Technische Universität Braunschweig

Institut für Anwendungssicherheit

Programmieren 1

Prof. Dr. Martin Johns

02.09.2019

Nachname:

Vorname:

Matrikelnummer:

Klausurnummer: 001

Studiengang: ☐ Bachelor ☐ Master ☐ Diplom ☐ Frühstudium ☐ Erasmus

Fachrichtung: ☐ Informatik ☐ Wirtschaftsinformatik ☐ Mathematik ☐ Physik
☐ Mobilität und Verkehr ☐ Psychologie ☐ Maschinenbau (☐ Mechatronik)
☐ Elektrotechnik ☐ Wirtschaftsingenieurwesen (☐ Bauing ☐ E-Technik ☐ Maschbau)
☐ Finanz- und Wirtschaftsmathematik ☐ IST ☐ Sonstige:_____

Versuch der Notenverbesserung: ☐

Die Bearbeitungszeit beträgt 120 Minuten. Die Klausur besteht aus 6 Aufgaben und hat maximal 80 Punkte.

Question:	1	2	3	4	5	6	Total
Points:	10	13	13	10	9	25	80
Score:							

Note:

Hinweise: Die Aufgaben befinden sich in der Regel auf den Rückseiten der Blätter! Bitte stellen Sie sicher, dass Sie keine der Aufgaben übersehen. Die Noten werden auf der Institutshomepage mit Punkten unter der Klausurnummer **vorläufig** veröffentlicht, merken Sie sich entsprechend Ihre Klausurnummer.

Während der Klausur darf sich nicht Unterhalten werden, Handys und vergleichbare Geräte sind auszuschalten und außerhalb der eigenen Reichtweite zu lagern. Verstoß gegen diese Regeln werden als Täuschungsversuch gewertet und führen entsprechend zum nicht Bestehen der Klausur sowie einer Meldung an das Prüfungsamt.

Sollten Sie eine Frage haben melden Sie sich und warten Sie bis eine Aufsicht kommt, stehen Sie nicht eigenständig auf!

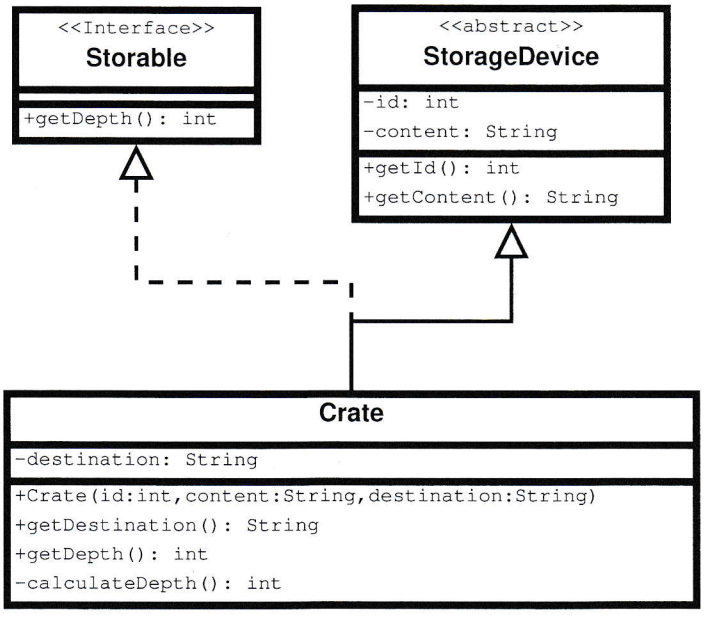
```java
interface Storable {
  public int getDepth();
}

abstract class StorageDevice() {
  private int id;
  private String content;

  public int getId() {...};
  public String getContent() {...};
}

class Crate extends StorageDevice implements Storable {

  String destination;

  public Crate(int id, String content, String destination) {...};
  public String getDestination() {...};
  public int getDepth() {...};
  private int calculateDepth() {...};
}
```

Figure 1: Java Quellcode.

1. In dieser Aufgabe beschäftigen Sie sich mit UML Diagrammen. Sie müssen ein UML Diagram in Java Quellcode übersetzen sowie umgegekehrt.

   (a) (5 Punkte) In Fig. 1 ist Java Quellcode gegeben. Übertragen Sie diesen in ein entsprechendes UML Diagram.

Solution:



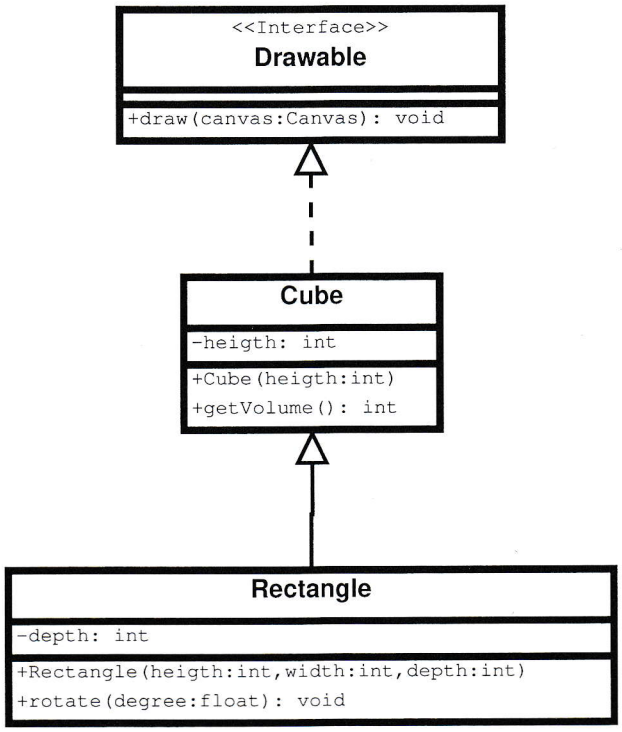


Figure 2: UML Diagram

(b) (5 Punkte) In Fig. 2 ist ein UML Diagram mit 3 Elementen gegeben. Übertragen Sie dieses in Java Quellcode. Lassen Sie dabei den Methodenkörper **leer**. Sollte die Methode einer Superklasse nicht explizit überschrieben werden, wird die Implementierung der Superklasse von der Kindklasse übernommen.

Solution:

```
interface Drawable {
  void draw(Canvas canvas) {...};
}

class Cube implements Drawable {
  private int heigth;
  public Cube(int heigth) {...};
  public int getVolume() {...};
  public void draw(Canvas canvas) {...};
}

public Rectangle extends Cube {
  private int depth;

  public Rectangle(int heigth, int width, int depth)
     {...};
  public void rotate(float degree) {...};
}
```

2. Welche der folgenden Schleifen terminieren?

   Geben Sie im Falle der Terminierung an, welche Werte die Variablen x und y annehmen, nachdem die jeweiligen Anweisungen ausgeführt wurden. Sollte eine Programmausschnitt nicht terminieren, lassen Sie die Felder für x und y leer.

   Ausgefüllte Felder für x und y werden als Terminierung gewertet!

   (a) (2 Punkte) Betrachten Sie den folgenden Java Code:

```
int x = 0;
int y = 18;
while(++x != y) {
  x = x + 2;
  if(y % 2 == 0) {
    --x;
  } else {
    x = y % 8;
  }
}
```

   terminiert: nein x = n/a y = n/a

   (b) (2 Punkte) Betrachten Sie den folgenden Java Code:

```
int x = 6;
int y = 3;
while(x > 0) {
  if(x % 2 == 0) {
     int buff = x;
     x = y - 1;
     y = buff;
   } else {
     --x;
   }
 }
```

   terminiert: ja x = 0 y = 4

   (c) (3 Punkte) Jedem der folgenden Ausdrücke wird die Deklaration int x = 42; vorrangesetzt.

   Geben Sie für jden der Ausdrücke den Typ und den Wert des Ausdrucks an.

   1. (x/.2) * 2 Typ: double Wert: 420.0
   2. (x++ % 21) + x Typ: int(eger) Wert: 43
   3. ++x == x++ Typ: bool(ean) Wert: true

(d) (6 Punkte) Geben Sie jeweils den Inhalt der einzelnen Array Felder von a nach der jeweiligen Ausführung der beiden äußeren Schleifen an.

```
int[] a = new int[6]
//Schleife 1
for(int i = 0; i != 6; ++i) {
   if(i == 3 || i == 5) {
      a[i] = i;
   } else {
      a[i] = 2 * i;
   }
}
//Schleife 2
int n = a.length;
for(int i = 0; i < n-1; i++) {
   for(int j = 0; j < n-i-1; j++)
      if(a[j] > a[j+1]) {
         int temp = a[j];
         a[j] = a[j+1];
         a[j+1] = temp;
      }
}
```

Nach der ersten Schleife:

$a[0] =$ 0 $a[1] =$ 2 $a[2] =$ 4
$a[3] =$ 3 $a[4] =$ 8 $a[5] =$ 5

Nach der zweiten Schleife:

$a[0] =$ 0 $a[1] =$ 2 $a[2] =$ 3
$a[3] =$ 4 $a[4] =$ 5 $a[5] =$ 8

3. (13 Punkte) Auf der folgenden Seite ist ein Java Programm gegeben.

   Füllen Sie die Lücken im Programmtext so aus, dass das Programm die nachfolgende Ausgabe erzeugt:

```
Cooled @ -5 are Popsicles
Cooled @ -10 are Icesicles
Contains 18 units of Coffee
```

   Sie dürfen dabei den vorhandenen Code auf keine Art und Weise erweitern oder ändern. Nicht jede Lücke muss notwendigerweise auch gefüllt werden.

```
interface Storable {
  int __getWidth()__;
  int getHeight();
  int getDepth();
  String format();
}

abstract ____class____ ShippingContainer implements __Storable__ {
  protected String content;
  public int getWidth() { return 10; }
  public int getHeight() { return 10; }
  public int getDepth() { return 20; }
}

class CooledShippingContainer ___extends___ ShippingContainer {
  private int temp;
  public CooledShippingContainer(int temp, String content) {
    ____this____.temp = temp;
    this.content = content;
  }
  public String format() {
    return "Cooled @ " + this.____temp____ + " are " + this.content;
  }
}

class CoffeeCrate __implements__ __Storable__ {
  private count;
  public int getWidth() { return 10; }
  public int getHeight() { return 20; }
  public int getDepth() { return 10; }
  public CoffeeCrate(int count) { this.count = count; }
  public String format() {
    return "Contains " + ____this____.count + " units of Coffee";
  }
}

class Main {
  public static void main(String[] args) {
    __Storable[]__ items = {
      new CooledShippingContainer(_____-5_____, ''Popsicles''),
      new CooledShippingContainer(-10, ''Icesicles''),
      new __CoffeeCrate__(18);
    }
    for(Storable item : ____items____) {
      System.out.println(item.format());
    }
  }
}
```

Es nicht notwendig

4. Gegeben sind die folgendenden rekursiven Funktionen:

```
static int global = 0;

static int f(int n) {
  if(n % 2 == 0) {
     return g(n-1);
   } else {
     return g(n);
   }
}

static int g(int n) {
  if(n % 2 != 0 && n > 0) {
    return f(n - 1) + f(n - 2);
  } else {
    return global++;
  }
}
```

(a) (1 Punkte) Terminiert der rekursive Aufruf für $f(3)$?

Solution: Ja

(b) (1 Punkte) Welchen Wert liefert der Aufruf $f(3)$

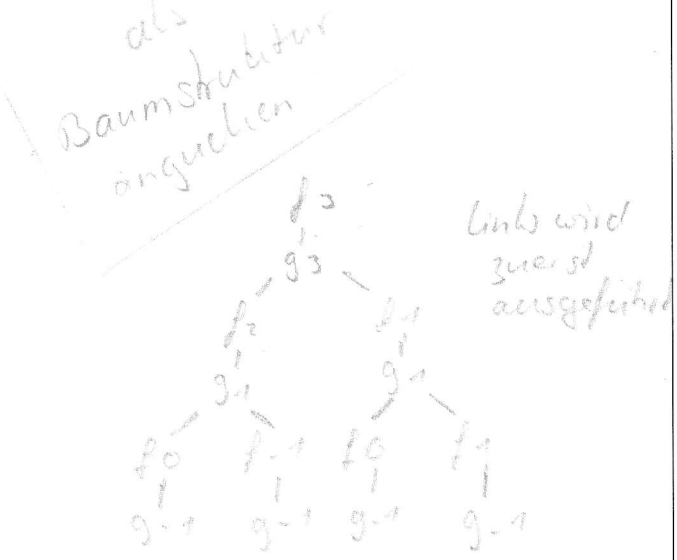
Solution: $f(3) = 6$

(c) (7 Punkte) In welcher Reihenfolge und mit welchen Parametern werden f und g bei dem Aufruf $f(3)$ aufgerufen?

Geben Sie entweder eine nummerierte Liste der Aufrufe an oder zeichnen Sie einen Rekursionsbaum, dessen linke Blätter in der Reihenfolge vor den rechten Blättern kommen.

Solution:

1. f(3)
2. g(3)
3. f(2)
4. g(1)
5. f(0)
6. g(-1)
7. f(-1)
8. g(-1)
9. f(1)
10. g(1)
11. f(0)
12. g(-1)
13. f(-1)
14. g(-1)



(d) (1 Punkte) Was ist die maximale Rekursionstiefe, dass heißt die maximale Anzahl gleichzeitig aktiver Aufrufe?

Solution: Maximale Anzahl gleichzeitiger Aufrufe ist 6.

5. Im Folgenden wird eine **statische** API (Application Programming Interface) gegeben, welche ein CustomArray definiert und Methoden zur Generierung und Manipulation bietet. Dieses Array ist **kein** typisches Java Array.

   Nutzen Sie **ausschließlich** diese API um die gegebenen Probleme zu lösen. **Sie dürfen keine weiteren Java Features nutzen** (e.g. Schleifen, If, Switch) und lediglich Variablen zum Zwischenspeichern anlegen, sowie die Grundrechenoperationen verwenden.

   Erstellen Sie für jede Teilaufgabe einen eigenen Programmausschnitt. Um den Schreibaufwand zu minimieren, dürfen Sie die Synonyme "AM" (ArrayManipulator) und "CA" (CustomArray) verwenden.

```
interface CustomArray {

  /** Liefert den Inhalt des Array an der angegebenen Position
      zurueck

      @param die angefragte Arrayposition
             (das erste Element hat Position 0)
      @return der Wert an der angefragten Array Position
   */
  int getAt(int position);

  /** Gibt das Array auf der Konsole aus
  */
  void print();

}

class ArrayManipulator {

  /** Generiert ein CustomArray mit zufaelligen Werten

      @param die Laenge des zu generierenden Array
      @return CustomArray der Laenge length
  */
  public static CustomArray generate(int length);

  /** Gibt ein neues Array zurueck, welches die Werte des
      uebergebenen Arrays in aufsteigend sortierter Form
         beinhaltet

      @param das zu sortierende Array
      @return ein neues Array, aufsteigend sortiert
    */
  public static CustomArray sort(CustomArray array);

  /** Summiert die Werte des uebergebenen Array auf

      @param das Array, dessen Werte aufsummiert werden sollen
      @return die Summe aller Werte des Array
  */
  public static float sum(CustomArray array);

}
```

(a) (2 Punkte) Generieren Sie ein Array der Länge 9 und geben Sie den Inhalt des Array aus.

Solution:

```
AM.generate(9).print();
```

(b) (4 Punkte) Generieren Sie ein weiteres Array der Länge 9. Finden Sie das größte, kleinste und Median Element[1] des Array und speichern Sie die Werte in entsprechend benannten Variablen.

Solution:

```
CA a = AM.sort(AM.generate(9));
int smallest = a.getAt(0);
int largest  = a.getAt(8);
int median   = a.getAt(4);
```

(c) (3 Punkte) Generieren Sie ein weiteres Array der Länge 9. Berechnen Sie den Durchschnittswert und ermitteln Sie die Differenz zwischen Durchschnittswert und Median. Speichern Sie die Ergebnisse in entsprechend benannten Variablen.

Solution:

```
CA a = AM.sort(AM.generate(9));
float avg = AM.sum(a) / 9.0;
float delta = avg - a.getAt(5);
```

[1] Der Median ist der Wert einer aufsteigend sortierten Liste, der genau in der Mitte liegt.

6. Gegeben ist ein einfacher Kompressions-Algorithmus für Zeichenketten.

   Der Algorithmus zählt zunächst die Anzahl gleicher aufeinander folgender Buchstaben, Daraufhin wird die Teilzeichenkette durch die Anzahl der gezählten Buchstaben als positive Zahl n sowie einem einem Repräsentanten des Buchstaben selbst ersetzt. Für die Aufgabe darf angenommen werden, dass $0 < n < 10$. Zum Beispiel wird die Zeichenkette *aaaa* zu $4a$.

   Dieser Algorithmus wird auf einem kompletten String angewendet. Am Ende wird ein String zurück gegeben wird, der aus Paaren von Zahlen und Buchstaben besteht. Zum Beispiel wird *aaaabbbbsswwrrrz* zu $4a4b2s2w3r1z$.

   Ihre Aufgabe ist es nun, den Algorithmus zum Entpacken einer vorher komprimierten Zeichenkette zu programmieren, sodass die Zeichenkette nach Entpacken gleich der Zeichenkette vor dem Komprimieren ist.

   (a) (16 Punkte) Schreiben Sie eine Hilfsfunktion, die einen komprimierten String erwartet und diesen Entpackt. Sollte die Funktion eine ungültige Zeichenkette bekommen, die eine 0 enthält (z.B. $0a4b$) sollen Sie eine *IllegalArgumentException* werfen. Ansonsten wird die entpackte Zeichenkette zurück gegeben. Weitere Fehlerfälle müssen nicht beachtet werden.

   (b) (9 Punkte) Schreiben Sie die Main Funktion, die

   - vom Nutzer als erstes Kommandozeilenargument eine Zahl zwischen 0 und 3 erwartet
   - sollte das Kommandozeilenargument nicht stimmen, geben Sie eine entsprechende Fehlermeldung aus und beenden Sie das Programm
   - sollte eine korrekte Eingabe gemacht worden sein, wählt das Programm basierend auf der Eingabe eine der folgenden Zeichenketten aus und dekomprimiert diese mit der Hilfsfunktion aus dem ersten Aufgabenteil.
   - Fangen Sie mögliche Fehler während des Entpacken und gegeben Sie im Falle eines Fehler eine **sinnvolle** Fehlermeldung. Sollte das Entpacken erfolgreich sein, geben Sie den nun entpackten String aus.

```
String first  = "4a5v2c1s9i";
String second = "6y7v3k9m";
String third  = "0s4b";
String fourth = "1a1b1c1d";
```

   Sie dürfen zum Lösen keine Pakete importieren oder Funktionalitäten verwenden, die einen Import erfordern, damit das Programm kompiliert. Das Programm muss erkennbar ein Javaprogramm sein, dass geeignet ist das gestellte Problem zu lösen ansonsten wird es mit 0 Punkten bewertet.

Lösung:

Solution:

```
public static String decompress(String compressed) {
    String ret = "";
    for(int i = 0; i < compressed.length(); i += 2) {
        char cnumber = compressed.charAt(i);
        int number = 0;
        char c = compressed.charAt(i + 1);
        switch(cnumber) {
            case '0':
                throw new IllegalArgumentException("0 is a
                    disallowed compression number");
            case '1':
                number = 1;
                break;
            case '2':
                number = 2;
                break;
            case '3':
                number = 3;
                break;
            case '4':
                number = 4;
                break;
            case '5':
                number = 5;
                break;
            case '6':
                number = 6;
                break;
            case '7':
                number = 7;
                break;
            case '8':
                number = 8;
                break;
            case '9':
                number = 9;
                break;
        }
        for(int p = 0; p != number; ++p) {
            ret += c;
        }
    }
    return ret;
}

public static void main(String arg[]) {
    String first  = "4a5v2c1s9i";
    String second = "6y7v3k9m";
    String third  = "0s4b";
    String fourth = "1a1b1c1d";
    String s = ""
```

```
    try {
        switch(arg[0]) {
            case "0":
                s = first;
                break;
            case "1":
                s = second;
                break;
            case "2":
                s = third;
                break;
            case "3":
                s = fourth;
                break;
            default:
                throw new IllegalArgumentException("You have
                    to input a value between 0 and 3");
        }
        System.out.println(decompress(s));
    } catch(IllegalArgumentException e) {
        System.out.println(e);
    }
}
```