1. Scrivete una nuova classe **SmallIntegerSet** che ha le stesse caratteristiche della **PositiveIntegerSet** ma ammette soltanto valori fra 1 e 1000. Analogamente a quanto fatto prima verificate la correttezza dei metodi e che venga lanciata un'eccezione quando provate a inserire un elemento non valido nell'insieme.

2. Scrivete un metodo static nella classe che contiene il main a cui passare una **ArrayList** di oggetti di qualsiasi tipo e che, tramite un **Iterator**, stampa l'intera collezione.

3. Scrivete un programma che usi una **Map** in cui sia le chiavi sia i valori sono stringhe: rispettivamente, i nomi degli studenti e i loro voti in un esame. Chiedete all'utente del programma se vuole inserire o rimuovere studenti, modificarne il voto o stampare tutti i voti. La visualizzazione dovrebbe avere un aspetto simile a questo:
   ```
   Carl: B+
   Joe: C
   Sarah: A
   ```

4. Utilizzando una **Map<String, Shape>** scrivete un programma in cui inserite in una mappa usando il metodo **put()** forme geometriche di vario tipo con valori casuali e a cui assocerete un nome univoco (ad esempio "cerchio1", "quadrato2", "Miorettangolo", etc…). Verificate come col metodo **get()** sia possibile accedere al valore dell'elemento della mappa in base al valore della chiave.

5. Scrivete un programma che, utilizzando il metodo split su una stringa contenente il testo di questo esercizio, determina il numero totale di parole presenti nel testo e la parola che compare con maggiore frequenza. Potreste anche pensare di utilizzare una **Map<String, Integer>** per memorizzare la frequenza di ciascuna parola utilizzando la parola stessa come chiave.

6. Definite una classe **VectorInteger** per la gestione di un array dinamico di oggetti **Integer**. La dimensione del **VectorInteger** viene definita come parametro del costruttore (di default sarà 10) e il **VectorInteger** viene inizializzato con il valore 0. L'accesso ai membri deve avvenire mediante metodi **set** e **get** che verificano le dimensioni del **VectorInteger** ed eventualmente lanciano un'eccezione. Prevedete metodi per la somma, la differenza e il prodotto scalare che restituiscono un nuovo **VectorInteger** contenente il risultato e che verificano che i **VectorInteger** su cui fare l'operazione siano della stessa dimensione (eventualmente lanciate un'eccezione). Prevedete anche un metodo che restituisce un **VectorInteger** ottenuto moltiplicandolo per uno scalare e un metodo che restituisce il modulo (double). Utilizzate un'ArrayList per memorizzare internamente i dati e implementate l'interfaccia **Comparable<VectorInteger>** col metodo **compareTo** che restituisce -1, 0 oppure 1 in base al confronto dei moduli dei **VectorInteger**. Scrivete infine un programma per testare la classe.

7. Utilizzando un ArrayList, si scriva un software che consenta la gestione di una **Rubrica** in grado di memorizzare sia i contatti relativi ad amici personali che relativi a colleghi d'ufficio. Ogni **Contatto** consente di memorizzare informazioni quali: nome, cognome e numero di telefono. Di ogni **Amico** si conosce anche l'indirizzo di casa mentre di ogni **Collega** si conosce la qualifica e il fax.
   Il software deve fornire un menù interattivo simile al seguente:
   a) inserisci un nuovo amico;
   b) inserisci un nuovo collega;
   c) visualizza contatto corrente;
   d) cancella contatto corrente;
   e) modifica contatto corrente;
   f) vai al prossimo;
   g) vai al precedente;
   h) stampa lista degli amici;
   i) stampa lista dei colleghi;
   j) stampa lista dei contatti;
   k) ricerca del numero telefonico del contatto a partire dal cognome e nome;
   l) cancella tutto;
   m) esci.

NOTE PER COMPILAZIONE E TEST A RIGA DI COMANDO IN AMBIENTE LINUX:

`javac -d ../classes -cp ../classes nomeClasse.java` *compila e genera il bytecode*
`java -cp ../classes nomePackage.nomeClasse` *esegue il bytecode sulla JVM*