CLUB APOLLO 13, 15. Wettbewerb
Aufgabe 1

„Saying ‚Hello World' in Java!"

Diese Aufgabe wird vom Fachgebiet Software Engineering an der Fakultät für Elektrotechnik und Informatik der Leibniz Universität Hannover gestellt.

Weitere Informationen zum Studiengang der Informatik und zum Fachgebiet findet ihr unter http://www.et-inf.uni-hannover.de und unter http://www.se.uni-hannover.de.

```java
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

In dieser Aufgabe wollen wir uns mit dem Programmieren in der Programmiersprache Java beschäftigen. Aber keine Angst, wenn ihr noch nicht programmieren könnt. Alles Wichtige lernt ihr hier.

Um ein Programm schreiben zu können, braucht man zunächst einen Editor und einen so genannten Compiler. Mit dem Editor schreibt man das Programm in einer für den Menschen verständlichen Sprache (Quellcode) und der Compiler übersetzt es dann in die Computersprache.

Für die Programmierung in Java bietet sich die Software *eclipse* an. Diese stellt gleichzeitig Editor und Compiler zur Verfügung. Ihr könnt sie kostenlos unter https://www.eclipse.org herunterladen.
Wenn ihr keine neue Software installieren wollt, dann könnt ihr auch den Onlinecompiler auf der Website http://www.compilejava.net verwenden. Dort müsst ihr lediglich den jeweiligen Quellcode einfügen und dann auf „Compile & Execute" klicken.
Ihr könnt aber auch andere Programme verwenden; die Bewertung ist unabhängig von der Wahl der Software. Ausschlaggebend ist lediglich der abgegebene Java-Programmcode.

Die Aufgaben

a) Grundlagen und Code-Konventionen (10 Punkte)

Programme bestehen aus unterschiedlichen „Bausteinen": Variablen, Anweisungen, Zuweisungen, Verzweigungen, Schleifen usw. Mit diesen Bestandteilen werdet ihr euch im ersten Aufgabenteil beschäftigen.

Bei Variablen unterscheidet man zwischen verschiedenen Datentypen für Zeichenketten, Zahlen und einigen speziellen Typen wie Feldern und Listen.

Einer Variablen könnt ihr in Java über folgende Vorschrift einen bestimmten Datentyp zuweisen:
Der Befehl `String wort` erzeugt beispielsweise eine Variable `wort`, die eine Zeichenkette speichern kann. Über den Befehl `wort = "Haus"` könnt ihr der Variablen das Wort `Haus` zuordnen.

1) Welche Datentypen gibt es in Java für Zahlen? Gebt drei Datentypen und jeweils ein Beispiel für eine Zahl an, die dem jeweiligen Datentyp entspricht.
   (Bsp.: Zeichenkette; String; „Haus")

2) Worin besteht der Unterschied zwischen `String wort = "7"` und `int wort = 7`? Wie wirkt sich die Änderung des Datentyps auf die Verarbeitung der Variablen aus?

Nun wollen wir uns die grundlegende Struktur eines kleinen Programms anschauen, die wir in den folgenden Aufgaben mit Leben füllen werden.

```
// Ein Kommentar wird durch zwei Schraegstriche am Anfang der Zeile gekennzeichnet.
// Alles, was hinter den zwei Strichen steht, wird beim Kompilieren und beim
// Ausfuehren des Programms ignoriert.
// Am Anfang des Programms steht die Deklaration und der Name der Klasse.
// Das Attribut 'public' sginalisiert, dass es sich um eine oeffentliche Klasse
// handelt. Daneben gibt es noch 'private' und 'protected'.

public class Klassenname {

    // Die main-Methode ist die Methode, die ausgefuehrt wird, wenn das Programm
    // gestartet wird.
    // 'void' signalisiert, dass es keinen Rueckgabewert gibt.
    public static void main(String[] args) {

        // Hier deklarieren wir die Variablen, die wir benoetigen.
        int zahl;
        String text;

        // An dieser Stelle weisen wir der Variablen zahl die Zahl 7 zu.
        zahl = 7;

        // Dies ist eine Verzweigung.
        // Zunaechst wird die Bedingung in den runden Klammern ueberprueft und dann
        // die entsprechende Anweisung ausgefuehrt.
        if (zahl == 7) {
            text = "a.";
        }
        else {
            text = "b.";
        }

        // Dies ist eine for-Schleife.
        // Sie faengt bei 0 an zu zaehlen, erhoeht den Wert von i immer um 1 und
        // wiederholt die Schleife dann so oft, bis i = 10 erreicht ist. Dabei wird
        // in jedem Schritt in der darunter stehenden Anweisung fuer i die aktuelle
        // Zahl eingesetzt.
        for (int i = 0; i <= 10; i++) {
            zahl = zahl + i;
        }

        // Diese Zeile ist fuer die Ausgabe zustaendig. Der Wert von 'text' und
        // 'zahl' wird in der Kommandozeile ausgegeben.
        System.out.println(text);
        System.out.println(zahl);
    }
}
```

Beim Programmieren gibt es Code-Konventionen, die genau vorgeben, wie Variablen zu benennen sind, welche Tabulatoren zur besseren Gesamtstruktur des Programms zu verwenden sind und ähnliches. Ein ausführliches Beispiel zu den Code-Konventionen von Oracle, dem Hersteller von Java, findet ihr unter http://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

In dem folgenden Programmcode wurden diese Code-Konventionen vernachlässigt.

```
public class name {
public static void main(String[] args) {
int temp1; String temp2; temp1 = 7;
if (temp1%2 == 0) {
temp2 = "Gerade.";}
else {
temp2 = "Ungerade.";}
System.out.println(temp2);}}
```

3) Verbessert den Quellcode, indem ihr
   a. die Variablen angemessen benennt.
   b. die Ausgabe so modifiziert, dass erkennbar ist, was das Programm macht.
   c. die Einschübe ergänzt.

4) Schreibt nun ein kleines Programm, das euren Gruppennamen in folgender Form in der Kommandozeile ausgibt:
   `Die Gruppe [Gruppenname] besteht aus [eure Namen].`
   Dabei sollen der Gruppenname und eure Namen in Variablen stehen.

b) Code-Analyse und JavaDoc (10 Punkte)

Neben dem eigenständigen Schreiben von Programmcode ist auch das Verstehen und Nachvollziehen von bereits geschriebenem Quellcode nicht zu unterschätzen.
Bei größeren Softwareprojekten nutzt man *JavaDoc* zur Dokumentation. Dabei handelt es sich auf den ersten Blick um Kommentare im Quellcode, die über bestimmte Attribute (bspw. „author“) verfügen. Aus den JavaDoc-Kommentaren wird dann eine HTML-Datei erzeugt, die das komplette Softwareprojekt dokumentiert.

1) Wie werden JavaDoc-Kommentare im Quellcode gekennzeichnet? Nennt auch drei wichtige Tags mit den Parametern.

Wir wollen nun folgenden Java-Quellcode auf seine Funktionsweise hin untersuchen.

```
// i)
public class Test    {
    public static void main(String[] args)   {
        // ii)
        int g = 10000;
        int v = 0;
        double x,y;
        double output;
        // iii)
        for (int i = 1; i <= g; i++)    {
        x = Math.random();
        y = Math.random();
        if (Math.hypot(x,y) <= 1)
            v++;
        }
        // iv)
        output = 4*(double)v / g;
        System.out.printf("Output: %g%n",output);
    }
}
```

2) Beschreibt in wenigen Sätzen die einzelnen Schritte des Algorithmus. Ergänzt dazu die Kommentare an den Stellen i) – iv) mit JavaDoc.

Java stellt viele Funktionen bereits in verschiedenen Packages zur Verfügung. Diese sind in der Java-API dokumentiert.

3) Recherchiert die Funktionsweise der drei Methoden:

I. `System.out.printf()`
II. `Math.random()`
III. `Math.hypot(x, y)`

Gebt dabei auch die Beschreibung der zweiten und dritten Methode aus der Java-API an. Welche Werte darf man für `x` und `y` in der Methode einsetzen?

4) Führt das Programm für verschiedene Werte von g aus. Was bewirkt eine Änderung von g?

5) Welcher Zahl nähert sich die Gleitkommazahl `output` am Ende des Algorithmus an und warum? Begründet eure Antwort mathematisch.

6) Wie heißt der Algorithmus und wofür wird er verwendet?

7) Wir wollen die Abbruchbedingung in der `for`-Schleife nun verändern und nicht eine bestimmte Anzahl an Iterationen betrachten, sondern das Ergebnis auf eine bestimmte Genauigkeit (0.000001) annähern. Wie sieht der neue Quellcode aus?

c) Sortieralgorithmen (10 Punkte)

Algorithmen kann man auf unterschiedlichste Arten darstellen, bevor man sie zum Schluss in Programmcode umwandelt.

1) Nennt und erklärt in jeweils einem Satz drei verschiedene Möglichkeiten, Algorithmen aufzuschreiben oder darzustellen.

Um eine Liste von verschiedenen Zahlen zu sortieren, gibt es viele verschiedene Algorithmen. Im Folgenden sollt ihr euch mit zwei dieser Sortieralgorithmen näher beschäftigen.

2) Beschreibt kurz die Funktionsweise des *Quicksort*-Algorithmus. Nutzt dabei zwei der drei Möglichkeiten, die ihr in Aufgabe c.1) genannt habt.
Hinweis: Wenn ihr bei Aufgabe c.1) zu keiner Lösung gekommen seid, könnt ihr die Funktionsweise in Worten beschreiben.

Zusätzlich betrachten wir nun folgendes Sortierverfahren:

```
Input: Liste mit ganzzahligen Einträgen.

Fange beim ersten Element der Liste (ganz links) an.
Solange das Ende der Liste noch nicht erreicht ist, mache Folgendes:
    Vergleiche das Element mit seinem rechten Nachbarn.
        Wenn das rechte Element kleiner ist, vertausche die beiden Elemente.
        Sonst gehe einen Schritt weiter nach rechts und wiederhole den Schritt.
Am Ende der Liste beginne wieder von vorne, bis keine Vertauschungen mehr notwendig
sind.

Output: sortierte Liste
```

3) Wie würde nach den beiden Algorithmen die Liste mit den Einträgen

5 – 7 – 4 – 8 – 1 – 15 – 3

sortiert werden?
Hinweis: Notiert beim Quicksort jeweils die einzelnen Teil-Listen am Ende jeden Schrittes. Bei dem anderen Algorithmus genügt es, wenn ihr die Liste nach jedem Vertauschen angebt.

4) Programmiert den oben beschriebenen Algorithmus. Fügt dabei an geeigneten Stellen Kommentare ein, die die einzelnen Schritte beschreiben.

Hinweise:

1. Es genügt, wenn euer Programm die oben angegebene Liste sortiert.
2. Die Ausgabe darf (und soll) über die Kommandozeile erfolgen.
3. Euch stehen `if`-Abfragen und `for`-Schleifen zur Verfügung.
4. Für die Ausgabe steht euch `System.out.print*` zur Verfügung.
5. Für die Datenspeicherung könnt ihr ein Array verwenden.

Viel Erfolg bei der ersten Aufgabe!

Allgemeine Hinweise

Einsendeschluss: Sonntag, 01. November 2015, 19:59 Uhr.

Gebt eure Lösungen über das Portal von uniKIK ab: http://www.unikik-portal.de/portal

Zulässige Dateiformate sind: PDF für die zusammengeschriebene Lösung (mit eingebetteten Bildern), sowie unter Windows gängige Videoformate, die sich ohne Installation von zusätzlicher Software abspielen lassen, z. B. mp4.
Die Dateien sollten nicht größer als 7,5 MB sein (die Dateien können gezippt sein)! Bitte gebt auch euren Teamnamen, die Namen der Gruppenmitglieder sowie deren Schulen an. Bitte benennt eure hochgeladenen Dateien nach dem Gruppennamen.

ACHTUNG bei Zip-Dateien! Um sicher zu gehen, dass eure Dateien wirklich fehlerfrei und für die Korrektoren/-innen zu öffnen sind, solltet ihr eure Zip-Dateien etc. noch mal von eurem Account herunterladen und öffnen. Dateien, die sich nicht öffnen lassen, können nicht bewertet werden!

Gebt eure Lösungen auch dann ab, wenn ihr nicht alle Fragen beantworten konntet! Vielleicht gelingt euch das ja bei den kommenden Aufgaben.

Die Teilnahmebedingungen und weitere Informationen findet ihr unter: http://www.unikik.de/apollo13.
Der Rechtsweg ist ausgeschlossen.