1) Analogamente a quanto fatto nell'esercitazione precedente con la classe **Razionale** scrivete una classe **Complex** che permetta di fare le operazioni di somma, sottrazione, distanza, confronto e conversione in String in forma binomiale *a + i b*.
   Potreste ad esempio implementare i seguenti costruttori e metodi:
   Complex(double re, double im)
   Complex(double re)
   Complex()
   Complex somma(Complex altro)
   Complex sottrai(Complex altro)
   double distanza(Complex altro)
   String toString()
   booelan equals(Complex altro)
   Scrivete un programma di prova in cui create alcuni oggetti con valori letti da tastiera utilizzando uno *Scanner* e verificate il corretto funzionamento dei metodi e costruttori implementati.

2) Definire una classe **Stack** che implementi una pila di 100 String tramite un *ArrayList*<*String*>. Le funzioni membro della classe devono essere:
   void push(String s)
   String pop()
   boolean isEmpty()
   boolean isFull().
   Scrivete un programma che crea un oggetto **Stack** e, tramite un menu testuale, verifica il corretto funzionamento della classe.
   Implementate infine i metodi *toString* e *equals* e verificatene il corretto funzionamento.

3) Si costruisca una gerarchia di classi per rappresentare veicoli su terra considerando le seguenti entità: **Veicolo, VeicoloAMotore, Ciclomotore, Automobile, Bicicletta**.
   Un **Veicolo** è caratterizzato da una posizione, una velocità iniziale e un'accelerazione. Per rappresentarli si usi una classe **Vettore2D** che rappresenta un vettore tramite le componenti x e y (double) e fornisce opportuni costruttori e metodi set e get.
   In base ai valori di velocità e accelerazione, un **Veicolo** segue la legge di moto uniformemente accelerato:

$$x = x_0 + v_{0x} * t + a_x * t * t$$
$$y = y_0 + v_{0y} * t + a_y * t * t$$

   (si utilizzi il metodo *muovi(double t)* dove *t* rappresenta la variazione di tempo durante cui il veicolo si muove).
   VeicoloAMotore è un **Veicolo** caratterizzato da un motore con una cilindrata predefinita.
   Ciclomotore è un **VeicoloAMotore** caratterizzato dal numero di telaio (*long*).
   Automobile è un **VeicoloAMotore** caratterizzato dal numero di targa (*String*).
   Bicicletta è un **Veicolo** caratterizzato dal modello (*String*).
   Scrivere un'applicazione che consenta di testare la gerarchia delle classi e di simulare il movimento nel tempo di una **Bicicletta**, un**'Automobile** e un **Ciclomotore** avviati tutti allo stesso istante di tempo.

   Si modifichi la gerarchia in modo da implementare classi astratte dove possibile. Rendere abstract il metodo *muovi()* della classe **Veicolo**. Si supponga che i **veicoliAMotore** si muovano di moto uniformemente accelerato come visto nell'esercizio precedente, mentre le **Biciclette** seguano la seguente legge di moto:

$$x = x_0 + v_{0x} * t + a_x * t * t$$
$$y = cos(x)$$

   Si utilizzi una classe Main per testare le nuove classi.

4) Scrivere un'applicazione che consenta di confrontare due testi in base alle distribuzioni di probabilità dei loro caratteri.
   Dato un testo in input, l'applicazione consente di calcolare un **istogramma** dei caratteri (a-z) presenti nel testo ignorando la punteggiatura e convertendo il testo in caratteri minuscoli.
   Si utilizzi Scanner per leggere il testo da tastiera.

Prevedere un metodo per normalizzare gli elementi di un istogramma (in quale classe andrà definito il metodo?). Normalizzare un istogramma significa dividere ciascun elemento dell'istogramma diviso la somma degli elementi di tutto l'istogramma (n è in questo caso il numero di caratteri da **a** a **z**):

$$somma = \sum_{i=1}^{n} h[i]$$

$$h_{norm}[i] = \frac{h[i]}{somma}$$

Prevedere un metodo che consenta di confrontare due istogrammi normalizzati (in quale classe andrà definito il metodo?) attraverso una misura di similarità nota come *histogram intersection* e definita come di seguito dettagliato. La similarità **s** (con s appartenente all'intervallo [0, 1]) di due istogrammi **h1** e **h2** con **n** elementi ciascuno (n è in questo caso il numero di caratteri da **a** a **z**) è calcolata come:

$$s = \sum_{i=1}^{n} \min\left(h1[i], h2[i]\right)$$

Prevedere anche una versione statica del metodo suddetto.
Utilizzare la similarità sopra definita per confrontare due testi dati in input.
Definite e implementate tutte le classi e i metodi che ritenete opportuni.

5) Si utilizzi la documentazione di Java 8 per studiare le caratteristiche dell'interfaccia java.lang.CharSequence. Quali sono i metodi che una classe che implementa CharSequence deve mettere necessariamente implementare?

6) Si scriva una classe **ReversedString** che implementa CharSequence e la si utilizzi per operare su una stringa scorrendo i caratteri da destra verso sinistra (piuttosto che da sinistra a destra come si fa di solito).

   Per esempio, si vuole che l'output del seguente codice in una classe Main:

```
String s = "Questo è solo un esempio";
ReversedString rs = new ReversedString(s);
System.out.println(rs.subSequence(0, 7));
```

   sia: "oipmese".