Prof. Dr. Ralf Hinze
Sebastian Schweizer, M.Sc.
Markus Heinrich, M.Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Programmiersprachen

Übungsblatt 1: Programmierpraktikum 2020

Ausgabe: 21. April 2020
Abgabe: 28. April 2020, 15 Uhr

Hinweis zum Template Auf der Website finden Sie ein Template als `.zip` Datei. Darin ist ein vollständiges Projekt für IntelliJ IDEA enthalten. Sie müssen die Datei auf Ihrem Rechner entpacken und dann den Ordner über den Button "Open or Import" öffnen. Wichtig ist dabei, dass Sie den Ordner auswählen, der die `.iml` Projektdatei enthält, nicht etwa einen übergeordneten Ordner oder die `.iml` Datei selbst.

Die zu implementierenden Methoden sind bereits vorgegeben, der Rumpf der Methode wirft eine Ausnahme. Sie müssen die Zeile `throw new RuntimeException("TODO");` durch Ihren Code ersetzen. In noch nicht implementierten Methoden sollten Sie diese Zeile stehen lassen, da sie sicherstellt, dass die Datei zusammen mit den Tests kompiliert. Außerdem ist für jede Aufgabe eine `main` Methode enthalten, die die zu implementierenden Methoden aufruft und Ausgaben auf die Konsole schreibt. Dies ist zur Fehlersuche gedacht. Sie können den Code der `main` Methode nach Belieben verändern.

Ein IntelliJ IDEA Projekt gibt die zu verwendende JDK Version vor. Im bereitgestellten Projekt haben wir das JDK Version 14 ausgewählt. Wenn Sie eine andere Version installiert haben, dann klicken Sie nach dem Importieren des Projekts auf File -> Project Structure und wählen Sie (wenn links Project ausgewählt ist) unter Project SDK eine installierte JDK Version aus. Sie können jedes JDK ab Version 11 verwenden.

Schauen Sie sich bitte die auf unserer Website verlinkten Videos zu IntelliJ IDEA an. Uns ist bewusst, dass die IDE sehr umfangreich ist und viele Funktionen bietet, die Sie für die Übungsblätter noch gar nicht benötigen. Sie sollen sich aber schon jetzt schrittweise mit der IDE vertraut machen, damit Ihnen in wenigen Wochen die Entwicklung der größeren Projekte deutlich leichter fällt.

Die im Projekt enthaltenen Testfälle können Sie benutzen, um Ihre Lösung vor dem Abgeben zu prüfen, ansonsten erfolgt die Prüfung aber auch durch das Exclaim System. Bestandene Testfälle sind keine Garantie für eine richtige Abgabe. Mehr zu JUnit Tests werden wir in den Projekten kennenlernen, hier werden die Dateien lediglich als zusätzlicher Service für Sie mit bereitgestellt.

Hinweis zur Abgabe Für jede Aufgabe enthält das bereitgestellte Projekt eine eigene Datei (`Aufgabe1.java` bis `Aufgabe3.java`, zu finden im `src` Ordner). Bearbeiten Sie die jeweilige Aufgabe nur in der entsprechenden Datei und laden Sie im Exclaim System *nur diese veränderte* Datei bei der jeweiligen Aufgabe hoch.

Eigenständiges Arbeiten / Plagiate Lösen Sie die Aufgaben bitte eigenständig und nur zusammen mit den Ihnen zugewiesenen Abgabepartner*innen. Sie dürfen gerne externe Materialien verwenden, müssen aber in ihrer Lösung die **Quelle** (Angabe des Fachbuchs, des Fachartikels oder der URL) und den **Umfang** benennen (haben Sie nur die Idee übernommen oder gar den kompletten Quelltext). Wenn sie im Internet recherchieren, bedenken Sie bitte, dass viele Seiten von zweifelhafter Qualität sind und die bloße Übernahme von Programmen nicht zum Lernerfolg beiträgt. Sollten Sie Code kopieren, ohne dabei die Quelle anzugeben, handelt es sich um ein **Plagiat**. Sollten Sie Code mit anderen Teams teilen, handelt es sich ebenfalls um ein Plagiat. **Bei Plagiaten werden alle beteiligten Teams vom Programmierpraktikum ausgeschlossen und können den Schein erst im nächsten Jahr erwerben!**

Aufgabe 1 Umformung Rekursion zu Schleifen (4 Punkte)

In der Vorlesung *Grundlagen der Programmierung* haben Sie bereits intensiv mit rekursiven Funktionen gearbeitet. Da in Java jedoch hauptsächlich mit Schleifen gearbeitet wird, soll folgende rekursive Methode in `Aufgabe1.java` umgeformt werden.

```
public static int magicRecursive(int n) {
    if (n <= 0) {
        return 0;
    }
    int recursiveResult = magicRecursive(n - 1);
    if ((n % 5 == 0) || (n % 7 == 0)) {
        return recursiveResult + n;
    } else {
        return recursiveResult;
    }
}
```

a) Implementieren Sie die Methode `magicFor`. Sie soll den obigen rekursiven Algorithmus iterativ (also ohne Rekursion) mit einer **`for`**-Schleife umsetzen. Es ist nicht erlaubt, die Methode `magicRecursive` (oder eine andere rekursive Methode) aufzurufen.

b) Implementieren Sie die Methode `magicWhile`. Sie soll den obigen rekursiven Algorithmus iterativ (also ohne Rekursion) mit einer **`while`**-Schleife umsetzen. Es ist nicht erlaubt, die Methode `magicRecursive` (oder eine andere rekursive Methode) aufzurufen.

Aufgabe 2 Arrays (8 Punkte)

a) Implementieren Sie die Methode **`int`** `getMin(`**`int`**`[] arr)`. Als Eingabe erhält die Methode ein nicht-leeres Array von Zahlen. Zurückgegeben werden soll die kleinste Zahl aus diesem Array. Diese Aufgabe soll mit Hilfe eines Schleifenkonstrukts gelöst werden; die Verwendung von Methoden aus der Standardbibliothek, durch welche Sie ohne Schleife auskommen würden, ist nicht erlaubt.

b) Implementieren Sie die Methode `String printArray(`**`int`**`[] arr)`. Als Eingabe erhält die Methode ein (möglicherweise leeres) Array von Zahlen. Zurückgegeben werden soll ein String, der dieses Array repräsentiert. Für das leere Array ist das der String `"[]"`. Für das Array aus den Zahlen 15, −5 und 27 ist das der String `"[15, -5, 27]"`. Auch hier sollen Sie eine Schleife benutzen; die Verwendung von `Array.toString` oder ähnlichem ist nicht gestattet.

c) Implementieren Sie die Methode **`int`**`[] mul(`**`int`**` n)`. Befolgen Sie dabei folgenden Algorithmus:

1. Erzeugen Sie drei Arrays (`a`, `b` und `c`) vom Typ **`int`**`[]`. Die Länge der Arrays soll `n` betragen.
2. Weisen Sie jedem Element in Array `a` als Wert seine Position im Array zu, sodass `a` wiefolgt aussieht: `[0, 1, 2, ..., n - 2, n - 1]`.
3. Weisen Sie den Elementen in Array `b` ihre Positionen in umgekehrter Reihenfolge zu, sodass `b` wiefolgt aussieht: `[n - 1, n - 2, ..., 2, 1, 0]`.
4. Führen Sie eine komponentenweise Multiplikation der Arrays `a` und `b` durch und speichern Sie das Ergebnis in `c`. Das `i`-te Element von `c` errechnet sich also aus der Multiplikation des `i`-ten Elements von `a` mit dem `i`-ten Element von `b`.
5. Geben Sie das Array `c` zurück.

Nehmen Sie keine Vereinfachungen vor und folgen Sie exakt dem vorgegebenen Algorithmus.

Aufgabe 3 Sieb des Eratosthenes (6 Punkte)

Mit dem *Sieb des Eratosthenes* können alle Primzahlen kleiner oder gleich einer vorgegebenen Zahl `n` ermittelt werden. Der Algorithmus basiert auf der Idee, dass ganzzahlige Vielfache von Primzahlen nicht prim sind. Wir gehen wie folgt vor:

1. Die Zahlen 2 bis n werden aufgeschrieben.
2. Wir starten bei der Zahl 2 und streichen alle Vielfachen (4, 6, 8, ...).
3. Wir gehen weiter zur Zahl 3. Diese ist noch nicht gestrichen, daher streichen wir alle Vielfachen (6, 9, 12, ...).
4. Die Zahl 4 ist bereits gestrichen, wir machen also nichts.
5. Die Zahl 5 ist noch nicht gestrichen, wir streichen also alle Vielfachen (10, 15, ...).

... Der Schritt wird wiederholt bis wir bei `n` angekommen sind.

Die nicht gestrichen Zahlen sind dann die Primzahlen kleiner oder gleich `n`.

Implementieren Sie die Methode `boolean[] sieb(int n)`. Das zurückgegebene Array gibt an, ob der jeweilige Index eine Primzahl ist. Für `n = 5` soll das Array `[false, false, true, true, false, true]` zurückgegeben werden. Die Indizes 0 und 1 sind hierbei für den Algorithmus nicht relevant. An den Indizes 2, 3 und 5 steht `true`, da diese Zahlen prim sind. An Index 4 steht `false`, da 4 keine Primzahl ist.