A1 – Vector, Arrays e StringBuffer	**MODULO L4**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Classe **Vector**
- Classe **Arrays**
- Classe **StringBuffer**

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Come realizza Java *l'allocazione dinamica* di varie classi?
2. Quali sono i vantaggi e le funzionalità della classe **Vector** e quali le differenze con il tipo array?
3. Quali sono i vantaggi e le funzionalità della classe **Arrays**?
4. Quali sono i vantaggi e le funzionalità della classe **StringBuffer** e quali le differenze con la classe **String**?

B2) Competenza

1. Come si utilizza in Java l'allocazione dinamica di tipi primitivi?
2. Come si dichiara un oggetto di classe **Vector** e quali sono i suoi metodi principali?
3. Come si dichiara un oggetto di classe **Arrays** e quali sono i suoi metodi principali?
4. Come si dichiara un oggetto di classe **StringBuffer** e quali sono i suoi metodi principali?

(C) ESERCIZI DI COMPRENSIONE

1. La tecnica dell'allocazione dinamica ottimizza l'utilizzo della ……. e, grazie al ……. …….. tipico della JVM, non richiede al programmatore la conseguente ……..
2. La classe **Vector** si serve dell'allocazione …….. e crea un oggetto che rappresenta un …… …… Sugli oggetti di questa classe è possibile operare con …….. prestabiliti per inserire, ispezionare o eliminare elementi. Per usare la classe **Vector** è necessario importare il *package* ……..
3. Un oggetto di classe Vector può contenere sequenze di elementi anche di tipo ……, grazie ache la classe …….. è la superclasse di tutte le classi. Ovviamente, in fase di lettura, il programmatore deve sapere la classe di appartenenza di ogni elemento.
4. La classe **Arrays** implementa un vettore e mette a disposizione del programmatore metodi di ……., di …….. e di ..... Per usare la classe **Arrays** è necessario importare il *package* ……..
5. La classe **StringBuffer** offre la possibilità di creare una stringa ……, ossia modificabile, al contrario di quanto avviene per gli oggetti di classe **String**.
6. Associare a ciascuno dei metodi riportati nella colonna di sinistra, il corrispondente effetto, scegliendolo tra quelli indicati nella colonna di destra..

	Metodo
1	Verifica se vettore vuoto
2	Costruttore classe **Vector**
3	Aggiunge elemento in coda
4	Elimina una posizione
5	Inserisce un elemento

	Effetto
A	**Vector <Double>** v =**new Vector** (10)
B	**void add** (**int** index, **Object** obj)
C	**boolean isEmpty**()
D	**void addElement**(**Object** *obj*)
E	**void removeElementAt** (**int** *index*)

7. Associare al codice riportato nella colonna di sinistra, tipico degli array statici, il corrispondente codice, scegliendolo tra quelli indicati nella colonna di destra, relativi agli array dinamici...

	Array statico
1	**int** v[] = **new int** [10],
2	Non richiede package
3	**for** (**int** i=0;i<v.**length** ;i++)..
4	**System.out.println** (v[3]);
5	Per la stampa serve un ciclo
6	v[i] = 5

	Array dinamico
A	**java.util.Vector;**
B	**Vector <Integer>** v=**new Vector<Integer>**(10);
C	v.**setElementAt** (5, i);
D	**for** (**int** i=0 **;** i<v.**size**() ; i++) ...
E	**System.out.println** (v.**elementAt** (3));
F	**System.out.println** (v);

8. Associare al codice riportato nella colonna di sinistra, relativo alla classe **Arrays**, la corrispondente descrizione, scegliendola tra quelle indicate nella colonna di destra..

	Metodo Arrays
1	**Arrays.fill** (a, **true**);
2	**Arrays.fill** (a, 2, 4, "?");
3	**Arrays.fill** (b, **false**);
4	**arraycopy** (a1, p1,a2, p2, n);
5	**Arrays.fill** (b, 0, 2, **true**);

	Effetto
A	Riempie le posizioni da 0 a 2 di *b* con **true**
B	Copia a1 dalla posizione p1 in a2 da p2
C	Riempie l'array *a* con **true**
D	Riempie l'array *b* con **false**
E	Riempie le posizioni da 2 a 4 di *a* con "?"

9. Nei seguenti frammenti di codice, individuare gli errori sintattici e/o logici e dare una breve descrizione dell'effetto.

a.
```
import java.util;
import java.io.*;
public class LeggiStampa
{   public static void main(String[] args)  Exception
    {   InputStreamReader In = new InputStreamReader (System.in);
        BufferedReader tastiera = new BufferedReader (In);
        Vector <String  = new Vector <String>(1,1);
        for (int i=0; i<5; i++)
            v.addElement(readLine());
        }
        System.out.println( v );
        v.clear;
    ……
}
```

b.
```
import java.util;
import java.io.*;
….
{
public static void main(String[] args)
{  Vector <Integer> v= new Vector <Integer>(1,1);
      System.out.println( v.size() );
v.add(0, 1);
v.add(1, 5);
v.add(2, 2);
System.out.println(size() );
clear();
System.out (v.isEmpty() );
….
}
….
```

c.
```
import java.util;
import java.io.*;
….
    public static void main(String[] args)
    {
        Vector <String>v = Vector <String> (2);
        v.add("Anna");
        v.add("Elio");
        v.add("Gianni");
        v.add("Elena");
        System.out.println( indexOf("Elio") );
        System.out.println( v.firstElement("Anna"));
        System.out.println( v.lastelement());
        ….
    }
    ….
```

d.
```
import java.util;
import java.io.*;
….
    public static void main(String[] args)
    {
        Vector <Object> a = new Vector <Object> (10);
        Integer x = new Integer (5);
        Double y = new Double (3.5);
        String z = new String ("cielo");
        a.add (x);    a.add (y);    a.add (z);
        System.out.println ("a: " + a);
        int intx = ((Integer) a.elementAt (0)).intValue();
        double doubley = ((Double) a.elementAt (1)).doubleValue();
        String stringz = (String) a.elementAt (2);
        System.out.println (intx);
        System.out.println (doubley);
        System.out.println (stringz);
        ….
    }
```

(D) ESERCIZI DI APPLICAZIONE

1. Occorre registrare i dati relativi ad articoli di magazzino, ciascuno dei quali è caratterizzato da un codice intero, descrizione, prezzo, quantità giacente. L'applicazione deve prevedere un menu con le seguenti opzioni:
   a. inserimento di nuovi articoli;
   b. stampa completa del magazzino;
   c. visualizzazione di un dato articolo;
   d. eliminazione di un dato articolo.
   e. conteggio degli articoli registrati

   Prevedere una classe Articolo, contenente i metodi di default per la gestione di un singolo articolo e la classe Magazzino, con i metodi di default e i metodi relativi alle opzioni richieste.
2. Occorre registrare i dati relativi ad una rubrica telefonica, ciascuno dei quali è caratterizzato da un nominativo ed un numero di telefono. L'applicazione deve prevedere un menu con le seguenti opzioni:
   a. inserimento di nuovi articoli;
   b. stampa completa del magazzino;
   c. visualizzazione di un dato articolo;
   d. eliminazione di un dato articolo.
   e. conteggio degli articoli registrati

   Prevedere una classe Contatto, contenente i metodi di default per la gestione di un singolo contatto e la classe Rubrica, con i metodi di default e i metodi relativi alle opzioni richieste.
3. Occorre registrare i dati relativi alle coordinate dei vertici di un poligono piano. L'applicazione deve prevedere un menu con le seguenti opzioni:
   a. inserimento dei valori dei vertici;
   b. stampa completa delle coordinate dei vertici;
   c. eliminazione di un dato vertice;
   d. calcolo del perimetro della figura corrente.

   Prevedere una classe Punto, contenente i metodi di default per la gestione di un singolo vertice e la classe Poligono, con i metodi di default e i metodi relativi alle opzioni richieste
4. Si vogliono registrare in una struttura dati una serie di nomi di squadre, con il relativo punteggio di classifica.. L'applicazione deve prevedere un menu con le seguenti opzioni:
   a. inserimento di una squadra;
   b. eliminazione di una data squadra;
   c. stampa del nome e dei punti della squadra in testa alla classifica.

   Prevedere una classe Squadra, contenente i metodi di default per la gestione di una singola squadra e la classe Classifica, con i metodi di default e i metodi relativi alle opzioni richieste
5. Scrivere un'applicazione che crei un oggetto di classe **StringBuffer** inizializzato con "RAM", lo trasformi in "PROGRAM", poi in "PROGRAMMER" e, infine in "PROGRAMMING".
6. Scrivere un'applicazione che letta da input una stringa, ne esegua il criptaggio tramite una funzione che aumenti di 5 il codice di ogni carattere. Un'altra funzione deve provvedere a decriptare la stringa e stamparla.
7. Scrivere un'applicazione che data una stringa, ne costruisca l'inversa e la stampi.

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 1

Obiettivi: dichiarazione ed utilizzo della classe **Vector**.
Problema: scrivere un'applicazione che simuli il funzionamento di una rubrica telefonica. Ogni nominativo della rubrica è detto contatto e contiene i dati della persona associata. Su ogni contatto deve essere possibile eseguire operazioni di immissione, eliminazione, visualizzazione ed elenco globale.

Traccia della soluzione.
1. Per realizzare la soluzione richiesta occorre creare una classe **Contatto**, contenente almeno i seguenti attributi:
   - *Cognome* (stringa);
   - *Nome* (stringa);
   - *Indirizzo* (stringa)
   - *Telefono* (stringa)

   Prevedere per la classe i metodi di default (costruttori e metodi accessori e modificatori)
2. Collaudare la classe **Contatto**, inserendo un contatto e ristampandolo.
3. Successivamente, implementare la classe **Rubrica**, avente come attributo:
   - un **Vector** *v* di N oggetti di classe **Contatto**

   Nella classe **Rubrica** implementare il metodo *main* che stampi un menu in modalità testuale con le opzioni seguenti:
   - inserimento di una scheda
   - eliminazione di una scheda;
   - visualizzazione di una scheda, immettendo il cognome;
   - elenco alfabetico delle schede.

   Osservazioni.
   L'inserimento deve creare un oggetto di classe **Contatto** e consentire all'utente di immettervi gli attributi corrispondenti.
   L'eliminazione di una scheda, chiede all'utente il cognome del contatto da eliminare e, se questo esiste, elimina la scheda corrispondente.
   La visualizzazione di una scheda, chiede all'utente il cognome del contatto da visualizzare e, se questo è presente, ne visualizza gli attributi.
   L'elenco alfabetico delle schede presenti visualizza i contatti ordinati in senso crescente in base al cognome e può essere ottenuto trasformando il Vector in un Arrays su cui si applica il metodo sort.