Esercitazione

Programmazione Java

ArrayList

- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

- Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list

Creare un oggetto ArrayList


- ```
  ArrayList<String> stringList =
                                  new ArrayList<String> ;
  // Generic ArrayList to store only String
  ```

- ```
  ArrayList<String> stringList = new ArrayList<>();
  // Using Diamond operator
  ```

Add

- stringList.add("Item");
  //no error (String OK)

- stringList.add(**new Integer**(2));
  //compilation error

Size & indexOf

- int size = stringList.size();
- **int** index = stringList.indexOf("Item");

Trovare elemento


```
for (int i = 0; i < stringList.size(); i++)
   String item = stringList.get(i);
   System.out.println("Item " + i + " : " + item);
}

// Da Java5

for(String item: stringList){
System.out.println("retrieved element: " + item);
}
```

Copy & Replace


```
ArrayList<String> copyOfStringList = new
ArrayList<String>();
copyOfStringList.addAll(stringList);
```

```
stringList.set(0,"Item2");
```

Esercizio 1

- Consideriamo due ArrayList di stringhe al1 e al2.
- Scrivere un programa che confronta le due ArrayLists usando il metodo contains() e memorizzando il risultato del confronto in un terzo oggetto di tipo ArrayList

Esercizio 2

- Si realizzi in Java uno stack “generico” e una batteria di test che ne esemplifichi l’utilizzo.

Generic Stack


```
/** Return the number of elements in the stack */
public int getSize();

/** Return the top element from the stack */
public E peek();

/** Push a new element to the top of the stack */
public void push(E o) {   }

/** Return and remove the top element from the stack */
public E pop();

/** Test whether the stack is empty */
public boolean isEmpty() ;

@Override // Override the toString in the Object class
public String toString()
```

Esercizio 3 (standard)

- Progettare una classe java che permette di gestire le informazioni relative agli studenti che frequentano un corso.

Struttura

- public void addStudent(String student);
- public String[] getStudents() ;
- public int getNumberOfStudents();
- public String getCourseName()
- public void dropStudent(String student);
- public void clear();

Esercizio 4

- Da wikipedia:
  - a **heap** is a specialized tree-based data structure that satisfies the *heap property:* if P is a parent node of C, then the *key* (the *value*) of P is either greater than or equal to (*max heap*) or less than or equal to (*min heap*) the key of C. The node at the "top" of the heap (with no parents) is called the *root* node.

Heap in Java

- Si progetti la classe che realizza la struttura (max) Heap in Java
- Si utilizzi un ArrayList come struttura interna di impmentazione
  - private ArrayList<E> list = new ArrayList<>()

Struttura


```
/** Add a new object into the heap */
        public void add(E newObject) ;

        /** Remove the root from the heap */
        public E remove() ;

        /** Get the number of nodes in the heap */
        public int getSize() ;

        @Override /** Override the proctected clone method defined in
                the Object class */
        public Object clone() throws CloneNotSupportedException ;

        @Override /** Override the equals method in the Object class
*/
        public boolean equals(Object other);
```