A. Creare un sistema di gestione dei prodotti per un negozio al dettaglio online. Il sistema deve consentire agli utenti di aggiungere, visualizzare e ordinare i prodotti utilizzando classi astratte, interfacce e raccolte (elenchi). Inoltre, deve dimostrare la funzionalità di ordinamento.

Requisiti:

1. **Classe astratta (`Prodotto`)**
    - Creare una classe astratta `Prodotto` con le seguenti proprietà:
        - `String name`
        - `Double price`
        - `String categoria`
    - Definire un metodo astratto `Double calculateDiscount()` che sarà implementato nelle sottoclassi.
    - Fornire un metodo `toString()` per visualizzare i dettagli del prodotto.
2. **Classi concrete (`Elettronica` e `Abbigliamento`)**
    - Creare due sottoclassi di `Prodotto`: `Elettronica` e `Abbigliamento`.
    - Implementare il metodo `calculateDiscount()`:
        - Per l'`Elettronica`, applicare uno sconto del 10% se il prezzo è superiore a 500 dollari.
        - Per l'`Abbigliamento`, applicare uno sconto del 15% se la categoria è “Abbigliamento invernale”.
3. **Interfaccia (`Ordinabile`)**
    - Definire un'interfaccia **`Ordinabile`** con un metodo `List<Product> sortByPrice(List<Product> products)`.
    - Implementare questa interfaccia in una classe `ProductManager`.
4. **Classe Product Manager (`ProductManager`)**
    - Questa classe deve gestire l'elenco dei prodotti:
        - Aggiungere prodotti a un elenco.
        - Visualizzare tutti i prodotti con i relativi dettagli.
        - Ordinare i prodotti per prezzo utilizzando il metodo `sortByPrice` (in ordine crescente).
    - Dimostrare l'ordinamento utilizzando `Collections.sort()` con un comparatore personalizzato (Interface Comparable<...>).
5. **Programma principale (`ProductApp`)**
    - Creare una classe `ProductApp` con un metodo main.
    - Istanziare `ProductManager` e aggiungere diversi prodotti di ogni tipo (`elettronica` e `abbigliamento`).
    - Visualizzare l'elenco dei prodotti prima e dopo l’ordinamento

B. Risolvere il problema delle 8 regine per un numero di regine pari a N. Su una scacchiera quadrata di lato N, posizionare N regine in modo tale che nessuna di esse sia sotto scacco. NB: le regine lavorano in verticale, in orizzontale e in diagonale. Quindi i vincoli sono:

1. non posso avere due regine sulla stessa riga
2. non posso avere due regine sulla stessa colonna
3. non posso avere due regine sulla stesa diagonale

NB: per utilizzare l'approccio ricorsivo tenere presente che per risolvere il problema delle 8 Regine, provo a piazzare la prima regina sulla prima riga (parto dalla posizione 0) e poi risolvo il problema per 7 righe da 8 posti, stando attendo a che le regine che posiziono non siano soto scacco di quelle già messe:

- stessa riga non è possibile poiché ne metto una per ogni riga
- stessa colonna va verificato
- Per la diagonale osserviamo che due elementi sono sulla stessa diagonale se dato il primo elemento che sta in posizione (r,c), la posizione dell'altro elemento non sia in una posizione ottenibile sommando/sottraendo lo stesso valore alle coordinate del'altra. Esempio
  - Esaminare tutte le posizioni successive (basso destra o basso sinistra rispetto alla regina corrente)
    - una regina sia in riga 2 e colonna 4: se una seconda regina è in posizione riga 2+k e colonna 4+k allora è sotto scacco (esempio: (2,4) (2+3, 4+3). provare per credere)
    - una regina sia in riga 2 e colonna 4: se una seconda regina è in posizione riga 2+k e colonna 4-k allora è sotto scacco (esempio: (2,4) (2+1, 4-1). provare per credere)
  - Esaminare tutte le posizioni precedenti (alto destra o alto sinistra rispetto alla regina corrente)
    - una regina sia in riga 2 e colonna 4: se una seconda regina è in posizione riga 2-k e colonna 4+k allora è sotto scacco (esempio: (2,4) (2-2, 4+2). provare per credere)
    - una regina sia in riga 2 e colonna 4: se una seconda regina è in posizione riga 2-k e colonna 4-k allora è sotto scacco (esempio: (2,4) (2-2, 4-2). provare per credere)

C. Implementare l'algoritmo di Fibonacci:

1. F(n) = F(n-1) + F(n-2) con condizioni iniziali F(0)=1 e F(1)=1
2. utiilzzare la tecnica della memoization dove se ho già calcolato il fattoriale di *m* allora inserisco *m* in una HashMap (HashMap<Integer, Integer> hm = new HashMap<>();) come chiave associata al valore F(*m*) e la prossima volta non dovrò ricalcolarlo.