Sorteggio ditte parimerito in calcolo aggiudicazione

Se come criterio di selezione del primo e secondo operatore economico classificato in presenza di parimerito nel calcolo aggiudicazione è previsto il "sorteggio", l'applicativo lo gestisce in automatico mediante un algoritmo di estrazione casuale.
Vengono svolte le operazioni di seguito descritte:

- viene definita una stringa contenente l'elenco dei codici anagrafici degli operatori economici risultanti parimerito ai fini dell'individuazione del primo e del secondo classificato.
- viene definito un vettore V di lunghezza N, dove N corrisponde al numero di operatori economici elencati nella stringa.
- viene estratto casualmente dal vettore V uno dei valori. Tale valore è il codice dell'operatore economico sorteggiato.
  L'estrazione viene effettuata utilizzando la funzione "Java Random"; tale funzione assicura che:
  - ogni numero ha la stessa probabilità di essere estratto ogni volta che viene lanciata la funzione sullo stesso intervallo di numeri;
  - non c'è mai dipendenza fra un numero estratto e il successivo
- viene ridefinita la stringa contenente l'elenco dei codici anagrafici degli operatori economici, togliendo il codice estratto, in modo da poterla utilizzare nell'eventuale selezione del secondo operatore economico classificato.

Per maggiore trasparenza, viene riportato un estratto dal codice del software, commentato passo-passo.

```
// Viene fornito nella stringa "listaModificabile"
// l'elenco dei codici anagrafici degli operatori economici
// parimerito, separati da virgola
// (es.: lista Modificabile= "'IMP0126','IMP0127','IMP0231','IMP0323','IMP0351'").
if(listaModificabile!=null && !"".equals(listaModificabile)){
// Si istanzia la classe Java Random
Random random = new Random();
// Si crea il vettore "elementi" di dimensione N
// con indice da 0 a N-1
// dove N è pari al numero di operatori economici parimerito
// elencati nella stringa "lista Modificabile".
String elementi[] = listaModificabile.split(",");
if(elementi.length>1){
// Si estrae casualmente un indice da 0 a N-1,
// dove N è pari alla dimensione del vettore;
// l'indice estratto viene assegnato alla variabile "rand".
int rand = random.nextInt(elementi.length - 1);
// Si seleziona la cella "rand" del vettore 'elementi'.
// Tale cella contiene il codice dell'operatore economico
// estratto, delimitato da "'" (es.: ret= "'IMP0127'").
ret = elementi[rand];
// Il codice dell'operatore economico estratto
// viene rimosso dalla stringa 'listaModificabile'.
// Viene rimossa anche l'eventuale "," prima o dopo il codice.
// La stringa così modificata viene utilizzata
// per l'eventuale estrazione del secondo classificato.
listaModificabile = listaModificabile.replace(ret , "");
if(rand==0)
  listaModificabile = listaModificabile.replaceFirst(",", "");
else if(rand == elementi.length - 1)
  listaModificabile = listaModificabile.substring(0, listaModificabile.length() - 1);
else
  listaModificabile = listaModificabile.replace(",,", ",");
// Il codice dell'operatore economico estratto viene ripulito dagli "'".
  ret=ret.replaceAll("'", "");
}else{
  //La lista è formata da un solo elemento
  ret = elementi[0];
  ret=ret.replaceAll("'", "");
  lista Modificabile = "";
}
```

Specifica pubblica della funzione (classe) Java Random.

java.util

Class Random

java.lang.Object
  └ **java.util.Random**

All Implemented Interfaces:
Serializable

Direct Known Subclasses:
SecureRandom

```
public class Random
extends Object
implements Serializable
```

An instance of this class is used to generate a stream of pseudorandom numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula. (See Donald Knuth, *The Art of Computer Programming, Volume 2*, Section 3.2.1.)

If two instances of `Random` are created with the same seed, and the same sequence of method calls is made for each, they will generate and return identical sequences of numbers. In order to guarantee this property, particular algorithms are specified for the class `Random`. Java implementations must use all the algorithms shown here for the class `Random`, for the sake of absolute portability of Java code. However, subclasses of class `Random` are permitted to use other algorithms, so long as they adhere to the general contracts for all the methods.

The algorithms implemented by class `Random` use a `protected` utility method that on each invocation can supply up to 32 pseudorandomly generated bits.