

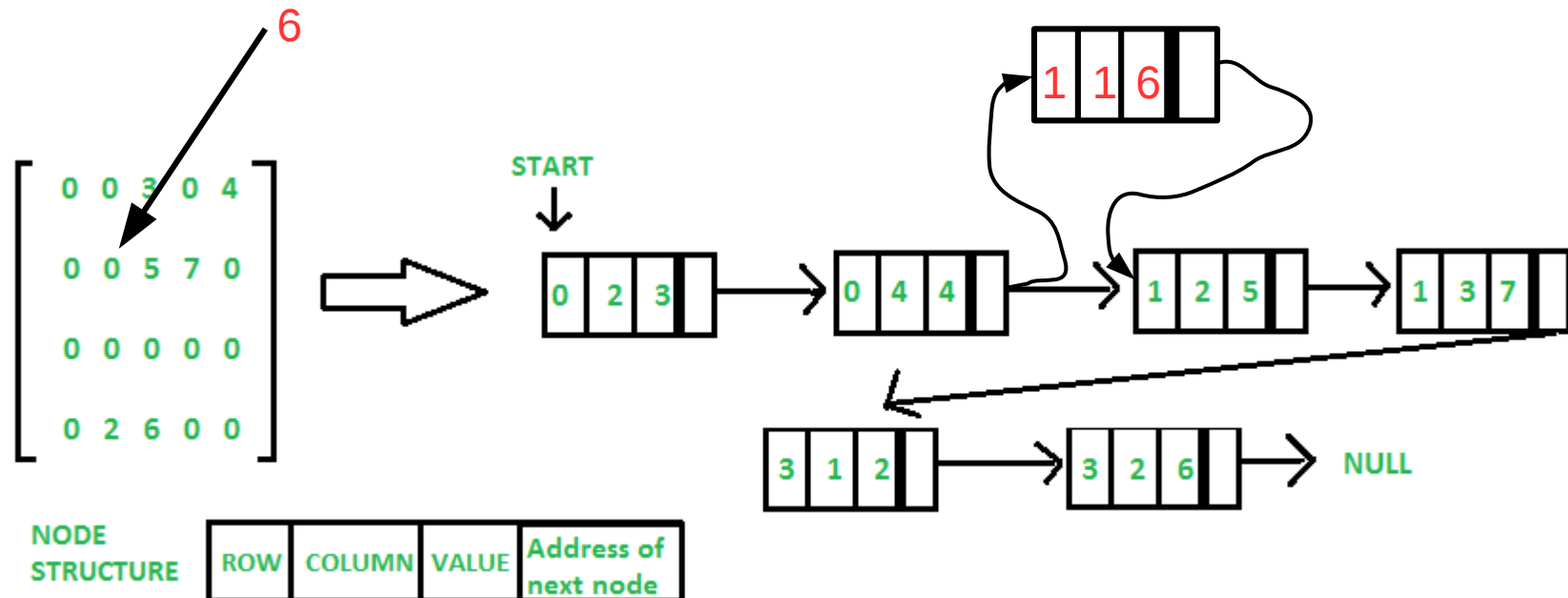
# Esercitazione 1 - matrici sparse

A.A. 2019/2020



# Rappresentazione di matrici sparse

- Si rappresentano soltanto le posizioni e i valori degli elementi diversi da 0



# Rappresentazione di liste in Java/C

- **In Java → si può usare `java.util.LinkedList`**
  - La soluzione proposta sarà C-like
- **In C: occorrerà implementare la lista e le operazioni necessarie su di essa**



# Scheletro della soluzione (cartella java-aux-050318)



Main.java



mat.dat



MatriceSparsa.java



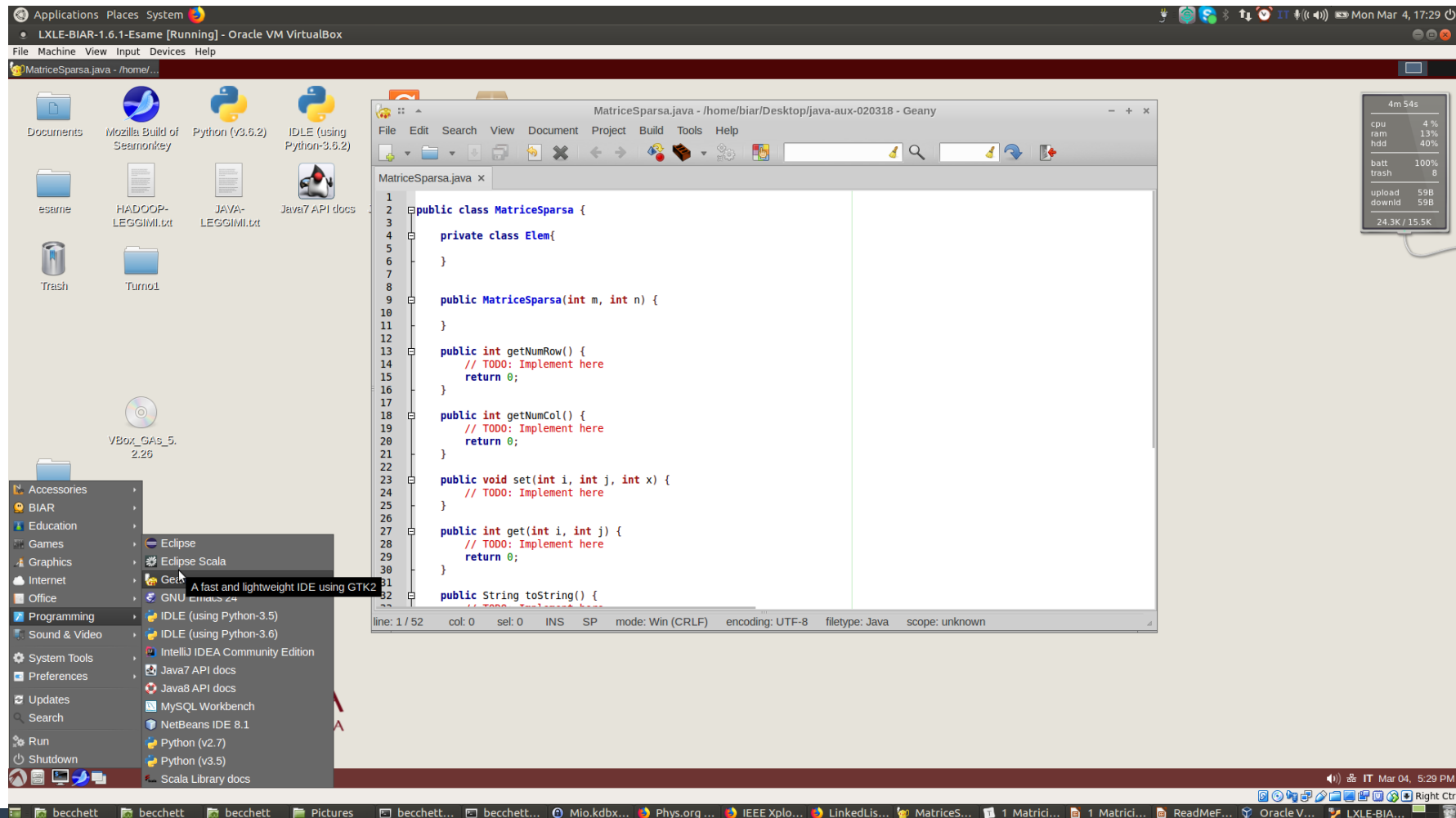
ran\_mat.c

- **Contenuto di C-aux-050318 analogo**
- **ran\_mat** consente di generare una matrice random
- **Main.java** consente di provare il programma
- **MatriceSparsa.java** conterrà la vostra implementazione
- **Nota bene**
  - Il codice compila correttamente anche se contiene soltanto metodi stub
  - Fate in modo che compili sempre durante la fase di implementazione



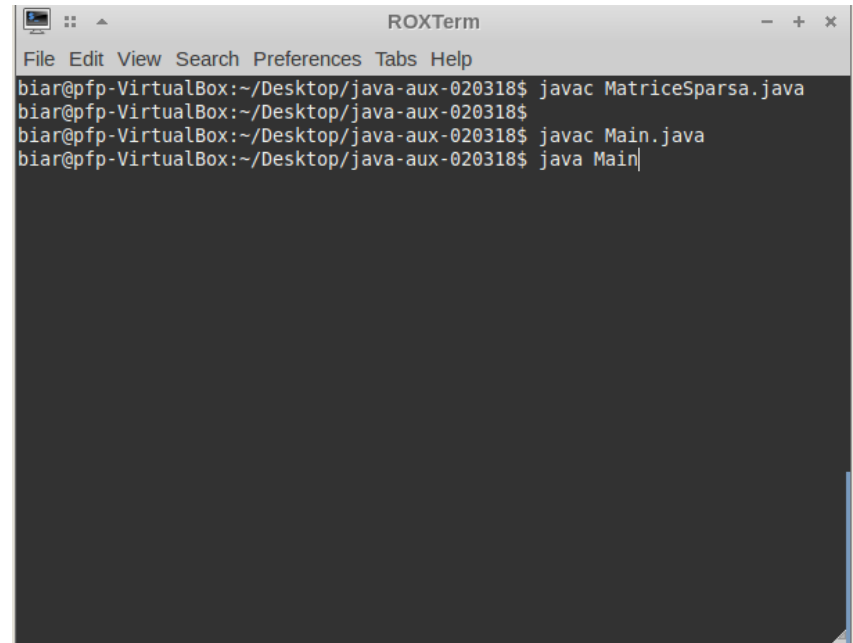
# Ambiente di sviluppo - Editor

- Java/C → Geany (v. figura), vi, Emacs ...



# Compilazione/esecuzione

- **Riga di comando**
  - In C: gcc -o <nome file eseguibile> <file da compilare>
- **In C è possibile usare gdb (debugger)**
  - Richiede di compilare con gcc -g
- **Ambiente integrato**
  - Es. Eclipse
    - Meglio editor + riga di comando se non si ha familiarità con l'ambiente Eclipse
  - Volendo, Geany permette di compilare ed eseguire



```
ROXTerm
File Edit View Search Preferences Tabs Help
biar@pfp-VirtualBox:~/Desktop/java-aux-020318$ javac MatriceSparsa.java
biar@pfp-VirtualBox:~/Desktop/java-aux-020318$
biar@pfp-VirtualBox:~/Desktop/java-aux-020318$ javac Main.java
biar@pfp-VirtualBox:~/Desktop/java-aux-020318$ java Main
```



# Consigli vari

- **Leggere attentamente il testo**
- **Procedere per gradi**
  - 1 task alla volta, probabilmente si finirà a casa
- **Provare continuamente il codice in modo che compili sempre**
  - Main.java e main.c servono a provare il codice dello studente
  - *Non è necessario creare nuovi file*

