Bioinformatikpraktikum
Wintersemester 2023/24
PD Dr. Jan Grau

Institut für Informatik
Universität Halle

Blatt 7

Aufgabe 7.1 Auf der Seite zur Veranstaltung im Stud.IP finden Sie für diese Übungsserie eine Reihe von Klassen, die als Basis für die Implementierungen dieses Übungsblattes dienen sollen.

(a) Implementieren Sie eine Unterklasse von `Alphabet` für ein DNA-Alphabet als Singleton. Beschränken Sie das DNA-Alphabet auf die Symbole A,C,G,T als Großbuchstaben.

(b) Implementieren Sie eine abstrakte Factory zur Erzeugung von Sequenzen. Die Basis-Klasse Ihrer Factory soll `AbstractSequenceFactory` sein. Diese abstrakte Factory wird in `FastAParser` bereits verwendet. Ihre konkreten Implementierungen der Factory sollen entweder `SimpleSequence`- oder `SparseDNASequence`-Objekte liefern.

(c) Implementieren Sie eine Klasse `CompositeSequence` als Kompositum. Diese Klasse soll das `Sequence`-Interface (direkt oder indirekt) implementieren und Aufrufe von Methoden an enthaltene Teil-`Sequence`-Objekte weiterleiten.

(d) Erweitern Sie `SequenceRegion` (und ggf. abgeleitete Klassen) um eine Methode `Sequence extract(Sequence)`, die die Teil-Sequenz zu einer Region extrahiert. Für Einzel-Regionen (Exons) soll ein entsprechendes `SubSequence`-Objekt (erzeugt mit `getSubSequence(int,int)` aus `Sequence`) zurückgegeben werden. Für zusammengesetzte Regionen (Transcript) soll das entsprechende `CompositeSequence`-Objekt aus Teilaufgabe (c) zurückgegeben werden, welches aus den Teilsequenzen für die einzelnen Exons besteht.

Die Strang-Information dürfen Sie entweder ignorieren oder die Generierung des reversen Komplements geeignet umsetzen.

(e) Testen Sie Ihre Implementierung, indem Sie den `FastAParser` auf eine FastA-Datei mit DNA-Sequenzen anwenden. Generieren Sie Beispielfälle für `SequenceRegion`-Objekte und wenden Sie diese an, um `SubSequence`- und `CompositeSequence`-Objekte zu erzeugen.

Abgabe: 29.11.2023