## Scaletta 10/11/22

### Parte Collettiva

1. Sintesi di un metodo iterativo che, dati due *array* `a` e `b`, produce un terzo *array* `c` in cui `a[0,...,a.length)` coincide con `c[0,...,a.length)` e `b[0,...,b.length)` coincide con `c[a.length,...,a.length+b.length)`.
   - AppendItTest.java è la classe di *test* che illustra il comportamento atteso del metodo
   - Da AppendIt.java a AppendItFinale.java che introduce il costrutto iterativo `for`
2. Sintesi di un metodo ricorsivo dicotomico che, applicato ad un *array* `a` e un valore `v`, restituisce un *array* i cui elementi sono quelli di `a` strettamente inferiori a `v`, rispettando il seguente vincolo:

   > "L'*array* `a` **deve essere percorso una sola volta**, cioè non è ammesso contare in anticipo quanti elementi occorrerà filtrare per fissare dimensione dell'*array* da restituire come risultato."

   - FiltroDiTest.java illustra cosa ci si aspetta
   - Da FiltroDi.java a FiltroDiFinale.java
   - FiltroDiSimulazione.java è una versione adatta per la simulazione per visualizzare quali e quanti *array*, diversi dei quali diventeranno inaccessibili, sono generati nello *heap*

### Parte in autonomia

1. Sintetizzare due classi Java:
   - `FiltriArrayDi.java`
   - `FiltriArrayDiTest.java`

   la prima con soli metodi ricorsivi dicotomici che soddisfano le specifiche qui sotto elencate, la seconda classe di *test* della prima.

   Siccome le specifiche richiedono di filtrare gli elementi di un *array* dato, ottimo sarebbe avere due versioni di ciascun metodo:
   - *Prima versione*. Si scorre due volte l'*array* da cui filtrare:
     - una per contare il numero di elementi da filtrare,
     - una per filtrarli effettivamente
   - *Seconda versione*. Si scorre l'*array* una sola volta.

   Le specifiche sono quelle già incontrate:

- Dati un *array* `a` ed un intero `limiteSuperiore`, restituire un *array* con tutti e soli i valori interi in `a` minori del valore `limiteSuperiore`
- Dato un *array* `a`, restituire un *array* con tutti e soli gli elementi dispari di `a`
- Dato un *array* `a`, restituire un *array* con tutti e soli gli elementi che in `a` occupano posizioni pari
- Dato un *array* `a`, restituire un *array* con tutti e soli gli elementi che in `a` occupano una posizione dispari avendo, simultaneamente, valore pari
- Dati un *array* `a` e due valori `min` e `max`, restituire tutti e soli gli elementi di `a` compresi tra `min` e `max`
- Dati un *array* `a` ed un valore `riferimento`, restituire *array* con tutti e soli i valori di `a` che sono il doppio del valore `riferimento`

2. Dati un *array* `a` ed un valore `v`, restituire un *array* con tutti gli indici, tranne quello di valore minimo e di valore massimo, degli elementi di `a` che contengono `v`. `E possibile visitare più volte l'*array*a‘` .
   - CercaNoMinNoMaxTest.java è la classe che contiene un insieme essenziale di *test* da soddisfare
   - CercaNoMinNoMax.java è una possibile soluzione
3. Scrivere una versione ricorsiva a piacere `appendRicorsiva(int[],int[])` di `appendIt(int[],int[])`. La versione più naturale dovrebbe essere quella contro-variante
4. Sintetizzare una versione ricorsiva dicotomica del metodo che, dati due array `a` e `b` determina se `b` è uguale ad almeno un sotto-*array* di `a`. In caso positivo restituisce l'indice di `a` in corrispondenza del quale inizia `b`. Se `b` non compare in `a`, allora il risultato è `-1`.
   - SubSeqItFinaleTest.java contiene degli esempi
   - SubSeqItFinale.java contiene una possibile soluzione iterativa
   - SubSeqDiFinaleTest.java contiene degli esempi
   - SubSeqDiFinale.java contiene una possibile soluzione ricorsiva dicotomica

## Video

- Parte I
- Parte II
- Parte III