Studiengang: Mechatronik, Flug- und Fahrzeuginformatik

Prüfung
Grundlagen der Programmierung 2

Prüfer: Regensburger, Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Studiengang	Dozent	Matrikelnummer	Semester	Raum	Platz
FFI	Regensburger				

Aufgabe	1	2	3	4	Σ	Note
Punkte						

Bitte beachten:

Tragen Sie Ihre persönlichen Angaben auf dieses Deckblatt ein.

Dieses Geheft enthält sowohl die Aufgabenstellung als auch den Platz für Ihre Antworten.
Schreiben Sie möglichst nur in die vorgegebenen Antwortrahmen.

Geben Sie am Ende der Prüfung wieder alle Blätter ordentlich geheftet ab.
Reißen Sie auf keinen Fall einzelne Seiten heraus!

Viel Erfolg!

Aufgabe 1 (Verständnisfragen - ca. 10 %)

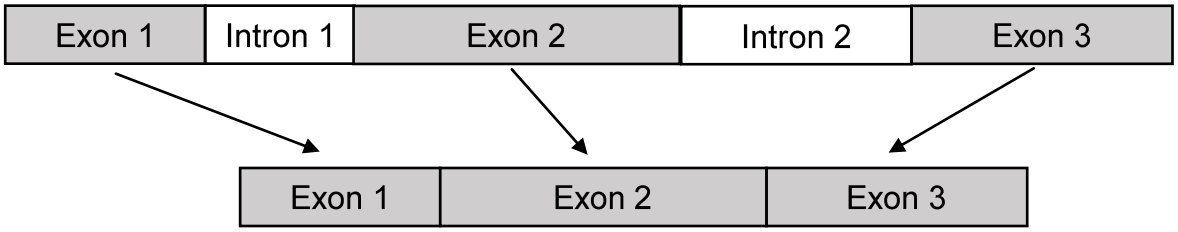
Welche Ausgaben erzeugt das folgende Programm?

```
class A {
    A() { System.out.println(this); };
    public String toString() { return "A"; }
}

class B extends A {
    public String tostring() { return "B"; }
}

class C extends B {
    public String toString() { return super.toString(); }
}

class Anwendung {
    public static void main(String[] args) {
        System.out.println("" + new A() + new B() + new C());
        String s = "Mechatronik", t = s.substring(5, 9);
        System.out.println(t == s.substring(5, 9));
        System.out.println(s != s + "");
        System.out.println(t);
        System.out.println(s.substring(0, s.indexOf(t)));
    }
}
```

Aufgabe 2: (Strings, ca. 20 %)

Im menschlichen Erbgut liegen die Gene gestückelt vor, d.h. ein Gen besteht aus mehreren DNA-Abschnitten, den *Exons*, die von nicht zum Gen gehörenden Abschnitten, den *Introns*, unterbrochen werden.

Die Zelle liest ein Gen aus der DNA ab, macht eine RNA-Kopie davon und schneidet dann alle Introns heraus. Dieser Vorgang, *Spleißen* genannt, ist im Folgenden schematisch dargestellt:



Exons und Introns bestehen aus einer Abfolge von *Nucleotiden*, dargestellt durch die Buchstaben A, C, G und U. Die Spleißmaschinerie erkennt Anfang und Ende eines Introns an bestimmten Nucleotidsequenzen.

Wir simulieren diesen Vorgang durch eine Methode *`spleissen`*. Der Methode werden drei Strings übergeben. Der erste Parameter stellt die ungespleißte RNA dar, der zweite Parameter die Nucleotidsequenz des Intron-Beginns, der dritte Parameter die Nucleotidsequenz des Intron-Endes. Die Funktion liefert die gespleißte RNA, die sogenannte *messenger RNA (mRNA)*, an den Aufrufer zurück.

Im folgenden Beispiel markiert die Sequenz `"GU"` den Beginn, die Sequenz `"AG"` das Ende eines Introns:

```
public static void main(String[] args) {
    System.out.println("mRNA: " +
        spleissen("AUAGUAAAAGCUCUGUUUAGGAGA", "GU", "AG"));
}
```

In diesem Beispiel würde das Programm `mRNA: AUACUCUGAGA` ausgeben.

Sie dürfen voraussetzen, dass bei Vorliegen eines Intron-Beginns auch das Intron-Ende in dem übergebenen RNA-Abschnitt vorhanden ist.

Ihre Aufgabe: Implementieren Sie die Methode *`spleissen`*.

Aufgabe 3 (Interfaces, Collections – ca. 40 %)

In vielen Anwendungen möchte man tabellarische Berichte *(Reports)* generieren. Datenbanken stellen dafür in der Regel *Reportgeneratoren* zur Verfügung. In dieser Aufgabe erstellen wir einen eigenen (kleinen) Reportgenerator.

Die Fähigkeit einer Klasse, einen Report zu liefern, signalisieren wir durch ein Interface:

```
import java.util.*;

interface Reportable {
    String                       getHeader();
    List<String>                 getLegend();
    Map <String, List<Integer>> getLines ();
}
```

Ein Report besteht aus einer *Überschrift*, einer *Legende* (d.h. den Spaltenüberschriften), und den eigentlichen, tabellarisch dargestellten *Daten* des Reports. Jede Zeile beginnt mit einem Schlüssel, dann folgen die dem Schlüssel zugeordneten Daten.

Das folgende Beispielprogramm zeigt, wie der Report einer Klasse `Statistics`, die das Interface `Reportable` implementiert, zeilenweise aufgebaut und dann ausgegeben wird:

```
class ReportGenerator {
    static void printReport(Reportable r)  { /* ... */ }

    public static void main(String[] args) {
        Statistics roboStat = new Statistics(
            "Zahl der Industrieroboter in Tausend",
            new String[] {"Land", "2010", "2011", "2012"} );
        roboStat.addLine("Japan", new int[] {308, 307, 311} );
        roboStat.addLine("USA"  , new int[] {180, 193, 207} );
        roboStat.addLine("D"    , new int[] {148, 157, 162} );
        printReport(roboStat);
    }
}
```

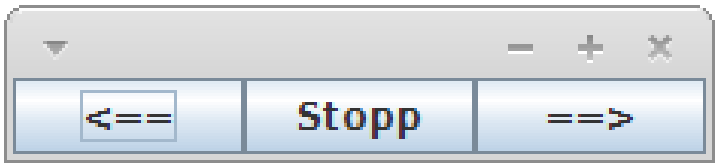
Dieses Programm erzeugt die folgende Ausgabe. Die erste Zeile ist die Überschrift, die zweite die Legende, dann kommen die Zeilen des Reports in alphabetischer Reihenfolge der Schlüssel, wobei die Spalten durch Tabulatorzeichen getrennt sind:

```
Zahl der Industrieroboter in Tausend
Land    2010    2011    2012
D       148     157     162
Japan   308     307     311
USA     180     193     207
```

Implementieren Sie die Klasse `Statistics` und die Methode `printReport`.

Aufgabe 4 (Graphik, Threads – ca. 30 %)

Implementieren Sie ein vollständiges Java-Programm, das folgende graphische Oberfläche mit drei Tasten zur Steuerung der Bewegung erzeugt, sowie die nachfolgend beschriebene Bewegungslogik realisiert:



Die linke bzw. rechte Taste setzen das anfangs stillstehende Fenster nach links bzw. rechts horizontal in Bewegung, mit einer gleichbleibenden Geschwindigkeit von 1 Pixel / 10 ms.
Die Stopp-Taste hält das Fenster an. Eine danach folgenden Betätigung der Pfeiltasten setzt das Fenster wieder in Bewegung.

Hinweise:

- Sie müssen keine `import`-Anweisungen angeben.
- Sie müssen keine Fehlerbehandlung für das Übertreten des linken bzw. rechten Bildrandes vorsehen.
- Folgende Methoden könnten für die Implementierung der Fensterbewegung hilfreich sein:

  Ein `JFrame` lässt sich mit `setLocation(x,y)` verschieben, die momentane Position lässt sich mit `getLocation().x` bzw. `getLocation().y` auslesen.

```
void java.awt.Window.setLocation(int x, int y)
Point java.awt.Component.getLocation()

void java.lang.Thread.sleep(long millis)
                throws InterruptedException

void java.lang.Thread.start()
```