1. Definite una classe generica **Stack<T>** che implementi una pila di 100 elementi di tipo T tramite un ArrayList e lanci un'eccezione personalizzata MyStackException quando opportuno. Le funzioni membro della classe devono essere:
   `void push(T), T pop(), boolean isEmpty(), boolean isFull()`
   Scrivete un programma che crea un oggetto **Stack<String>** e uno **Stack<Integer>** e verifica il corretto funzionamento della classe.

2. Implementate i metodi equals() e hashcode() e l'interfaccia **Comparable** per le classi **Razionale** e **Data** viste in esercitazioni precedenti e verificatene il corretto funzionamento con un programma di test. Verificate inoltre che il metodo compareTo è utilizzato per ordinare un ArrayList di oggetti della classe **Data**.

3. Utilizzando l'interfaccia **Comparable**, si scriva un'applicazione per ordinare un array di oggetti della classe **Data**. Scrivete quindi una classe che implementa l'interfaccia **Comparator** per confrontare due oggetti **Data** e si scriva un'applicazione per ordinare un ArrayList di oggetti della classe **Data** utilizzando questa classe.

4. Scrivete un programma che, utilizzando il metodo **join**, concatena le stringhe "uno", "due", "tre" separandole con uno spazio, le memorizza in una nuova stringa e la stampa. Convertite quindi tutta la stringa in maiuscolo e, utilizzando **replace**, sostituite allo spazio due trattini "**--**". Stampate quindi il risultato. Riconvertite ora la stringa in minuscolo e, utilizzando il metodo **split("--")**, estraete nuovamente le stringhe "uno", "due", "tre" e stampatele.

5. Modificare l'esercizio 3 dell'esercitazione 5 utilizzando una generica **Collection** invece dell'**ArrayList**.

6. Modificare l'esercizio 3 dell'esercitazione 7 utilizzando una generica **Collection** invece dell'**ArrayList**.

7. Si implementi una gerarchia di classi per rappresentare **Razionale** e **Complex** come classi derivate di una classe astratta **Numero**. Si definisca un'interfaccia **Aritmetica<T>** con le 4 operazioni aritmetiche tra coppie di elementi T che saranno opportunamente implementate nelle classi **Razionale** e **Complex**. Verificate che, contrariamente alla versione dell'esercizio senza generics dell'esercitazione 8, nel caso in cui si provi a fare operazioni fra elementi di tipo diverso il programma non compilerà.

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*