgenden casts möglichst detailliert zu erklären.

```
1 int a = 98305;
2 short b = (short) a;
3 System.out.println(b);
4
5 // Ausgabe: -32767
```

Listing 15: Snippet 16

```
1 int a = 98305;
2 byte b = (byte) a;
3 System.out.println(b);
4
5 // Ausgabe: 1
```

Listing 16: Snippet 17

2.3 Binäroperationen

Der Operator `&` sorgt für die bitweise `UND` Verknüpft zweier Zahlen. Der Operator `|` sorgt für die bitweise `ODER` Verknüpfung zweier Zahlen. Dies bedeutet also, dass jeweils die Bit-Stelle x der beiden Zahlen `UND` bzw. `ODER` Verknüpft wird und den Wert der Bit-Stelle x des Ergebnisses ergibt. Denken Sie an die logische Verknüpfung von boolschen Werten, welche Sie schon kennen. Bei der bitweisen `UND` bzw. `ODER` Verknüpfung, entspricht ein Bit-Wert `1` einem `true` und ein Bit-Wert `0` einem `false`. Beispiel:

```
1  short a = 33;
2  short b = 32;
3  short c = a & b;
4
5  // Ergebnis : c = 32
6  //
7  // a = 33 = 0000 0000 0010 0001
8  // &
9  // b = 32 = 0000 0000 0010 0000
10 // ----------------------------
11 // c = 32 = 0000 0000 0010 0000
```

Listing 17: Snippet 18

Lösen Sie die folgenden Aufgaben. Konstruieren Sie die jeweils fehlende Variable so, dass die bitweise Verknüpfung das geforderte Ergebnis liefert. Beispiel:

```
1  // Beispiel Aufgabe: c soll 1 sein. Welches b passt?
2
3  short a = 3;
4  short b = ???;
5  short c = a & b;
6
7  // Loesung:
8  //
9  // a ist 0000 0000 0000 0011 (3) und c soll 0000 0000 0000 0001 (1) sein.
10 // Daher ist b = 0000 0000 0000 0001 (1), weil:
11 //
12 // a = 3 = 0000 0000 0000 0011
13 // &
14 // b = 1 = 0000 0000 0000 0001
15 // ----------------------------
16 // c = 1 = 0000 0000 0000 0001
```

Listing 18: Snippet 19

Beachten Sie, dass es teilweise mehrere richtige Lösungen gibt. Versuchen Sie alle zu finden.

```
1  // Aufgabe: c soll 16 sein. Welches b passt?
2
3  short a = 48;
4  short b = ???;
5  short c = a & b;
```

Listing 19: Snippet 20

```
// Aufgabe: c soll 24 sein. Welches b passt?

short a = 8
short b = ???;
short c = a | b;
```

Listing 20: Snippet 21

```
// Aufgabe: d soll 13 sein. Welches b passt?

short a = 4;
short b = 8;
short c = ???;
short d = a | b | c;
```

Listing 21: Snippet 22

3 Aufgabe: Programmierung

3.1 Lernziele

1. Kenntnisse über objektorientierte Programmierung und Wissen aus den technischen Grundlagen kombinieren und anwenden.
2. Methoden der Klassen `String`, `Math` und `Integer` kennen lernen.
3. Arbeit mit Schleifen und Verzweigungen üben.

3.2 BinaryConverter

Schreiben Sie eine Klasse `BinaryConverter`, welche es ermöglicht, gegeben Binärzahlen in Dezimalzahlen umzuwandeln. Dabei soll die gegebene Binärzahl wahlweise Vorzeichenlos oder als Einerkomplement interpretiert werden. Jede der beiden Umwandlungen soll in einer eigenen Klasse implementiert werden. Die Klasse `BinaryCoverter` soll diese Klassen verwenden, um die passende Umwandlung durchzuführen. Das Ergebnis soll ausgegeben werden. Verwenden Sie die folgenden Klassengerüste.

```
1  class BinaryConverter {
2      ...
3      /*
4       * String bNumber: The binary number
5       * int mode: How to interpret the number.
6       * 0 = Without sign. 1 = One's complement
7       */
8      public void convert(String bNumber,int mode)
9      ...
10 }
```

```
1  class NoSignConversion {
2      ...
3      /*
4       * returns: Decimal value of the given binary number
5       */
6      public Integer convert(String bNumber)
7      ...
8  }
```

```
1  class OnesComplementConversion {
2      ...
3      /*
4       * returns: Decimal value of the given binary number
5       */
6      public Integer convert(String bNumber)
7      ...
8  }
```

Hinweise

1. Überlegen Sie sich, wie sie die Umwandlung handschriftlich durchführen würden. Versuchen Sie, diesen Ablauf zu implementieren.
2. Sie werden Methoden der Java Klasse `Math` brauchen. Diese bietet verschiedene mathematische Operationen. Informieren Sie sich in der Java Doc über diese Klasse.
3. Sie werden Methoden der Java Klasse `String` brauchen. Denken Sie daran, dass ein String eine Aneinanderreihung von `char` ist. Die Klasse `String` ermöglicht Ihnen, auf die einzelnen Zeichen zuzugreifen. Suchen Sie die passende Methode in der `String` Java Doc.
4. Sie können die Methode `valueOf` der Klasse `Integer` nutzen, um eine Zeichenkette, welche einen Dezimalwert repräsentiert, in einen `Integer` umzuwandeln.
   Beispiel: `Integer a = Integer.valueOf("42")`
5. Sie können die Methode `getNumericValue` der Klasse `Character` nutzen, um einen `char` in seinen numerischen Wert umzuwandeln. Informieren Sie sich in der Java Doc über diese Methode.

3.3 TestDriver

Schreiben sie einen Klasse `BinaryConverterTestDriver` und testen Sie die Klasse `BinaryConverter` mit verschiedenen Werten.

3.4 Validierung

Implementieren Sie eine Klasse `BinaryNumberValidator` mit einer Methode `validate(String)` welche einen `Boolean` liefert. Diese Methode liefert `true`, falls der gegebene `String` eine valide Binärzahl ist und `false`, falls dies nicht der Fall ist. Der `BinaryConverter` soll eine Instanz dieser Klasse verwenden, um die gegebene Binärzahl zu prüfen. Falls es sich nicht um eine Binärzahl handelt, soll eine Fehlermeldung ausgegeben werden.

3.5 Alternative Umwandlung

Finden Sie mit Hilfe der Java Doc herraus, wie Sie die Methode `parseInt(String,int)` der Klasse `Integer` nutzen können, um die vorzeichenlose Umwandlung zu implementieren. Erzeugen Sie dazu eine eigene Klasse `AlternativNoSignConversion` und erweitern Sie die Klasse `BinaryConverter` so, dass ein dritter Modus - welcher die neue Klasse verwendet - unterstütz wird.