Esercitazioni Java: creazione di piattaforma bioinformatica per la rappresentazione di sequenze DNA, RNA, proteine (riferimento progetto Bioinformatics1).

1. Creare interfaccia java per rappresentare un simbolo delle sequenze.
   Nome interfaccia: Symbol
   Metodi:
   public String getName();
   // Ritorna il nome del simbolo (es. “Adenina”)
   public String getDescription();
   // Ritorna una descrizione del simbolo (es: “A”)
   public Alphabet getAlphabet();
   // Ritorna l’alfabeto a cui appartiene il simbolo

2. Creare interfaccia java per rappresentare un alfabeto di simboli.
   Nome interfaccia: Alphabet
   Metodi:
   public void addSymbol(Symbol s);
   // Aggiunge il simbolo s all’alfabeto
   public Iterator<Symbol> iterator();
   // Iteratore sui simboli dell’alfabeto
   public void removeSymbol(Symbol s);
   // rimuove il simbolo s dall’alfabeto
   public int size();
   // numero di simboli dell’alfabeto
   public boolean contains(Symbol s);
   // true se il simbolo s è contenuto nell’alfabeto

3. Creare interfaccia java per rappresentare una lista di simboli.
   Nome interfaccia: SymbolList
   Metodi:
   public Alphabet getAlphabet();
   // Ritorna l’alfabeto a cui si riferisce la SymbolList
   public void setAlphabet(Alphabet a);
   // Imposta a come alfabeto della SymbolList
   public Iterator<Symbol> iterator();
   // Iteratore sui simboli della SymbolList
   public int length();
   // Numero di simboli nella SymbolList
   public String seqString();
   // Stringa relative alla SymbolList
   public String subStr(int start, int end);
   // Sottostringa da start a end della SymbolList

public Symbol symbolAt(int index);
// Simbolo in posizione index della SymbolList
public void add(Symbol s);
// Aggiunge il simbolo s alla SymbolList

4. Creare interfaccia java per rappresentare una sequenza di simboli. Una sequenza è una SymbolList con un campo nome. L'interfaccia deve quindi estendere l'interfaccia SymbolList.
   Nome interfaccia: Sequence
   Metodi (oltre a quelli di SymbolList):
   public String getName();
   // Nome della sequenza
   public void printSequenceDescriptions();
   // Stampa a video della sequenza nel seguente format:
   // nomeSimbolo1 – Descrizione simbolo1
   // nomeSimbolo2 – Descrizione simbolo2
   // ….
   // Es:
   // A – Adenine
   // A – Adenine
   // T – Thymine
   // ….

5. Creare classe SimpleAtomicSymbol che implementa l'interfaccia Symbol.
   Variabili d'istanza della classe:
   String name
   // Il nome del simbolo (es. "A")
   String description
   // Una descrizione del simbolo (es: "Adenine")
   Alphabet alphabet
   // L'alfabeto a cui si riferisce il simbolo.

   Metodi della classe (oltre a quelli dell'interfaccia Symbol):
   SimpleAtomicSymbol(String symbolName, String symbolDesc, Alphabet symbolAlphabet)
   // Costruttore
   public void setName(String name)
   // Ritorna attributo name dell'oggetto
   public void setDescription(String description)
   // Ritorna attributo description dell'oggetto
   public void setAlphabet(Alphabet alphabet)
   // Ritorna alphabet dell'oggetto

6. Creare classe SimpleAlphabet che implementa l'interfaccia Alphabet.
   Variabili d'istanza della classe:
   String name
   // Il nome dell'alfabeto
   List symbols
   // Lista di simboli presenti nell'alfabeto (implementare la lista con un oggetto di tipo ArrayList. Vedi Javadocs 1.6)

   Metodi della classe (oltre a quelli dell'interfaccia Alphabet):
   public SimpleAlphabet(String alphabetName)
   // Costruttore 1
   public SimpleAlphabet(List alphabetSymbols, String alphabetName)
   // Costruttore 2

7. Creare classe SimpleSymbolList che implementa interfaccia SymbolList
   Variabili d'istanza della classe:
   List<Symbol> list
   // Lista di simboli (implementare la lista con un oggetto di tipo ArrayList. Vedi Javadocs 1.6)
   Alphabet alphabet:
   // Alfabeto a cui la lista si riferisce

   Metodi della classe (oltre a quelli dell'interfaccia SymbolList):
   public SimpleSymbolList()
   // Costruttore
   public List<Symbol> getList()
   // Ritorna la lista di simboli
   public void setList(List<Symbol> list)
   // Imposta la lista di simboli

8. Creare classe SimpleSequence che implementa interfaccia Sequence
   Variabili d'istanza della classe:
   List<Symbol> list
   // Lista di simboli (implementare la lista con un oggetto di tipo ArrayList. Vedi Javadocs 1.6)
   Alphabet alphabet
   // Alfabeto a cui la lista si riferisce
   String name
   // Nome della sequenza

Metodi della classe (oltre a quelli dell'interfaccia Sequence):

public SimpleSequence(String seqName)
// Costruttore 1
public SimpleSequence(ArrayList l, String seqName)
// Costruttore 2
public List<Symbol> getList()
// Ritorna la lista di simboli
public void setList(List<Symbol> list)
// Imposta la lista di simboli
public void setName(String name)
// Impost ail campo name dell'oggetto sequenza

9. Creare classe AlphabetManager con i seguenti metodi:
   - public static Alphabet alphabetForName(String name)
     // Restituisce un alfabeto di simboli di DNA se name="DNA", un alfabeto di simboli di RNA se name="RNA", un alfabeto di simboli di codoni se name="Codon", un alfabeto di simboli di Amminoacidi se name="Protein" ("+" = ammino acido start, "*"= amminoacido stop). NB: l'alfabeto ritornato deve essere lo stesso (stesso oggetto) anche se il metodo viene richiamato piu' volte. Quindi, ad esempio, l'alfabeto di simboli di DNA deve essere univoco. Si consiglia di creare un metodo locale (privato) per la generazione di ciascun alfabeto.

10. Creare classe DNATools con i seguenti metodi:
    - public static Sequence createDNASequence(String dna, String name) throws Exception
      // Genera una sequenza di DNA a partire da una stringa dna ed una stringa name, dove la stringa dna può essere ad esempio dna="ATTACAGGA". L'eccezione deve essere lanciata se si trova un simbolo non appartenente all'alfabeto di simboli del dna (es: dna="TTAFCC")
    - public static Symbol a()
      // Restituisce il simbolo dell'adenina per l'alfabeto di DNA
    - public static Symbol t()
      // Restituisce il simbolo della timina per l'alfabeto di DNA
    - public static Symbol c()
      // Restituisce il simbolo della citosina per l'alfabeto di DNA

- public static Symbol g()
  // Restituisce il simbolo della guanina per l'alfabeto di DNA
- public static Symbol complement(Symbol s)
  // Restituisce il simbolo complementare rispetto ad s (es: s="A" -> complement ="T")
- public static Sequence complement(Sequence seq)
  // Restituisce la sequenza complementare rispetto a seq
- public static Sequence reverse(Sequence seq)
  // Restituisce la sequenza inversa rispetto a seq
- public static Sequence reverseComplement(Sequence seq)
  // Restituisce la sequenza complementare ed inversa rispetto a seq
- public static Symbol toRNA(Symbol s)
  // Restituisce il simbolo RNA relativo ad un simbolo DNA (es: T→U)
- public static Sequence toRNA(Sequence seq)
  // Restituisce la sequenza RNA relativa ad una sequenza DNA (es: ATTACG→AUUACG)

Suggerimento: utilizzare la classe HashMap per rappresentare le mappe per la conversione di simboli (complemento, trasformazione DNA->RNA).

11. Creare le classi RNATools e ProteinTools che consentano di eseguire le operazioni descritte al punto 10 ma su sequenze di RNA ed Amminoacidi rispettivamente. Nella classe RNATools implementare anche un metodo

    public static Sequence translate(Sequence seq)

    che esegue la traduzione di una sequenza di RNA in una sequenza proteica per mezzo della tabella dei codoni http://en.wikipedia.org/wiki/Genetic_code#RNA_codon_table Fare in modo che il metodo possa generare le tre traduzioni ottenibili dai tre reading frames.

12. Creare una classe Main contenente un metodo main per testare le classi sopra elencate ed i loro metodi. Ad esempio:

    - Generare una sequenza s di DNA "TTACGTA" con nome "My first DNA sequence".
    - Stamparla a video con il metodo printSequenceDescriptions().
    - Calcolare il complemento di s e stamparlo a video
    - Calcolare l'inversa di s e stamparla a video
    - Calcolare il complemento inverso di s e stamparlo a video
    - Trasformare la sequenza s in una sequenza sRNA di RNA e stamparla a video
    - Generare una sequenza proteica p "+EASTTARE*" (dove + è l'amminoacido di start e * quello di stop) con nome "My first protein sequence"
    - Eseguire la traduzione di una stringa di DNA in una stringa di amminoacidi