PROGRAMMAZIONE (9 CFU) – A.A. 2020-2021	ES. 10a

1) Definite una classe generica **Stack<T>** che implementi una pila di 100 elementi di tipo T tramite un ArrayList. 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) Definite una classe **ListaCircolare<T>** per la gestione di una lista circolare di oggetti **T**. La classe dovrà prevedere metodi per inserire un nuovo elemento alla posizione corrente, per cancellare l'elemento corrente, per leggere l'elemento corrente e per spostarsi avanti e indietro.

3) Implementate i metodi **equals()** e **hashcode()** per le classi **Razionale**, **Complex** e **Date** viste in esercitazioni precedenti e verificatene il corretto funzionamento con un programma di test.

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

5) 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 0 sono uguali e -1 oppure 1 in base al confronto dei moduli dei **VectorInteger** nel caso in cui questi sono diversi. Scrivete infine un programma per testare la classe.

6) Scrivete le classi necessarie per la simulazione di un mazzo di carte siciliane con tutti i metodi che ritenete utili. Potreste pensare a una classe **Carta** che rappresenta una singola carta e una classe **Mazzo** che contiene una lista di carte e utilizza le funzionalità del JCF per mescolare, ordinare, prendere e posare carte nel mazzo.

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

Digitare per ciascuna classe:
`javac nomeClasse.java` *(compila e genera il bytecode)*
Digitare per la classe che contiene il main:
`java nomeClassePrincipale` *(esegue il bytecode sulla JVM)*