Probeklausur

- Für die Studiengänge BI und BWI -

Vom Prüfer auszufüllen:

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Σ**
Maximale Punkte	13	10	11	5	17	14	14	16	20	120
Erreichte Punkte										

⚠ Achtung

Bitte gehen Sie nicht davon aus, dass die Aufgaben in identischer Form in der Klausur vorkommen. Die Probeklausur dient ausschließlich der Veranschaulichung.

Bei der Bearbeitung aller Aufgaben sind folgende Regeln einzuhalten:

- **Code Conventions**: Halten Sie sich in Ihrem Code an die in der Lehrveranstaltung vorgestellten Java Code Conventions.
- **Kommentare sparsam einsetzen**: Kommentare sollten nur verwendet werden, wenn sie zur Verständlichkeit notwendig sind. Übermäßiges oder unnötiges Kommentieren sollten Sie aus Zeitgründen vermeiden.
- **Attribute privat deklarieren**: Sämtliche Instanzattribute sind als `private` zu definieren, sofern die Aufgabenstellung keine abweichende Sichtbarkeit verlangt.
- **Nur bekannte Konstrukte verwenden**: Es dürfen ausschließlich Sprachkonstrukte, Bibliotheken und Konzepte genutzt werden, die in der Vorlesung oder in den zugehörigen Übungen behandelt wurden.
- **Import Statements** Vernachlässigen Sie in Ihrem Code Import Statements.
- **Herleitung deutlich machen:** Machen Sie Ihre Ergebnisse bei Berechnungen durch Zwischenschritte deutlich. Für das reine Endergebnis gibt es **keine Punkte**

Viel Erfolg bei der Klausur!

Aufgabe 1: Interne Zahlendarstellung (6+3+4 Punkte)

a) Überführen Sie die Zahl `-148,7` in IEEE-754 32-Bit Darstellung. Geben Sie das Endergebnis in hexadezimaler Darstellung an.

b) Geben Sie zum Wert −17 die Zweierkomplementdarstellung an, die mit möglichst wenig Ziffern auskommt.

c) Bestimmen Sie die Ausgabe der folgenden Zeile. Machen Sie Ihre Herleitung durch Zwischenschritte deutlich.

```
System.out.println(0b100 + 024 + 0x2f + 7);
```

Mögliche Lösung:

a) Vorzeichenbit: - $\rightarrow 1$
Vorkomma: $148_{10} \rightarrow 10010100_2$
Nachkomma: 0.7_{10} :

$$
\begin{aligned}
0,7 * 2 = 1,4 &\rightarrow 1 \\
0,4 * 2 = 0,8 &\rightarrow 0 \\
0,8 * 2 = 1,6 &\rightarrow 1 \\
0,6 * 2 = 1,2 &\rightarrow 1 \\
0,2 * 2 = 0,4 &\rightarrow 0 \\
0,4 * 2 = 0,8 &\rightarrow 0 \\
&\dots \\
&\rightarrow 1011001100\dots
\end{aligned}
$$

Zusammen: $10010100,1011001100\dots_2 = 1,00101001011001100\dots_2 * 2^7$
Exzess-127: $127_{10} + 7_{10} = 134_{10} = 128_{10} + 4_{10} + 2_{10} \Leftrightarrow 10000110_2$
Gesamtdarstellung: $11000011000101001011001100110011_2 \Leftrightarrow C314B333_{16}$

- Bewertungshinweise:
  - Vorkomma korrekt (1P)
  - Nachkomma korrekt (2P)
  - Exponent korrekt (1P)
  - Gesamtdarstellung (1P)
  - Hexadezimale Darstellung (1P)

b) $-17_{10} = -32_{10} + 15_{10} \rightarrow 100000_2 + 1111_2 \rightarrow 101111_2^*$
- Bewertungshinweise:
  - Darstellung als Summe der nächst größeren Zweierpotenz (1P)
  - Überführung in das Binärsystem (1P)
  - Korrekte Darstellung als Zweierkomplement (1P)
  - Berechnung mit dem Variante über das Einerkomplement ist auch okay.
  - Wenn die Darstellung zusätzliche 1en beinhaltet (-1P)

c) $0b100 : 100_2 \rightarrow 4_{10}$
$024 : 24_8 \rightarrow 4 * 8^0 + 2 * 8^1 = 4_{10} + 16_{10} = 20_{10}$
$0x2f : 2f_{16} \rightarrow 15 * 16^0 + 2 * 16^1 = 15_{10} + 32_{10} = 47_{10}$
$7 : 7_{10}$

Summe: $4 + 20 + 47 + 7 = 78$

- Bewertungshinweise:
  - Binär korrekt (1P)

- Oktal korrekt (1P)
- Hexadezimal korrekt (1P)
- Summe korrekt (1P)

Aufgabe 2: Grammatik (2+6+2 Punkte)

Gegeben sei das Alphabet $\Sigma = \{a, b, c\}$ und die Sprache $L = \{a^n bc^m \mid n, m \in \mathbb{N}_0\}$

a) Beschreiben Sie mit eigenen Worten, wie die Wörter allgemein aussehen.

b) Geben Sie eine **vollständige** Typ3 Grammatik G an, die genau diese Sprache erzeugt.

c) Leiten Sie das Wort `abc` schrittweise aus Ihrer Grammatik ab.

Mögliche Lösung:

a) Alle Wörter, die aus beliebig vielen a's, gefolgt von genau einem b, gefolgt von beliebig vielen c's besteht.

b)

$$G = \{\Sigma, N, S, P\} \text{ mit :}$$
$$\Sigma = \{a, b, c\}$$
$$N = \{S, C\}$$
$$S = S$$
$$P := \begin{cases} S \to \text{aS}|\text{bC} \\ C \to \varepsilon|\text{cC} \end{cases}$$

c) $S \vdash \text{aS} \vdash \text{abC} \vdash \text{abcC} \vdash \text{abc}$

- Bewertungshinweise
  - Teil a): Sprache beschreiben (2P)
  - Teil b):
    - Grammatik vollständig angegeben (2P)
    - Korrekte *Typ3* Grammatik (2P)
    - Wörter können korrekt hergeleitet werden (2P)
  - Teil c): Ableitung angegeben (2P)
  - Für alle Unterpunkte in dieser Aufgabe gilt: Alles korrekt 2P; ein Fehler 1P; sonst 0P

Aufgabe 3: Berechnung der verbleibenden Downloadzeit (11 Punkte)

Implementieren Sie die folgende statische Methode in Java und beachten Sie dabei folgende Hinweise:

- Die Dateigröße wird in Gigabyte (GB) übergeben.
- Die Datenrate wird in Megabit pro Sekunde (MBit/s) übergeben.
- Beachten Sie, dass:
- Die Berechnung der Zeit erfolgt auf Basis einer konstanten Datenrate.
- Runden Sie die berechnete Zeit nicht, sondern schneiden Sie Nachkommastellen ab.
- Sie dürfen davon ausgehen, dass beide Parameter größer als 0 sind.

```
/**
 * Bestimmt die verbleibende Zeit einer Datenübertragung
  * @param dateigroesse in GB
  * @param datenrate in MBit/s
  * @return Verbleibende Zeit in Stunden:Minuten:Sekunden
  *
  */
public static String restZeit(double dateigroesse, double datenrate) {
}
```

Beispiel

Der Aufruf: `System.out.println(restZeit(2, 10));`
Soll folgende Ausgabe liefern: 0:26:40

Der Download einer Datei mit einer Größe von 2GB und einer Datenrate von 10 MBit/s benötigt 26 Minuten und 40 Sekunden.

Info

$1\text{GB} = 10^9$ Byte
$1\text{MB} = 10^6$ Byte

Mögliche Lösung:

```
/**
 * Bestimmt die verbleibende Zeit eines Downloads
 * @param dateigroesse in GB
 * @param datenrate in MBit/s
 * @return Verbleibende Zeit in hh:mm:ss
 *
 */
public static String restZeit(double dateigroesse, double datenrate) {
     dateigroesse *= 1000; //Umrechnung in MB
     dateigroesse *= 8; //Umrechnung in MBit
     int sekunden = (int)(dateigroesse/datenrate);
     int minuten = sekunden/60;
     sekunden %= 60;
     int stunden = minuten/60;
     minuten %= 60;
     return stunden + ":" + minuten + ":" + sekunden;
}
```

- Bewertungshinweise:
  - Korrekte Umrechnung der Dateigröße von GB in Bit (2P)
  - Korrekte Berücksichtigung der Datenrate in MBit/s (1P)

- Korrekte Berechnung der Gesamtdauer in Sekunden (2P)
- Korrekte Zerlegung der Zeit in Stunden, Minuten und Sekunden (3P)
- Korrekte Darstellung der Rückgabe im Format hh:mm:ss (2P)
- Der String muss nicht zwingend zweistellig formatiert sein (z. B. 1:5:9 ist zulässig)
- Nachkommastellen werden korrekt abgeschnitten und nicht gerundet (1P)
- Kleinere Rechenungenauigkeiten bei ansonsten korrektem Lösungsweg (-1P)
- Alternative Rechenwege sind möglich.

Aufgabe 4: Code Analyse (5 Punkte)

Sie sollen eine Funktion entwickeln, um Potenzen in Java zu bestimmen. Im Rahmen Ihres Entwicklungsprozess haben Sie mit Hilfe einer generativen KI immer weiter versucht, Ihren Code kompakter zu gestalten. Am Ende sah der Code wie folgt aus:

```
public static double pow(double base, double exp){
  return base^exp;
}
```

Der Code lässt sich kompilieren, die Ergebnisse weichen aber von Ihrer Erwartung ab.

1. Wo ist der Fehler im Quellcode?
2. Welchen Wert berechnet die Funktion für den Aufruf: `pow(2,3)`?

Mögliche Lösung:

1. Bei dem ^-Operator handelt es sich um das Bitweise XOR.
2. 2_{10} XOR $3_{10} \rightarrow 01_2$ XOR $11_2 = 01_2 \rightarrow 1_{10}$

- Bewertungshinweise:
  - Fehler korrekt identifiziert (2P)
  - Bestimmung der Ausgabe korrekt (3P)

Aufgabe 5: Euler-Zahl e (9+8 Punkte)

Die Euler'sche Zahl e hat folgende Reihendarstellung:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \ldots$$

a) Schreiben Sie zunächst eine rekursive Funktion, welche die Fakultät einer natürlichen Zahl berechnet.

Vorgaben:
- Die Implementierung muss rekursiv erfolgen.
- Schleifen (for, while, do-while) sind nicht erlaubt.
- Es gilt: $n \in \mathbb{N}_0$
- Sie dürfen davon ausgehen, dass keine Überläufe auftreten.

b) Schreiben Sie ein Java-Programm mit dem Namen `Euler`, das den Wert der obigen Summe berechnet. Der Parameter k gibt an, bis zu welchem Summanden die Reihe berechnet wird, und wird dem Programm **über die Kommandozeile** übergeben.

Vorgaben:
- Rufen Sie die Funktion aus Aufgabe a) auf. Auch wenn Sie keine Lösung erarbeitet haben.
- Der Parameter k ist eine natürliche Zahl.
- Das Programm soll korrekt terminieren und ein numerisches Ergebnis ausgeben.

Info

Die Fakultätsfunktion ist wie folgt definiert:

$$n! := \prod_{k=1}^{n} k = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

$$0! = 1$$

Mögliche Lösung:

a)

```
public static int fac(int n) {
  if(n == 0 || n == 1) return 1;
  return n*fac(n-1);
}
```

- Bewertungshinweise:
  - Korrekte Signatur (2P)
  - Korrekte rekursive Implementierung der Fakultätsfunktion (3P)
  - Korrekte Wahl des Rekursionsankers (n = 0 bzw. n = 1) (2P)
  - Korrekte rekursive Reduktion (n ∗ fac(n-1)) (2P)

b)

```
public class Euler {
  public static void main(String[] args) {
    int k = Integer.parseInt(args[0]);
    double sum = 0;
    for(int i = 0; i <= k; i++) {
      sum += 1.0/fac(i);
    }
    System.out.println(sum);
```

```
    }
}
```

- Bewertungshinweise:
  - Verwendung der rekursiven Fakultätsfunktion in der Summenberechnung (2P)
  - Korrekte Berechnung der Summe von i = 0 bis k (2P)
  - Korrekte Verwendung von Gleitkommadivision (1.0 / fac(i)) (2P)
  - Korrekte Verarbeitung des Kommandozeilenparameters (1P)
  - Ausgabe des berechneten Ergebnisses (1P)

Aufgabe 6: Farbbild zu Graustufen (14 Punkte)

In der Vorlesung haben wir gelernt, dass man einen 8-Bit Farbwert in einem int-Wert codieren kann. Hierbei repräsentieren das 0. bis 7. Bit den Blauwert, das 8. bis 15. Bit den Grünwert und das 16. bis 23. Bit den Rotwert.

Man kann solche Farbwerte in Graustufen umwandeln, z.B. um ein Farbbild in schwarzweiß auszudrucken. Für einen einzelnen Farbwert bestehend aus (rot, gruen, blau) kann man das mit der Formel grau=(rot + gruen + blau)/3 machen. Dieser Grauwert muss dann entsprechend an alle Stellen der Farbwerte zurückgeschrieben werden, so dass der Farbwert dann (grau, grau, grau) ist.

Schreiben Sie eine Klassenmethode colorToGray, die ein Bild annimmt (welchen Datentyp hat das Bild?) und alle Pixelwerte in-place in Grauwerte umwandelt.

Info

- Beachten Sie, dass Sie für die Berechnung des Grauwerts die Bits der Werte für Rot, Grün und Blau in geeigneter Weise aus der int-Zahl isolieren müssen.
- Beachten Sie, wie Sie den Grauwert an die entsprechenden Stellen der Bits zurückschreiben müssen.
- Beachten Sie außerdem, dass ein Bild nicht nur quadratisch sein kann, sondern auch im Format 16:9, also rechteckig

Mögliche Lösung:

```
public class GrayFilterArray {

  /**
   * Wandelt Farbwerte in einem Array in Graustufen vor, ohne alpha-Kanal
   * @param das Bild mit Farbwerten
   *
   */
  public static void colorToGray(int[][] img) {
    int height = img.length;
    int width  = img[0].length;

    for (int y = 0; y < height; y++) {
      for (int x = 0; x < width; x++) {

        int rgb = img[y][x];

        int r = (rgb >> 16) & 0xff;
        int g = (rgb >> 8)  & 0xff;
        int b =  rgb        & 0xff;

        int gray = (r + g + b) / 3;

        int newRgb = (gray << 16) | (gray << 8) | gray;
        img[y][x] = newRgb;
      }
    }
  }
}
```

- Bewertungshinweise:
  - Korrekte Bilddimensionen (2P)
  - Korrekte Geschachtelte Schleife (2P)
  - Korrekte Zugriff auf pixel an (x,y) (1P)

- Korrektes auslesen der einzelnen Farbwerte mit Bitshift und Bitmaske (4P)
- Korrekte Umwandlung in Grauwert (2P)
- Grauwert korrekt an alle Stellen des int Werts mit Bitoperationen zurückgeschrieben (3P)

Aufgabe 7: Exception (5+9 Punkte)

Gegeben ist folgender Programmcode:

```
import java.util.Scanner;

public class Demo {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int dividend = sc.nextInt();
    int divisor = sc.nextInt();
    System.out.println(dividend/divisor);
    sc.close();
  }
}
```

Sie sollen diesen Code durch den Einsatz von Exceptions sicherer machen. Im oben genannten Code können viele Fehler auftreten. Wird dem Programm keine gültige Zahl übergeben, kommt es zu einer `InputMismatchException`. Wird für den `divisor` der Wert 0 eingelesen, so wird eine `ArithmeticException` geworfen.

a) Entwickeln Sie eine eigene `unchecked` Exception `UngueltigeWerteException`. Halten Sie sich an die in der Vorlesung vorgestellten „best practices“.

b) Fangen Sie die beiden Exceptions, die potenziell geworfen werden und lösen Sie stattdessen eine `UngueltigeWerteException` aus. Sorgen Sie dafür, dass der Scanner weiterhin korrekt geschlossen wird.

Mögliche Lösung:

a)

```
public class UngueltigeWerteException extends RuntimeException{
  public UngueltigeWerteException() {
    super("Ungültige Werte eingegeben");
  }

  public UngueltigeWerteException(String msg) {
    super(msg);
  }
}
```

- Bewertungshinweise:
  - Ableitung der eigenen Exception von RuntimeException (unchecked) (2P)
  - Implementierung eines parameterlosen Konstruktors (1P)
  - Implementierung eines Konstruktors mit Fehlermeldung (String) (1P)
  - Verwendung von super(...) in allen Konstruktoren (1P)

b)

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ExceptionAufgabe {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    try {
      int dividend = sc.nextInt();
      int divisor = sc.nextInt();
      System.out.println(dividend / divisor);
```

```
    } catch (InputMismatchException e) {
      throw new UngueltigeWerteException("Ungültige Eingabe: Keine ganze Zahl");
    } catch (ArithmeticException e) {
      throw new UngueltigeWerteException("Ungültige Eingabe: Division durch 0");
    } finally {
      sc.close();
    }
  }
}
```

- Bewertungshinweise:
  - Korrektes Abfangen der InputMismatchException (2P)
  - Korrektes Abfangen der ArithmeticException (2P)
  - Werfen einer IllegalArgumentException anstelle der ursprünglichen Exceptions (2P)
  - Sicherstellen, dass der Scanner unabhängig vom Fehlerfall geschlossen wird (finally oder äquivalent) (2P)
  - Verwendung spezifischer catch-Blöcke und kein pauschales catch(Exception) (1P)

Aufgabe 8: Statische und dynamische Bindung (12+2+2 Punkte)

Gegeben ist folgendes Programm:

```
interface IFace {
    public void f(float g);
}


class K1 implements IFace {

    public void f(float g) {
        System.out.print("K1-f , ");
        g();
    }

    void g() {
        System.out.print("K1-g , ");
        h();
    }

    static void h() {
        System.out.println("K1-h ");
    }
}


public class K1Test {

    public static void main(String[] args) {
        K1 k1 = new K1();
        k1.f(1.0F);

        K2 k2 = new K2();
        k2.f(2.0F);

        ((K1) k2).f(3.0F);

        IFace ik = (IFace) k2;
        ik.f(4.0F);
    }
}
```

```
class K2 extends K1 {

  public void f(float g) {
      System.out.print("K2-f , ");
      g();
  }

  static void h() {
      System.out.println("K2-h ");
  }

  void m() {
  }
}
```

a) Was ist die Ausgabe des Programms?
b) Was bewirkt es, wenn man am Ende von `main` die Zeile einfügt: `((K1)k2).m();`
c) Was bewirkt es, wenn man am Ende von `main` die Zeile einfügt: `((K2)k1).m();`

Mögliche Lösung:

a) K1-f, K1-g, k1-h (3P)
K2-f, K1-g, k1-h (3P)
K2-f, K1-g, k1-h (3P)
K2-f, K1-g, k1-h (3P)

b) Es kommt zu einem Compilerfehler, da die Sichtbarkeit eingeschränkt wurde. (2P)

c) Es kommt zu einer ClassCastException, da das K1 Objekt intern kein K2 Objekt ist. (2P)

Aufgabe 9: Modellierung eines Druckerraums (20 Punkte)

Sie sollen einen Raum mit Druckern in Software modellieren. Jeder Drucker hat ein Netzteil, das entweder auf 220V oder 110V eingestellt ist. Ein neuer Drucker ist auf 220V eingestellt. Man kann jeden existierenden Drucker einschalten und ausschalten sowie ein Blatt drucken. Es wird allerdings nur etwas ausgedruckt, wenn der Drucker eingeschaltet ist.

Es gibt zwei Druckertypen:

- Tintenstrahldrucker haben einen Tintentank, dessen Füllhöhe (in %) man abfragen kann. Die Füllhöhe des Tanks verringert sich mit dem Druck eines jeden Blatts um 1 Prozentpunkt.
- Laserdrucker haben eine Trommel, die nach 1000 gedruckten Blättern ausgetauscht werden muss. Dazu kennt jeder Laserdrucker eine Aktion austauschen (den Methodenrumpf leer lassen), die automatisch nach dem Druck des 1000. Blatts aktiviert wird.

Bilden Sie obige Druckerbeschreibungen geeignet in Java ab. Alle Aktivitäten sollen dabei als Methode reali siert werden (z.B. das Einschalten und das Ausschalten). Sind für eine Aktion keine weiteren Details bekannt, so lassen Sie den Methodenrumpf ihrer Java-Methode leer. Beispiel: Die Methode austauschen() beim Laserdrucker. Erzeugen Sie in ihrem Druckerraum insgesamt 3 Drucker, einen Tintenstrahldrucker und 2 Laserdrucker. Fragen Sie anschließend in ihrem Programm die Füllhöhe der Tanks des Tintenstrahldruckers ab und lassen auf einem Laserdrucker und dem Tintelstrahldrucker ein Blatt ausdrucken.

Mögliche Lösung:

```
/**Test Druckerraum
* @author Rudolf Berrendorf
* @version 1.0
*/
public class Druckerraum{
  public static void main(String[] args){
    Tintenstrahldrucker t = new Tintenstrahldrucker();

    Laserdrucker l1 = new Laserdrucker();
    Laserdrucker l2 = new Laserdrucker();

    System.out.println(t.getfuellhoehe());

    t.drucken();
    l1.drucken();
  }
}

public class Drucker{
  pivate int netzteil;
  private boolean eingeschaltet;

  public Drucker() {
    netzteil=220;
    eingeschaltet=false;
  }

  public void einschalten(){
    eingeschaltet=true;
  }

  public void ausschalten(){
    eingeschaltet=false;
  }
}

public class Tintenstrahldrucker extends Drucker{
  int fuellhoehe;

  public Tintenstrahldrucker(){
    super();
    fuellhoehe=100;
  }

  public int getfuellhoehe(){
    return fuellhoehe;
  }

  public void drucken(){
    fuellhoehe--;
  }
}

public class Laserdrucker extends Drucker{
  private int gedruckt;
```

```
  public Laserdrucker(){
    super();
    gedruckt=0;
  }

  public void drucken(){
    ++gedruckt;
    if(gedruckt>=1000){
      austauschen();
    }
  }

  public void austauschen(){}
}
```

- Bewertungshinweise:
  - Korrekte Vererbung (2P)
  - Instanzvariablen sinnvoll in Klassen (3P)
  - Sinnvolle Zugriffsrechte (2P)
  - Konstruktoren korrekt, inkl Variableninitialisierung (3P)
  - Methoden korrekt und sinnvoll in Klassen (7P)
  - main-Methode korrekt (3P)