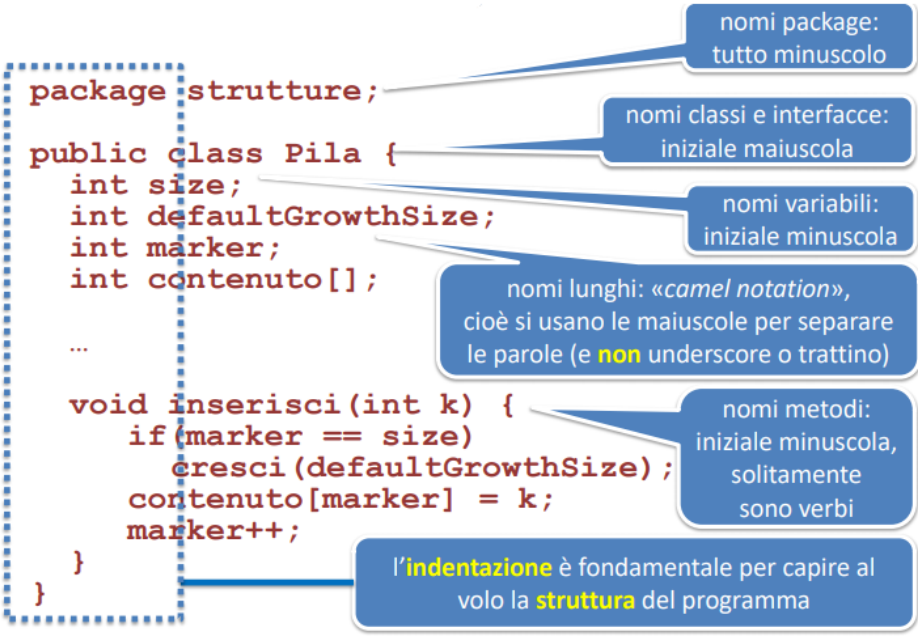
CONVENZIONI

- Maiuscole e minuscole nelle nomenclature



- Ordine metodi e attributi in funzione dei modificatori

Ordine dei modificatori per attributi e metodi:

– le dichiarazioni `static` vengono prima

– `public`, `protected`, (package), `private`

Nomi costanti: tutto maiuscolo

– Es: `static final int MAX_DIM = 5;`

NETBEANS

- Parametri del main

Tasto destra sul progetto>Propietà>Run>Arguments (scrivici qualcosa…)

- Refactor

Tasto destro>refactor>rename

- Default run file

Tendina Projects (pannello sinistra) > "name of project" > Proprieties > Run > Main class

- Equals(), HashCode()

Tasto destro su file>"Insert Code">equals() and hashCode()

- Javadoc

Run>Generate Javadoc

GENERICHE INFO

- Constructor

```
Public class Myclass{
       Public Myclass(){
              X=5;
       }
}
```

- Extends

```
Class Vehicle{ … }
Class Car extends Vehicle{ … }
```

- Override

Quando si overrida una funzione si inserisce:

```
@Override public String toString()
```

- Tipi base
  - Wrapper

Si usano per rendere i tipi base degli oggetti (spesso Java fa la sostituzione in automatico)

```
ArrayList<int> myNumbers = new ArrayList<int>();           // Invalid
ArrayList<Integer> myNumbers = new ArrayList<Integer>();   // Valid
```

  - Converto stringa in tipo base

```
int i = Integer.parseInt("10");
double d = Double.parseDouble("10.5");
boolean b = Boolean.parseBoolean("true");
```

  - Converto tipo base in stringa

```
String Integer.toString(int i);
String Double.toString(double i);
```

  - Float: assegnamento

```
Float f = 3.0f;
```

  - Controlla se il carattere è un numero o una lettera

```
Character.isDigit(char a)        //true se a è un numero
Character.isLetter(char a)       //true se a è una lettera
```

- Math e Random
  - Generare un numero a caso tra un intervallo

```
int max = 10;
int min = 1;
int range = max - min + 1;
// generate random numbers within 1 to 10
int rand = (int)(Math.random() * range) + min;
```

```
Random random = new Random();

int randomInteger = random.nextInt(10);

//genera un numero tra 0 e 9
```

  - Math.max(a, b) / Math.min(a, b)

```
Static base_type max(base_type a,base_type b);      //int, float, double
Static base_type min(base_type a,base_type b);
```

  - Math.abs(a)

```
Static base_type abs(base_type e)      //ritorna il valore assoluto
```

  - Arrotondamento

```
Static double floor(double a)          //ritorna l'intero senza decimali
Static ... round(base_type a)          //ritorna l'arrotondamento. Il tipo di
                                         ritorno varia in base al parametro
```

- Enum
  - Definizione

An enum is a special "class" that represents a group of constants

- Esempio enum

```
public class MyClass {
  enum Level {LOW,MEDIUM,HIGH}
  public static void main(String[] args) {
    //Stampa "MEDIUM"
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}
```

- Name() / toString()

```
public class MyClass {
  enum Level {LOW,MEDIUM,HIGH}
  public static void main(String[] args) {    //stampa "MEDIUM"
    Level myVar = Level.MEDIUM;
    System.out.println(myVar+" "+myVar.name());
  }
}
```

- Ordinal()

```
enum Level {LOW,MEDIUM,HIGH} //stampa la posizione in enum
Level myVar = Level.MEDIUM;
Level myVar2 = Level.LOW;

System.out.println(myVar.ordinal()+" "+myVar2.ordinal());       //"1 0"
```

- compareTo(…)

```
enum Level {LOW,MEDIUM,HIGH} //ordina in base alla posizione in enum
Level myVar = Level.MEDIUM;
Level myVar2 = Level.LOW;

System.out.println(myVar.compareTo(myVar2)); //1
```

- Seme.values()

```
For(Seme x:Seme.values()){     //ritorna tutti i valori dell'enum
    System.out.println(x);
}
```

- Try – Catch

```
try {
  //codice da "provare"
}
catch(Exception e) {
  //codice per gestire gli errori
}
```

JAVADOC

- Scrittura

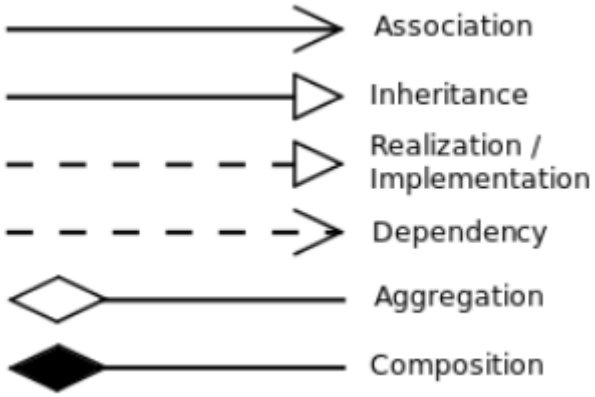
Importante è iniziare il commento con "/**" e terminarlo con "*/"

```
/**        comments          */
```

- Commento di metodi
  - Si commenta prima del metodo
  - "@param name" per descrivere i parametri (dopo il nome del parametro vanno lasciati 2 spazi)
  - "@return" per descrivere il valore di ritorno
- Commento di attributi
  - Si commenta prima dell'attributo
- Commento della classe
  - Si commenta prima della classe
  - È solito usare "@author" per indicare l'autore della classe

UML

- Convenzioni



- Quantità

Su ogni freccia va riportata la quantità di entità che prende parte alla relazione. Si usano poi i puntini per dividere il minimo dal massimo (se non c'è massimo si usa un asterisco)

Classe Object

- clone()

```
Protected Object clone();              //crea una copia dell'oggetto

//l'implementazione è ostica, ma forniamo un esempio:
Test2 t2 = (Test2)t1.clone();
```

- getClass + instanceOf + Object.class

```
Class getClass()                       //la classe istanziata. NON le sottoclassi
Boolean (Object instanceof Class)      //vero se è di quell'istanza o sottoclassi

If(p.getClass() == Persona.class)             //controlla se p è di classe Persona
```

- toString()

```
String toString()
```

- Hash()

```
Int Objects.hash(Object …)       //restituisce gli hashCode degli oggetti
```

- Equals()

```
Boolean equals()    //vero se hanno lo stesso puntatore
```

ARRAY

- Definizione

```
int[] anArray = new int[10];
int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };

int[][] a = new int[3][4];


Object obArray[] = new Object[10];
```

- Arraycopy

```
public static void arraycopy(Object source_arr, int sourcePos,
                             Object dest_arr, int destPos, int len)
source_arr : array sorgente
sourcePos : posizione iniziale dell'array di sorgente
dest_arr : array destinazione
destPos : posizione nell'array destinazione
len : numero di robe da copiare
```

- Ciclare in modo bello

```
Appointment[] aplist=new Appointment[100];
for (Appointment a : aplist){   //viene usato a come iteratore
      System.out.println(a.description);
}
```

STRING

- Definizione

```
String str = "abc";
```

- Metodi
  - Contains

Returns true if and only if this string contains the specified sequence of char values.

```
String s1 = "My name is GFG";

// prints true
System.out.println(s1.contains("GFG"));

// prints false
System.out.println(s1.contains("geeks"));
```

  - Equals

If all the contents of both the strings are same then it returns true. It is **case sensitive**

```
String s1 = "Ciao";
String s2 = "Ciao";
String s3= "ciao";

System.out.println(s1.equals(s2)); //true
System.out.println(s1.equals(s3)); //false
```

  - Split

Ritorna un puntatore ad array di tutte le stringhe divise dal separatore (tra parentesi)

```
String Nomi = "Claudio, Stefano, Antonio";
String[] ArrStr = Nomi.split(", "); //ArrStr[0] è 'Claudio'
```

- Hash

```
String p = "stringa di prova";
p.hashCode();       //restituisce l'hashcode
```

- CompareTo

```
Int i = str.compareTo(str2);     //compara in valore alfabetico
```

- charAt

```
Char String.charAt(i);     //ritorna il carattere a quell'indice
```

- subString

```
Substring(int startIndex)                    //crea una stringa da quell'indice
Substring(int startIndex, int EndIndex)
```

```
String s="SachinTendulkar";
System.out.println(s.substring(6));          //Tendulkar
System.out.println(s.substring(0,6));        //Sachin
```

- Conversione stringhe da altri tipi base

```
String value;
int a = Integer.parseInt(value);
float b = Float.parseFloat(value);
Double c = Double.parseDouble(value);
```

- Array di stringhe e lunghezza

```
//create String array with those values
String[] strArray = new String[]{"Java", "String", "Array", "Length"};

int length = strArray.length;
```

GENERICS

I generics specificano un tipo come fosse un parametro

- Esempio

```
Public class Pila <T>{                  //T diventa un parametro
       Public T estrai() {…}
       Public void inserisci(T o){…}
}

Pila <Integer> p = new Pila <Integer>(10);    //devo impostare il tipo tra <>
Int x = p.estrai();        //quando ritorna il valore NON serve castare
```

- Definizione di un metodo con generics

Se voglio usare questa tipizzazione solo in un metodo di una classe (in cui non uso generics o ne uso di diversi) devo PRIMA definire il generic che uso per la classe e poi il resto della firma normalmente.

```
Public static <E> parameter funcName(…){ … }
```

- Generic non noto

Si può lasciare il tipo non definito:

```
<?>    //si lascia il tipo non definito

//utile nelle funzioni
Static void stampaInsieme(Insieme<?> set){…}
```

- Sottotipi/sovratipi nei generics

È possibile definire un generic che si interessi delle sottoclassi/sovraclassi (evitando i problemi causati dal fatto che con i generic non funziona il principio di Liskov):

```
<? Extends A>                    //qualsiasi classe che estenda A
<? Extends A & B & C & …>        //classe che estende A e
                                 //implementa l'interfaccia B,C,…
<? Super A>                      //qualsiasi classe sopra A
```

COLLECTION

- Definizione con il tipo di elemento

```
List<MyObject> list = new ArrayList<MyObject>();
```

- Metodi comuni
  - Add(Object e)

```
boolean add(Object e);     //aggiungi un elemento
```

  - Remove(Object e)

```
boolean remove(Object e);        //elimina un elemento
```

  - Clear()

```
void clear();       //elimina tutti gli elementi
```

  - Contains(Object o)

```
boolean contains(Object o);
```

  - isEmpty()

```
boolean isEmpty();         //vero se è vuota
```

  - iterator()

```
Iterator iterator();       //ritorna l'iteratore della collection
```

```
int size();
```

  - size()
- Bulk operation comuni
  - addAll(Collection c)

```
boolean addAll(Collection c);    //aggiunge tutti i propri
                                   elementi alla Collection c
```

  - containsAll(Collection c)

```
boolean removeAll(Collection c);        //Toglie tutti gli elementi di c
```

  - retainAll(Collection c)

```
boolean retainAll(Collection c);        //Toglie tutti gli elementi
                                          che NON sono in c
```

  - toArray()

```
Object toArray();          //Ritorna un array con
                             quegli stessi elementi
```

- Sort con comparator

```
Collections.sort(list, class o); //list è una List, mentre o deve essere
                                   una classe che implementa comparator
```

- Iterator
  - Metodi

```
Boolean hasNext();
Object next();      //ritorna il prossimo elemento
Void remove();
```

- List
  - Istanze possibili

```
List a = new ArrayList();

List b = new LinkedList();

List c = new Vector();

List d = new Stack();
```

  - Metodi comuni
    - Get(int index)

```
Object get(int index);     //ritorna l'elemento nell'indice i
```

    - Set(int index, Object element)

```
Object set(int index, Object element);        //crea l'oggetto e lo mette all'indice
```

    - Add(int index, Object element)

```
Void add(int Index, Object element);          //aggiunge l'elemento
```

    - Remove(int index)

```
Object remove(int index);        //elimina l'elemento nell'indice
```

    - Collections.shuffle(List name)

```
Void Collections.shuffle(List l);      //mischia la lista
```

    - indexOf(Object o)

```
Int indexOf(Object o);     //ritorna il primo indice dell'oggetto o
```

    - lastIndexOf(Object o)

```
Int lastIndexOf(Object o);       //ritorna l'ultimo indice dell'oggetto o
```

    - subList(int fromIndex, int toIndex)

```
List subList(int fromIndex, int toIndex);     //crea una lista con
                                                intervallo di quegli indici
```

- LinkedList: metodi
  - addLast(Element e)

```
Void addLast(Element e);
```

  - addFirst(Element e)

```
Void addFirst(Element e);
```

  - getFirst()

```
Element getFirst();        //mostra il primo elemento
```

  - getLast()

```
Element getLast();         //mostra l'ultimo elemento
```

  - removeFirstOccurence(Object o)

```
Boolean removeFirstOccurrence(Object o)
```

  - removeLastOccurence(Object o)

```
Boolean removeLastOccurrence(Object o)
```

- Map
  - Istanzia

```
Map <Integer, Double> map = new HashMap();
```

  - Metodi
    - Put(Object Key)

```
Object put(Object Key, Object value);  //crea il legame tra l'oggetto e la chiave
```

    - Get(Object key)

```
Object get(Object key);    //ritorna l'oggetto legato a quella chiave
```

    - Remove(Object key)

```
Object remove(Object Key);//ritorna l'elemento che elimina alla chiave
```

    - Replace(K key, V Object)

```
Void replace(K Key, V Object); //sostituisce il valore alla chiave
```

    - putAll(map m)

```
Void putAll(Map m);        //prende tutte le coppie da m
```

    - containsKey(Object key)

```
boolean containsKey(Object key); //verifica se c'è un elemento con quella chiave
```

    - containsValue(Object value)

```
boolean containsValue(Object value);    //Verifica se c'è quell'elemento
```

    - keySet()

```
Set keySet();       //crea il set delle chiavi
```

    - values()

```
Collection values();       //ritorna la collezione degli oggetti
```

- Loop

```
for (Map.Entry<String, String> entry : map.entrySet()) {
      System.out.println("Key :"+entry.getKey()+"Value: "+entry.getValue());
}
```

CONFRONTO: Comparable / Comparator

- Comparable

Interfaccia da applicare alla classe stessa per poter ordinare in modo totale.

- Metodo compareTo

```
Int compareTo(Object o);          // <0 this<o
                                  // =0 this=o
                                  // >0 this>o
```

- Esempio

```
Class Car implements Comparable<Car>{
       Public int maxSpeed;
       public String name;
       …
       Public int compareTo(Car o){
              If(this.equals(o)) return 0;
              if(maxSpeed<o.maxSpeed) return(-1);
              return 1;
       }
}
```

```
Class Car implements Comparable{
       …
       Public int compareTo(Object o){ //attenzione a considerare il NULL
              If(!(o instanceof Car)){
                     System.exit(1);
              }
              if(this.equals(o)){
                     return 0;
              }
              if(maxSpeed< ((Car) o).maxSpeed){
                     return -1;
              }
              return 1;
       }
}
MAIN
Collections.sort(list);
```

- Comparator

Interfaccia che permette di delegare il confronto in una classe separata

- Metodi

```
Int compare(Object o1, Object o2);      //Come compareTo
```

- Esempio

```
Class NamedPointComparatorByName implements Comparator<NamedPoint>{
       Public int compare(NamedPoint p1,NamedPoint p2){
              Return(p1.getName().compareTo(p2.getName());
       }
}
```

```
Class NamedPointComparatorByName implements Comparator{
       Public int compare(Object p1, Object p2){//attenzione al NULL
              NamedPoint np1 = (NamedPoint) p1;
              NamedPoint np2 = (NamedPoint) p2;
              return (np1.getName().compareTo(np2.getName()));
       }
}

MAIN
Collections.sort(list, new NamedPointComparatorByName());
```

JAVA FX

- Poligoni
  - Triangolo

```
Polygon polygon = new Polygon();
polygon.getPoints().addAll(new Double[]{
      0.0, 0.0,
      20.0, 10.0,
      10.0, 20.0 });
```

  - Rettangolo/quadrato

```
Rectangle r = new Rectangle();
r.setX(50);
r.setY(50);
r.setWidth(200);
r.setHeight(100);
r.setStroke(Color.BLACK); //bordo nero
r.setStrokeWidth(2); //spessore bordo
```

  - Cerchio

```
Circle circle = new Circle();
circle.setCenterX(100.0f);
circle.setCenterY(100.0f);
circle.setRadius(50.0f);
```

  - Linea

```
Line line = new Line();
line.setStartX(100.0);
line.setStartY(150.0);
line.setEndX(500.0);
line.setEndY(150.0);
```

- Id

È importante ricordare che ogni oggetto può avere un id che si può impostare e leggere in ogni momento. Deve essere univoco e identificare univocamente gli oggetti.

```
Node n.setId("…")
Node n.getId()
```

- Group
  - Costruttori

```
Group()
Group(children)
```

  - Metodi
    - getChildren()

Ritorna la lista degli elementi contenuti nel Group. Usato per aggiungere elementi al Group dopo averlo creato.

```
Group g = new Group();
Rectangle r = new Rectangle();
g.getChildren().add(r);                //aggiunge un elemento
```

- Button
  - Costruttore

```
Button(String text)
```

  - Metodi
    - setMinWidth()

```
setMinWidth(100);   //imposta la larghezza
```

    - setMinHeight()

```
setMinHeight(100);  //imposta l'altezza
```

    - setDisable(boolean)

```
setDisable(true);   //Disabilita il pulsante
```

- Label

È preferibile usare Label invece che un Text in caso il testo NON debba essere modificato

- Text

Classe che permette di aggiungere del testo nelle scene, nei groups e così via.

➢ Costruttori

```
Text()
Text(String text)
```

➢ Metodi

- setText(String s)

```
setText("First row\nSecond row");
```

- setTextAlignment(…)

```
setTextAlignment(TextAlignment.CENTER); //CENTER, JUSTIFY, LEFT, RIGHT
```

- setX / setY

```
setX(Double x);
setY(Double y);
```

- setFont(…)

```
text.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
text.setFont(new Font(20));      //cambia SOLO la dimensione
```

- setFill(…)

```
setFill(Color.BLACK); //cambia il colore
```

- Textfield

Sono le caselle di testo in cui si può scrivere

➢ Costruttori

```
TextField()
TextField(String text) //imposta un testo iniziale
```

➢ Metodi

- clear()

```
clear();      //svuota
```

- setDisable()

```
setDisable(true);   //non modificabile
```

- Scene

➢ Costruttori

```
Scene(Parent root)
Scene(Parent root, double width, double height)
Scene(Parent root, double width, double height, Paint fill)
```

➢ Metodi

```
Group root = new Group();
Scene s = new Scene(root, 300, 300, Color.BLACK);
```

```
setFill(Color.BLACK); //GREEN, BLUE, …
```

- Stage

Corrisponde alla finestra che viene creata, all'interno abbiamo scene, Group ecc…

➢ Metodi

- setScene(Scene s)

```
setScene(Scene s);
```

- setTitle(String s)

```
setTitle(String s);
```

- setX / setY

```
setX(Double x);
setY(Double y);
```

- show() / showAndWait()

```
show();              //mostra
showAndWait();       //mostra e attende
```

- setMinWidth()

```
setMinWidth(100);   //imposta la larghezza
```

- setMinHeight()

```
setMinHeight(100);  //imposta l'altezza
```

- setResizable(boolean)

```
stage.setResizable(false);
```

➢ Impostare le dimensioni dinamicamente

```
primaryStage.widthProperty().addListener(new ChangeListener<Number>() {
      public void changed(ObservableValue<? extends Number> ov, Number oldValue, Number newValue) {
                double w = newValue.doubleValue() * 3 / 5;
                box.setMaxWidth(w);     //imposta un GridPane
                double iw = Math.floor(w / 4);       //imposta tutti i bottoni
                sum.setOBwidth(iw);
                subtract.setOBwidth(iw);
                divide.setOBwidth(iw);
                multiply.setOBwidth(iw);
            }
        }
);
```

- Centrare in mezzo allo schermo

```
Javafx.geometry.Rectangle2D bounds = Screen.getPrimary().getVisualBounds();
double centerX = bounds.getMinX() + (bounds.getWidth() - stage.getWidth())*(1.0f/2);
double centerY = bounds.getMinY() + (bounds.getHeight() - stage.getHeight())*(1.0f/3);

stage.show();
stage.setX(centerX);       //non funziona se messo prima
stage.setY(centerY);
```

- Imposto la larghezza in funzione di un Text

```
Text t = new Text("esempio");
Stage.setWidth(t.getLayoutBounds().getWidth() +20);  //la larghezza va in base al testo
                                                     //più 20 pixel di margine
```

Alert

- AlertType

```
Alert alert = new Alert(AlertType.INFORMATION);
alert.setTitle("Alert Title");
alert.setHeaderText("Alert header:");
alert.setContentText("Alert Text");
alert.showAndWait();
```

CONFIRMATION	
ERROR	Qualcosa è sbagliato
INFORMATION	
NONE	
WARNING	Avverte l'utente di una scelta

Listener / Eventhandler

- Convenience Method

```
Btn.setOnAction(EventHandler a);
```

- Esterno

```
Public class MyClass extends Application{
       Public void start(){
              Button btn = new Button();
              Listener a = new Listener(this);
              btn.addEventHandler(ActionEvent.ACTION, a);
       }
}

class Listener implements EventHandler<ActionEvent>{
       MyClass a = null;

       Public Listener(MyClass temp){
              a=temp;
       }

       Public void handle(ActionEvent t){
              …
       }
}
```

- Interno anonimo

```
Public class MyClass{
      Public void start(){
            Button btn = new Button();
            btn.addEventHandler(ActionEvent.ACTION, new EventHandler<ActionEvent>()
                                                    {public void handle(Event e){…});

            EventHandler a = new Eventhandler(){public void handle(ActionEvent e){…}};
            btn.addEventHandler(ActionEvent.ACTION, a);
      }
}
```

- Tipi di evento

ActionEvent	//eventi della pressione (logica) del pulsante
MouseEvent	//eventi eseguiti dal mouse (es: click su elementi non cliccabili)
KeyEvent	//eventi della tastiera

- Metodi di "Event"
  - fireEvent(Eventtarget a, Event e)

```
Button a = new Button();
Event.fireEvent(a, new ActionEvent()); //preme il pulsante
```

  - N.fireEvent(Event e)

```
Button a = new Button();
a.fireEvent(new ActionEvent()); //preme il pulsante
```

  - e.consume()

```
e.consume(); //termina la propagazione dell'evento
```

  - e.getCharacter()

```
e.getCharacter(); //ritorna il carattere premuto: "u","U"
```

  - e.getCode()

```
e.getCode()==KeyCode.U //Controlla se un tasto è premuto
```

- Layout e posizionamento

I seguenti Layout posizionano con delle regole gli elementi che inseriamo al loro interno. Esse modificano il valore "`layoutX`" ma possiamo personalizzarlo modificando "`setTranslateX`"

➢ Metodi in comune

- setPadding(…)

Imposta i margini: serve a centrare bene la scritta

```
setPadding(new Insets(0, 20, 10, 20));
```

➢ Hbox

```
Pane layout() = new HBox();      //dispone orizzontalmente
layout.getChildren().add(…);
layout.getChildren().add(…);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

- setAlignment(Pos v)

```
setAlignment(Pos value) //indentazione per tutte le celle
```

- setMargin(Node node, Insets value)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

➢ VBox

```
Pane layout = new VBox();        //dispone verticalmente
layout.getChildren().add(…);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

- setMargin(Child a, Inset b)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

- setAlignment(Pos v)

```
setAlignment(Pos.CENTER); //imposta l'indentazione
```

- FlowPane

```
FlowPane layout = new FlowPane();       //dispone orizzontalmente
                                          e va a capo se serve

layout.getChildren().add(…);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

  - setAlignment(Pos v)

```
setAlignment(Pos.CENTER) //indentazione per tutte le celle
```

  - setMargin(Node node, Insets value)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

- StackPane

```
StackPane layout = new StackPane();     //impila le componenti
layout.getChildren().add(…);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

  - setMargin(Node node, Insets value)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

  - setAlignment(Node node, Pos v)

```
setAlignment(Node child,Pos.CENTER) //indentazione per solo "child"
```

  - setAlignment(Pos v)

```
setAlignment(Pos.CENTER) //indentazione per tutte le celle
```

- AnchorPane

```
AnchorPane layout = new AnchorPane();          //posiziona relativamente
Button buttonCancel = new Button("Cancel");

layout.getChildren().add(buttonCancel);
layout.setBottomAnchor(buttonCancel, 8.0);     //lo posiziona in basso a destra
layout.setRightAnchor(buttonCancel, 5.0);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

-

TilePane

```
TilePane layout = new TilePane(); //Crea una griglia di celle con uguali dimensioni
Layout.setVgap(10);       //selezione la distanza verticale
Layout.setHgap(20);       //selezione la distanza verticale
Layout.setPrefColumns(2); //imposta la preferenza nel numero di colonne


Button buttonCancel = new Button("Cancel");
layout.getChildren().add(buttonCancel);

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```

- setMargin(Node node, Insets value)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

- setAlignment(Node node, Pos v)

```
setAlignment(Node child,Pos.CENTER) //indentazione per solo "child"
```

- setAlignment(Pos v)

```
setAlignment(Pos.CENTER) //indentazione per tutte le celle
```

- GridPane

```
GridPane layout = new GridPane(); //Crea una griglia di celle
Button buttonCancel = new Button("Cancel");
layout.add(buttonCancel, 0,1); //specifico riga e colonna

Scene scene = new Scene(layout, 100, 200); //imposto anche la dimensione
stage.setScene(scene);
stage.show();
```

- Restituisce l'oggetto nella cella i,j

```
Node getElementAt(GridPane gp, int i, int j) {
      for (Node x :gp. getChildren()) {
            if ((GridPane.getRowIndex(x) == i) && (GridPane.getColumnIndex(x) == j)) {
                  return x;
            }
      }
      return null;
}
```

- Impostare larghezza colonne

```
//IMPOSTA LA LARGHEZZA DELLE COLONNE IN PIXEL
ColumnConstraints column1 = new ColumnConstraints(100);
ColumnConstraints column2 = new ColumnConstraints(50);
layout.getColumnConstraints().addAll(column1,column2);
```

```
//IMPOSTA LA LARGHEZZA IN PERCENTUALE
GridPane gridpane = new GridPane();
ColumnConstraints col1 = new ColumnConstraints();
col1.setPercentWidth(25);
ColumnConstraints col2 = new ColumnConstraints();
col2.setPercentWidth(50);

gridpane.getColumnConstraints().addAll(col1,col2,col3);
```

- setAlignment(Pos v)

```
setAlignment(Pos.CENTER) //indentazione per tutte le celle
```

- setMargin(Node node, Insets value)

```
setMargin(Node child,Insets value) //margini

->new Insets(0,10,10,10)
```

- setHalignment(Node a, HPos value)

```
setHalignment(Node child,HPos.CENTER) //allineamento orizzontale
```

- setValignment(Node a, VPos value)

```
setValignment(Node child,VPos.CENTER) //allineamento verticale
```

- getRowIndex(Node)/getColumnIndex(Node)

```
GridPane.getRowIndex(Node a);    //ritorna il numero di riga
GridPane.getColumnIndex(Node a);
```

- BorderPane

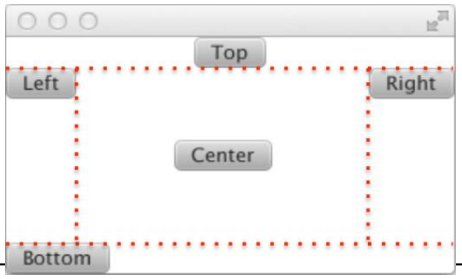
```
BorderPane layout = new BorderPane(); //divide in 5 sezioni la finestra

Button top = new Button("Top");
BorderPane.setAlignment(top, Pos.TOP_CENTER); //imposta l'allineamento SOPRA-CENTRO

layout.setTop(top);

layout.setBottom(new Button("Bottom"));
layout.setLeft(new Button("Left"));
layout.setRight(new Button("Right"));
layout.setCenter(new Button("Center"));

Scene scene = new Scene(layout);
stage.setScene(scene);
stage.show();
```



- setAlignment(Node node, Pos v)

```
setAlignment(Node child,Pos.CENTER) //indentazione per solo "child"
```

USER INPUT

- Senza grafica

```
Import java.util.Scanner;
…
Scanner scanner = new Scanner(System.in);
String s = scanner.nextLine(); //attende finchè non viene dato invio
Int i = scanner.nextInt();
float f = scanner.nextFloat; //per ogni tipo base c'è il lettore
```

- Con grafica (poco raccomandato)

Introduzione a come funziona l'input

```
TextInputDialog dialog = new TextInputDialog("Default answer");
dialog.setTitle("Titolo");
dialog.setHeaderText("Messaggio");
dialog.setContentText("Rispondi qui:");
String s=dialog.showAndWait().get();
```

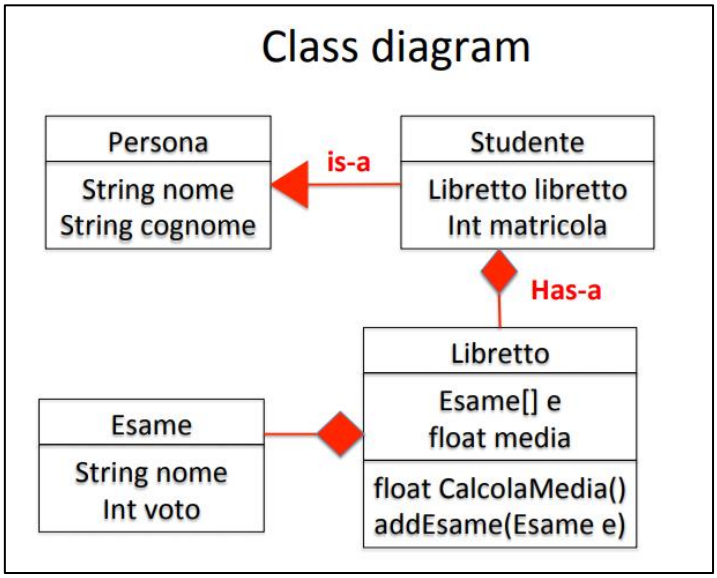
Lettura ciclica dell'input:

```
Scanner scanner = new Scanner(System.in);
String inputString;
int z;

boolean failure=true;
do {
	try {
		System.out.println("Dammi un numero");
		inputString= scanner.nextLine();
		int z=Integer.parseInt(inputString);
		failure=false;
	} catch (Exception ex) {
		failure=true;
	}
} while (failure);
```

ESEMPI

- Studente all'università




```java
class Persona{
    String nome;
    String cognome;
    Persona(String nome, String cognome){
        this.nome=nome;
        this.cognome=cognome;
    }
    @Override public String toString(){
        return(this.nome+" "+this.cognome);
    }
}
```

```java
class Studente extends Persona{
    Libretto libretto;
    int matricola;
    Studente(String nome, String cognome, int matricola){
        super(nome,cognome);
        this.matricola = matricola;
        libretto = new Libretto();
    }
    @Override public String toString(){
        return(super.toString()+" "+matricola);
    }
}
```

```java
class Esame{
    String nome;
    int voto;
    Esame(String nome, int voto){
        this.nome=nome;
        this.voto = voto;
    }
}
```

```
class Libretto{
    Esame []esame;
    int number_exams;
    float media;
    Libretto(){
        esame = new Esame[0];
        number_exams = 0;
        media=0;
    }
    public void addEsame(Esame esame){
        Esame []esame_update=new Esame[number_exams+1];
        arraycopy(this.esame,0,esame_update,0,number_exams);
        esame_update[number_exams]=esame;
        number_exams++;
        this.esame = esame_update;
        calcolaMedia();
    }
    private void calcolaMedia(){
        float somma=0;
        for(int i=0;i<number_exams;i++){
            somma+=esame[i].voto;
        }
        media = somma/number_exams;
    }
    @Override public String toString(){
        String frase;
        frase = "#esami:"+number_exams+" media:"+media;
        frase+="\n ESAME     ||  VOTO";
        for(int i=0;i<number_exams;i++){
            frase+="\n"+esame[i].nome+"    ||  "+esame[i].voto;
        }
        return(frase);
    }
}
```

```
public static void main(String[] args) {
    Studente s=new Studente("Giulio","Cesare",125);
    Esame e=new Esame("Geometria",27);
    s.libretto.addEsame(e);
    s.libretto.addEsame(new Esame("Fisica",22));
    System.out.println(s);
    System.out.println(s.libretto.media);
    System.out.println(s.libretto);
}
```

- 2 Finestre

```
public class TwoWindows extends Application {
    
    @Override public void start(Stage stage) {
        //Creo un testo da visualizzare
        Text t = new Text(50,100, "Prova");
        t.setTextAlignment(TextAlignment.RIGHT);
        t.setWrappingWidth(50);
        t.setFill(Paint.valueOf("GREEN"));

        //FINESTRA 1 (default)
        Group g = new Group(t);
        Scene scene = new Scene(g);
        stage.setTitle("Titolo1");
        stage.setScene(scene);
        stage.setWidth(400);
        stage.setHeight(300);
        stage.setX(1000);
        stage.setY(300);
        stage.show();

        //FINESTRA 2
        Group g2 = new Group(t);
        Scene scene2 = new Scene(g2);
        Stage stage2 = new Stage();
        stage2.setTitle("Titolo 2");
        stage2.setScene(scene2);
        stage2.sizeToScene();
        stage2.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

- Sfera o cilindro

L'utente sceglie un poligono e le misure e viene disegnato.

```
abstract class Poligono {

    int numeroLati;

    Poligono(int numeroLati) {
        this.numeroLati = numeroLati;
    }
}

abstract class Poligono3D extends Poligono {

    int numeroFacce;

    Poligono3D(int numeroLati, int numeroFacce) {
        super(numeroLati);
        this.numeroFacce = numeroFacce;
    }
}
```

```
class Cilindro extends Poligono3D {

    float altezza;
    float raggio;

    Cilindro(float altezza, float raggio) {
        super(2, 3); //convenzione
        this.altezza = altezza;
        this.raggio = raggio;
    }
}

class Sfera extends Poligono3D {

    float raggio;

    Sfera(float raggio) {
        super(0, 1); //convenzione
        this.raggio = raggio;
    }
}
```

```java
public class Poligoni extends Application {

    @Override
    public void start(Stage primaryStage) {

        boolean failure;    //per il do-while
        String nomePoligono = ""; //distinguo i poligoni

        do {
            try {
                /**************sfera/cilindro*************/
                TextInputDialog dialog = new TextInputDialog("Risposta");
                dialog.setTitle("Richiesta poligono");
                dialog.setHeaderText("Inserisci il poligono: sfera/cilindro");
                nomePoligono = dialog.showAndWait().get();
                /*************verifico se è corretto************/
                switch (nomePoligono) {
                    case "sfera":
                    case "cilindro":
                        failure = false;
                        break;
                    default:
                        failure = true;
                }
            } catch (Exception ex) {
                failure = true;
                /********Errore**********/
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("Errore nell'inserimento");
                alert.setHeaderText("Hai dato un valore non valido!");
                alert.setContentText("Clicca 'ok' e immetti un valore valido ");
                alert.showAndWait();
            }
        } while (failure);
        String dimensioni = "";   //dimensioni per il costruttore Poligoni3D
        Poligono3D poligono;    //dinamic binding

        //Dimensioni per Poligoni3D
        double raggio;
        double altezza;

        //lo creo qua così ci lavoro nel do-while
        Group root = new Group();

        //lo uso per i 2 poligoni
        TextInputDialog dialog = new TextInputDialog("Risposta"); //mostra la finestra di richiesta dati
        dialog.setTitle("Richiesta dati");

        do {
            try {
                if (nomePoligono.equals("cilindro")) {
                    /******chiedo dimensioni*****/
                    dialog.setHeaderText("Inserisci raggio e altezza divisi da una virgola e uno spazio (, )");
                    dimensioni = dialog.showAndWait().get();

                    /******ricavo dimensioni + creo oggetto*****/
                    String[] arrSplit = dimensioni.split(", ");
                    poligono = new Cilindro(Float.parseFloat(arrSplit[0]), Float.parseFloat(arrSplit[1]));
                    raggio = ((Cilindro) poligono).raggio;
                    altezza = ((Cilindro) poligono).altezza;
                    Cylinder cilindro = new Cylinder(raggio, altezza);

                    /********Posiziono il cilindro******/
                    cilindro.setTranslateX(100); //posizione
                    cilindro.setTranslateY(100);

                    /*********mostro il cilindro*******/
                    root.getChildren().add(cilindro);
                } else if (nomePoligono.equals("sfera")) {
                    /*****chiedo dimensioni****/
                    dialog.setHeaderText("Inerisci raggio");
                    dimensioni = dialog.showAndWait().get();

                    /******creo l'oggetto****/
                    poligono = new Sfera(Float.parseFloat(dimensioni));
                    raggio = ((Sfera) poligono).raggio;
                    Sphere sfera = new Sphere(raggio);

                    /******posizione*****/
                    sfera.setTranslateX(100); //posizione
                    sfera.setTranslateY(100);

                    /*****mostro la sfera****/
                    root.getChildren().add(sfera);
                }
                failure = false;

            } catch (Exception ex) {
                failure = true;

                /********ERRORE******/
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("Errore nell'inserimento");
                alert.setHeaderText("Hai dato un valore non valido!");
                alert.setContentText("Hai inserito: '" + dimensioni + "'");
                alert.showAndWait();
            }
        } while (failure);

        Scene scene = new Scene(root, 400, 400);

        primaryStage.setTitle("Poligono");
        primaryStage.setScene(scene);
        Scene scene = new Scene(root, 400, 400);

        primaryStage.setTitle("Poligono");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Faccio scegliere il poligono all'utente

Faccio scegliere le dimensioni

Mostro il tutto nella finestra

- Polinomi

Genera dei polinomi e li somma e moltiplica tra loro

```java
public class Polinomio {
    List<Monomio> polinomio;

    public Polinomio(){
        polinomio = new ArrayList();
    }
    public Polinomio(Monomio a){
        polinomio = new ArrayList();
        polinomio.add(a);
    }

    public boolean add(Monomio a){
        try{
            polinomio.add(a);
            normalizza();
            return(true);
        }catch(Exception e){
            System.out.println("Errore");
            return(false);
        }
    }

    //somma i coefficienti con lo stesso esponente
    private void normalizzaSomma(int esponente){
        int first= -1;  //mi appoggio sul primo elemento con quell'esponente
                        //a questo sommo i coefficienti
        for(int i=0;i<polinomio.size();i++){
            if(polinomio.get(i).espon==esponente){
                if(first==-1){  //Sommo i coefficienti nel primo
                    first=i;
                }
                else{
                    polinomio.get(first).coeff+=polinomio.get(i).coeff;
                    polinomio.remove(i);    //tolgo quello che ho sommato
                }
            }
        }
    }
    //Se ci sono monomi con lo stesso coefficiente li sommo
    public void normalizza(){
        Map<Integer,Integer> esponenti = new HashMap<Integer,Integer>();

        //creo mappa: (esponente/#elementi con quel esponente)
        for(int i=0;i<polinomio.size();i++){
            int temp_espo;
            temp_espo = polinomio.get(i).espon;
            if(esponenti.containsKey(temp_espo)){
                //aumento il numero di elementi con quell'esponente
                esponenti.replace(temp_espo, esponenti.get(temp_espo)+1);
            }
            else{
                //dico che almeno un elemento ha questo esponente
                esponenti.put(temp_espo, 1);
            }
        }

        //controllo quanti monomi hanno uno stesso esponente
        for(Map.Entry<Integer,Integer> entry: esponenti.entrySet()){
            //se c'è più di un monomio con quell'esponente sommo i coefficienti
            if(entry.getValue() > 1){
                normalizzaSomma(entry.getKey());
            }
        }
    }

    @Override public String toString(){
        String result="";
        //per stampare il polinomio basta stampare tutti i monomi
        for(int i=0;i<polinomio.size();i++){
            result+=polinomio.get(i).toString();
        }
        return(result);
    }
}
```

```java
public class Monomio {
    int coeff;
    int espon;

    Monomio(int coeff, int espon){
        this.coeff=coeff;
        this.espon=espon;
    }

    @Override public String toString(){
        if(coeff>0){    //stampo anche il segno del coefficiente
            if(espon!=0)
                return("+"+this.coeff+"x^"+espon);
            else
                return("+"+this.coeff);
        }
        else{           //se è negativo o 0 stampo senza il segno
            if(espon!=0)
                return(this.coeff+"x^"+espon);
            else
                return(this.coeff+"");
        }
    }

    public Monomio prodotto(Monomio fattore){
        int coeff = this.coeff*fattore.coeff;
        int espon = this.espon+fattore.espon;

        return(new Monomio(coeff,espon));
    }
}
```

```java
public class Matematica {

    public static Polinomio somma(Polinomio addendo1, Polinomio addendo2){
        Polinomio result = new Polinomio();

        //basta che metto tutti i monomi all'interno di un polinomio e normalizzo
        result.polinomio.addAll(addendo1.polinomio);
        result.polinomio.addAll(addendo2.polinomio);
        result.normalizza();
        return(result);
    }

    public static Polinomio prodotto(Polinomio fattore1, Polinomio fattore2){
        Polinomio result=new Polinomio();

        //basta moltiplicare tutti i monomi per gli altri monomi
        for(int i=0;i<fattore1.polinomio.size();i++){
            for(int j=0;j<fattore2.polinomio.size();j++){
                result.add(fattore1.polinomio.get(i).prodotto(fattore2.polinomio.get(j)));
            }
        }

        result.normalizza();
        return(result);
    }


     public static void main(String arg[]){
        Polinomio p = new Polinomio(new Monomio(-2,1));
        Polinomio p2 = new Polinomio(new Monomio(-2,1));

        p.add(new Monomio(5,-2));
        p2.add(new Monomio(3,2));

        System.out.println("("+p+") + ("+p2+")="+somma(p,p2));
        System.out.println("("+p+") * ("+p2+")="+prodotto(p,p2));
    }
}
```

- Gioco di carte

Gioco: da un mazzo vengono pescate N carte: se tra queste ce ne sono di doppie il giocatore vince, se poi si estrae una carta dal mazzo e se ne trova una già presente nell'estrazione iniziale, si vince doppio

```
public class Carta {
    String nome;
    int valore;
    String seme;

    public Carta(int valore, String seme){
        this.valore = valore;

        switch(valore){
            case 1:this.nome="A";break;
            case 11: this.nome="J";break;
            case 12: this.nome="Q";break;
            case 13: this.nome="K";break;
            default: this.nome=String.valueOf(valore);
        }

        this.seme = seme.toUpperCase();
    }


    @Override public int hashCode() {
        int hash = 5;
        hash = 53 * hash + this.valore;
        hash = 53 * hash + Objects.hashCode(this.seme);
        return hash;
    }


    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Carta other = (Carta) obj;
        if (this.valore != other.valore) {
            return false;
        }
        if(!this.seme.equals(other.seme)){
            return false;
        }
        return true;
    }

    @Override public String toString(){
        return(nome+" di "+seme);
    }

    public static void main(String a[]){
        Carta p = new Carta(2,"Cuori");
        System.out.println(p);
    }
}
```

```
public class Mazzo_ramino extends Mazzo{

    public Mazzo_ramino(){
        //aggiunge le solite carte
        super();
        for(int i=1;i<14;i++){
            mazzo.add(new Carta(i,"Cuori"));
            mazzo.add(new Carta(i,"Quadri"));
            mazzo.add(new Carta(i,"Fiori"));
            mazzo.add(new Carta(i,"Picche"));

            //ripeto perchè ci sono doppioni
            mazzo.add(new Carta(i,"Cuori"));
            mazzo.add(new Carta(i,"Quadri"));
            mazzo.add(new Carta(i,"Fiori"));
            mazzo.add(new Carta(i,"Picche"));
        }
    }
```

```java
public class Mazzo {
    LinkedList<Carta> mazzo;

    public Mazzo(){
        mazzo = new LinkedList();
    }
    public Mazzo(Collection<Carta> c){
        mazzo = new LinkedList(c);
    }

    public void mescola(){
        Collections.shuffle(this.mazzo);
    }

    //estrae le prime n carte
    public Mazzo estrai(int n){
        if(n!=0){
            Mazzo sub_mazzo = new Mazzo();

            for(int i=0;i<n;i++){
                sub_mazzo.mazzo.add(this.mazzo.get(i));
            }

            return(sub_mazzo);
        }
        else return(null);
    }



     //ricerca i doppioni
     public boolean doppelCheck(){
         for(int i=0;i<this.mazzo.size();i++){
             for(int j=i+1;j<this.mazzo.size();j++){
                 if(mazzo.get(i).equals(mazzo.get(j))){
                     return true;
                 }
             }
         }
         return false;
     }

     //Ritorna il mazzo con i doppioni in singola copia
     public Mazzo doppelSet(){
         Mazzo doppie = new Mazzo();
         for(int i=0;i<this.mazzo.size();i++){
             for(int j=i+1;j<this.mazzo.size();j++){
                 if(mazzo.get(i).equals(mazzo.get(j))){
                     doppie.mazzo.add(mazzo.get(i));
                 }
             }
         }
         return doppie;
     }

     @Override public String toString(){
         String risultato="";
         for(Carta i : mazzo){
             risultato+=i.toString()+"\n";
         }

         return(risultato);
     }


     //stampo le carte con i simboli
     public String SymbolicToString(){
         String risultato="";
         for(Carta i : mazzo){
             String seme="";
             switch (i.seme){
                 case "CUORI":seme="\u2764";break;
                 case "PICCHE": seme="\u2660";break;
                 case "FIORI": seme="\u2663";break;
                 case "QUADRI": seme="\u2666";break;
                 default: seme="";
             }

             risultato+=i.nome+seme+" ";
         }

         return(risultato);
     }
```

```
public class Gioco_di_carte extends Application {

    //creo la stringa delle carte estratte
    public String testoFinestra(Mazzo estrazione, Mazzo doppioni){
        String carte = estrazione.SymbolicToString();
        String doppione_vincente = doppioni.SymbolicToString();

        if(!doppione_vincente.equals("")){  //se ci sono doppioni=vinto
            carte+="\nDoppione vincente: "+doppione_vincente;
        }
        return carte;
    }

    @Override public void start(Stage stage) {
        //centro la finestra delle carte nello schermo
        javafx.geometry.Rectangle2D bounds = Screen.getPrimary().getVisualBounds();
        double centerX = bounds.getMinX() + (bounds.getWidth() - stage.getWidth())* (1.0f / 2);
        double centerY = bounds.getMinY() + (bounds.getHeight() - stage.getHeight())* (1.0f / 3);

        Mazzo_ramino p = new Mazzo_ramino();
        p.mescola();
        Mazzo estrazione = p.estrai(10);
        Mazzo doppioni = estrazione.doppelSet();    //il mazzo dei doppioni della estrazione

        //mostro le carte estratte
        Text t = new Text(10, 20, testoFinestra(estrazione,doppioni));
        t.setTextAlignment(TextAlignment.CENTER);
        t.setFill(Paint.valueOf("BLACK"));

        //costruisco la finestra
        Group g = new Group(t);
        Scene scene = new Scene(g);
        stage.setTitle("ESTRAZIONE");
        stage.setScene(scene);
        stage.setWidth(t.getLayoutBounds().getWidth() + 20);    //imposto la larghezza in base al testo
        stage.setHeight(50);


        if (estrazione.doppelCheck()) {
            //estraggo solo una carta e vedo se è la stessa per la vittoria doppia
            Carta singola = p.estrai(1).mazzo.get(0);

            if (doppioni.mazzo.contains(singola)) {
                stage.show();
                //IMPOSTARE X,Y DELLO STAGE FUNZIONA SOLO DOPO show()
                stage.setX(centerX);
                stage.setY(centerY);

                //annuncio vittoria doppia
                Alert vittoria = new Alert(AlertType.INFORMATION);
                vittoria.setTitle("Vittoria");
                vittoria.setHeaderText("Hai conquistato una VITTORIA DOPPIA!");
                vittoria.setX(200);
                vittoria.setY(200);
                vittoria.showAndWait();

                stage.hide();//nascondo lo stage
            } else {
                stage.show();
                stage.setX(centerX);
                stage.setY(centerY);

                //annuncio vittoria
                Alert vittoria = new Alert(AlertType.INFORMATION);
                vittoria.setTitle("Vittoria");
                vittoria.setHeaderText("Hai vinto!");
                vittoria.setX(200);
                vittoria.setY(200);
                vittoria.showAndWait();

                stage.hide();//nascondo lo stage
            }
```

```
        } else {//Sconfitta
            stage.show();
            stage.setX(centerX);
            stage.setY(centerY);

            //Annuncio sconfitta
            Alert vittoria = new Alert(AlertType.INFORMATION);
            vittoria.setTitle("Sconfitta");
            vittoria.setHeaderText("Hai perso!");
            vittoria.setX(200);
            vittoria.setY(200);
            vittoria.showAndWait();

            stage.hide();//nascondo lo stage
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

- Carta con comparatore

Implementa un compareTo() nella classe e un Comparator

```java
public class Carta implements Comparable{
    String nome;
    int valore;
    String seme; //Cuori, Fiori, Quadri, Picche

    public Carta (int valore, String seme){
        this.valore = valore;

        switch(valore){
            case 1:this.nome="A";break;
            case 11: this.nome="J";break;
            case 12: this.nome="Q";break;
            case 13: this.nome="K";break;
            default: this.nome=String.valueOf(valore);
        }

        this.seme = seme.toUpperCase();
    }


    @Override public int hashCode() {
        int hash = 5;
        hash = 53 * hash + this.valore;
        hash = 53 * hash + Objects.hashCode(this.seme);
        return hash;
    }

    @Override public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Carta other = (Carta) obj;
        if (this.valore != other.valore) {
            return false;
        }
        if(!this.seme.equals(other.seme)){
            return false;
        }
        return true;
    }

    @Override public String toString(){
        return(nome+" di "+seme);
    }


    //PRIVILEGIA IL SEME    C>Q>F>P    poi controllo i valori
    @Override public int compareTo(Object o) {
        if(!(o instanceof Carta)){
            System.exit(1);
        }

        //p.seme ritorna il seme in CAPS
        switch (((Carta) o).seme){
            case ("CUORI"):
                if(this.seme.equals("CUORI")){
                    break;
                }
                else{return(-1);}

            case ("QUADRI"):
                if(this.seme.equals("CUORI")){
                    return(1);
                }
                else if(this.seme.equals("FIORI") || this.seme.equals("PICCHE")){
                    return(-1);
                }
                else { break; }

            case ("FIORI"):
                if(this.seme.equals("CUORI") || this.seme.equals("QUADRI")){
                    return(1);
                }
                else if(this.seme.equals("PICCHE")){ return(-1); }
                else{ break;}

            case ("PICCHE"):
                if(this.seme.equals("CUORI") || this.seme.equals("QUADRI") || this.seme.equals("FIORI")){
                    return(1);
                }
                else{break;}
        }
        if(this.valore >((Carta) o).valore)  {  return(1); }
        else if(this.valore <((Carta) o).valore){  return(-1); }
        else{return(0);}
    }
}
```

Equals() e HashCode()

compareTo()

Privilegia il seme

```java
public class CompareCartaByValue implements Comparator {
    //prima considera il valore poi il seme
    //C>Q>F>P
    @Override public int compare(Object o1, Object o2) {
        if(!(o1 instanceof Carta) || !(o2 instanceof Carta))
            System.exit(1);

        Carta c1 = (Carta) o1;
        Carta c2 = (Carta) o2;

        //prima controllo i valori
        if(c1.valore>c2.valore){
            return(1);
        }
        else if(c1.valore<c2.valore){
            return(-1);
        }
        else{   //controlla i semi
            switch (c1.seme){
                case "CUORI":
                    if(c2.seme.equals("CUORI")){ return(0); }
                    else{ return(1); }

                case "QUADRI":
                    if(c2.seme.equals("CUORI")){  return(-1); }
                    else if(c2.seme.equals("QUADRI")){ return(0); }
                    else{ return(1); }

                case "FIORI":
                    if(c2.seme.equals("CUORI") || c2.seme.equals("QUADRI")){
                        return(-1);
                    }
                    else if(c2.seme.equals("FIORI")){
                        return(0);
                    }
                    else{  return(1);  }
                case "PICCHE":
                    if(c2.seme.equals("CUORI") || c2.seme.equals("QUADRI") || c2.seme.equals("FIORI")){
                        return(-1);
                    }
                    else if(c2.seme.equals("PICCHE")){
                        return(0);
                    }
            }
        }
        return(0);
    }
}
```

- Directory

È un controllo sui nomi delle directory: vengono ordinate prima in base all'ordine alfanumerico, poi si valuta, per lo stesso nome, il numero (solitamente il sistema operativo non fa così…), esempio: dir1, dir10, dir11, zac1

```
 5 public class CompareByDirectoryName implements Comparator{
 6
 7     // -1 => NON ha numeri finali
 8     // >=0 => ha un numero finale a quell'indice
 9     public static int firstIndexOfInteger(String str){
10         for(int i=str.length()-1;i>=0;i--){
11             if(!Character.isDigit(str.charAt(i))){
12                 if(i==str.length()-1) //l'ultimo carattere è una lettera: non mi interessa
13                     return(-1);
14                 else{   //non è l'ultimo carattere: la stringa ha almeno un numero
15                     return(i+1);
16                 }
17             }
18         }
19         return(0);
20     }
21
22     //ritorna la parte numerica della stringa
23     //se non c'è ritorna -1 => sarà considerato il primo
24     private static int numericalNameExtrapolator(String str){
25         int index = firstIndexOfInteger(str);//guardo dove inizia la parte numerica finale
26         if(index==-1){
27             return(-1);
28         }
29         else{
30             String temp = str.substring(index,str.length());//prendo solo i numeri
31             return(Integer.parseInt(temp));
32         }
33     }
35     //ritorna la parte verbosa della stringa
36     //se non c'è ritorna ""
37     //nell'ordine vengono prima i soli numeri e poi il resto
38     private static String verbalNameExtrapolator(String str){
39         int index = firstIndexOfInteger(str);
40         if(index==-1){//se non ha numeri ritorna tutta la stringa
41             return(str);
42         }
43         else{
44             return(str.substring(0,index));
45         }
46     }
47
48     @Override public  int compare(Object o1, Object o2) {
49         if(!(o1 instanceof String) || !(o2 instanceof String))
50             System.exit(1);
51
52         String dir1 = ((String) o1);
53         String dir2 = ((String) o2);
54
55         //numero finale + nome verbale
56         int ind1 = numericalNameExtrapolator(dir1);
57         int ind2 = numericalNameExtrapolator(dir2);
58         String vName1 = verbalNameExtrapolator(dir1);
59         String vName2 = verbalNameExtrapolator(dir2);
60         if(vName1.equals(vName2)){
61             //Le directry hanno stesso nome verbale
62             if(ind1>ind2){
63                 return(1);
64             }
65             else if(ind1==ind2){
66                 return(0);
67             }
68             else{
69                 return(-1);
70             }
71         }
72         else{
73             //Le directory hanno valore diverso:
74             return(vName1.compareTo(vName2));
75         }
76     }
```

- Tombola

```
 3 public class Cartella {
 4     static private int CARTELLE_EMESSE=12345;
 5
 6     private int id;
 7     private final int[][] cartellaNumeri = new int[3][5];
 8     private final boolean[][] cartellaCoperture = new boolean[3][5];
 9
10     public Cartella(){
11         Sacchetto p = new Sacchetto();
12         for(int i=0;i<3;i++){
13             for(int j=0;j<5;j++){
14                 cartellaNumeri[i][j] = p.estrai();
15                 cartellaCoperture[i][j] = false;//nessun numero è stato coperto
16             }
17         }
18         id = CARTELLE_EMESSE++; //imposta l'id e aumenta il numero di cartelle emesse
19     }
20
21     public Cartella(int[] numbers){
22         if(numbers.length!=15){ //controllo se ho abbsatnza elementi
23             System.exit(1);
24         }
25
26         //inserisce i 15 elementi nella cartella
27         for(int i=0;i<3;i++){
28             for(int j=0;j<5;j++){
29                 cartellaNumeri[i][j] = numbers[(i*5)+j];
30                 cartellaCoperture[i][j] = false;
31             }
32         }
33
34         id = CARTELLE_EMESSE++;
35     }
49     public boolean checkTombola(){
50         for(int i=0;i<3;i++){
51             for(int j=0;j<5;j++){
52                 if(cartellaCoperture[i][j]==false){//devono essere tutti true
53                     return(false);
54                 }
55             }
56         }
57         return(true);
58     }
59
60     @Override public String toString(){
61         String result = "";
62         for(int i=0;i<3;i++){
63             for(int j=0;j<5;j++){
64                 if(cartellaCoperture[i][j]==true){
65                     result += "["+cartellaNumeri[i][j]+"] ";
66                 }
67                 else{
68                     result += cartellaNumeri[i][j]+" ";
69                 }
70
71             }
72             result+="\n";
73         }
74         return(result);
75     }
76
77     public int getCartellaId(){
78         return(this.id);
79     }
```

cartellaCoperture array di boolean dove segno true gli elementi coperti

```java
public class Giocatore {
    public String nome;

    private List <Cartella> cartelle;

    public Giocatore(String nome){
        this.nome = nome;

        cartelle = new ArrayList();
        cartelle.add(new Cartella());
    }

    public Giocatore(String nome, int numCartelle){
        this.nome = nome;
        cartelle = new ArrayList();

        if(numCartelle<=0){
            cartelle.add(new Cartella());
        }
        else{
            for(int i=0;i<numCartelle;i++){
                cartelle.add(new Cartella());
            }
        }
    }

    public boolean copri(int n){
        boolean result = false;


        for(Cartella p:cartelle){
            boolean cartellaResult = p.copri(n);
            result = result || cartellaResult;
        }

        if(result)
            System.out.println(nome+" copre "+n);
        return(result);
    }
    public boolean checkTombola(){
        boolean result=false;
        List <Cartella> cartelleVincenti = new ArrayList();

        for(Cartella p:cartelle){
            boolean cartellaResult = p.checkTombola();
            result = result || cartellaResult;
            if(cartellaResult){
                cartelleVincenti.add(p);
            }
        }

        if(!cartelleVincenti.isEmpty()){
            vince(cartelleVincenti);
        }

        return(result);
    }

    private void vince(List <Cartella> cartelleVincenti){
        System.out.println("\n\n *******"+nome + " ha vinto!!!******");
        for(Cartella p:cartelleVincenti){
            System.out.println(p);
            System.out.println("id cartella: "+ p.getCartellaId()+"\n\n");
        }


    }

    @Override public String toString(){
        String result=nome+"\n";
        for(Cartella p:cartelle){
            result+=p.toString()+"\n";
        }
       return(result);
    }
```

Imposta a "true" sugli indici dove c'è il numero Ritorna true se viene coperto

```java
public class Tabellone {
    private final Cartella[] tabellone = new Cartella[6];

    public Tabellone(){
        tabellone[0] = new Cartella(new int[]{1,2,3,4,5,11,12,13,14,15,21,22,23,24,25});
        tabellone[1] = new Cartella(new int[]{6,7,8,9,10,16,17,18,19,20,26,27,28,29,30});
        tabellone[2] = new Cartella(new int[]{31,32,33,34,35,41,42,43,44,45,51,52,53,54,55});
        tabellone[3] = new Cartella(new int[]{36,37,38,39,40,46,47,48,49,50,56,57,58,59,60});
        tabellone[4] = new Cartella(new int[]{61,62,63,64,65,71,72,73,74,75,81,82,83,84,85});
        tabellone[5] = new Cartella(new int[]{66,67,68,69,70,76,77,78,79,80,86,87,88,89,90});
    }

    public boolean copri(int n){
        boolean result = false;
        for(Cartella p:tabellone){
            result = result || p.copri(n);
        }
        return(result);
    }

    public boolean checkTombola(){
        boolean result = false;
        for(Cartella p:tabellone){
            result = result || p.checkTombola();
        }
        return(result);
    }

    @Override public String toString(){
        String result ="";
        for(Cartella p:tabellone){
            result += p.toString()+"\n";
        }
        return(result);
    }
```

```java
public class Tombola {
    private int numeroEstrazioni;

    public Croupier banco;
    public Set <Giocatore> giocatori;

    public Tombola(){
        banco = new Croupier();
        giocatori = new HashSet();
        numeroEstrazioni = 1;
    }

    public boolean addGiocatore(Giocatore p){
        try{
            giocatori.add(p);
            return(true);
        }
        catch(Exception e){
            return(false);
        }
    }

    private void copriGiocatori(int n){
        for(Giocatore p: giocatori){
            p.copri(n);
        }
    }

    private List <Giocatore> checkTombolaGiocatori(){
        List vincitori = new LinkedList();
        for(Giocatore p: giocatori){
            if(p.checkTombola()){
                vincitori.add(p);
            }
        }
        return(vincitori);
    }
    public boolean round(){
        int estrazione = banco.estrazione();
        System.out.println("\nEstrazione #"+numeroEstrazioni++ +": "+estrazione);

        //copro le cartelle dei giocatori e del banco
        banco.copri(estrazione);
        copriGiocatori(estrazione);

        //verifico se qualcuno vince
        if(banco.checkTombola()){
            return(true);
        }
        else{
            boolean temp=false;
            for(Giocatore p:checkTombolaGiocatori()){
                temp=true;
            }
            return(temp);
        }
    }
```

Fa estrazioni e controlla se qualcuno vince

```java
public class Croupier {
    private final Tabellone tabellone;
    private final Sacchetto sacchetto;

    public Croupier(){
        tabellone = new Tabellone();
        sacchetto = new Sacchetto();
    }

    public int estrazione(){
        return(sacchetto.estrai());
    }

    public boolean copri(int n){
        boolean result = tabellone.copri(n);
        if(result){
            System.out.println("Il banco copre "+n);
        }
        return(result);
    }

    public boolean checkTombola(){
        boolean result = tabellone.checkTombola();
        if(result){
            System.out.println("**********Vince il BANCO!*********");
            System.out.println(tabellone);
        }
        return (result);
    }

    @Override public String toString(){
        return(tabellone.toString());
    }
}
```

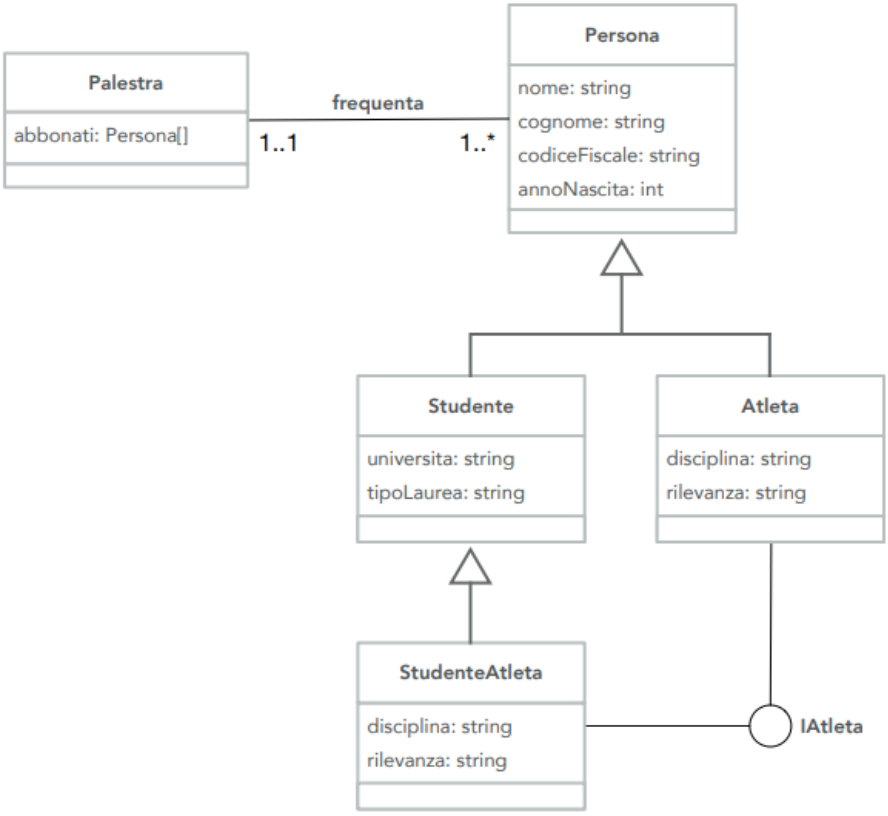
```java
public class Sacchetto {
    public final List <Integer> sacchetto = new ArrayList();

    //aggiungo i 90 numeri
    public Sacchetto(){
        for(int i=0;i<90;i++){
            sacchetto.add(i+1);
        }
    }

    public int estrai(){    //estrae a caso SENZA ripetizione
        if(sacchetto.size()>0){
            Collections.shuffle(sacchetto);
            return(sacchetto.remove(0));
        }
        else return(0);
    }
}
```

- Database palestra

Per quanto strano, con questo diagramma se aggiungo nuove classi (ProfessoreAtleta, GenitoreAtleta…) la tariffa viene gestita nella classe stessa, evitando di andare a toccare le altre classi. Inoltre dovessimo controllare le istanze ci basterebbe verificare se una persona è di istanza “IAtleta” per verificare se è un atleta.



- MiniCalculator

```
public class MiniCalculator extends Application {

    @Override public void start(Stage primaryStage) {

        //creo i 4 pulsanti in fila con un TilePane
        TilePane pulsanti = new TilePane();
        pulsanti.setPrefColumns(4);

        Button piu = new Button("+");
        Button meno = new Button("-");
        Button per = new Button("x");
        Button diviso = new Button(":");

        piu.setMinWidth(100);
        meno.setMinWidth(100);
        per.setMinWidth(100);
        diviso.setMinWidth(100);

        pulsanti.getChildren().addAll(piu,meno,per,diviso);

        //creo i text input e i pulsanti in un gridpane
        GridPane input = new GridPane();

        Text title= new Text("MINICALCULATOR");
        title.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR,20));
        GridPane.setHalignment(title, HPos.CENTER); //lo imposto al centro

        TextField n1 = new TextField();
        TextField n2 = new TextField();
        TextField n3 = new TextField();
        n3.setDisable(true);    //lo rendo NON modificabile
        n3.setStyle("-fx-opacity: 1;");

        //inserisco le componenti nel GridPane
        input.add(title,0,1);
        input.add(n1, 0,2);
        input.add(n2, 0,3);
        input.add(pulsanti, 0,4);
        input.add(n3, 0,5);
        //metto insieme tutto in un borderPane
        BorderPane calcolatrice = new BorderPane();

        Button clear = new Button("Clear");
        //il pulsante svuota le textfield
        clear.addEventHandler(ActionEvent.ACTION, new EventHandler(){
            @Override public void handle(Event e){n1.clear();n2.clear();n3.clear();}});

        Text left = new Text("Left");
        Text copyright = new Text("magicSoftware");

        BorderPane.setAlignment(input, Pos.CENTER_RIGHT);
        BorderPane.setAlignment(clear, Pos.CENTER_RIGHT);
        BorderPane.setAlignment(left, Pos.CENTER_RIGHT);
        BorderPane.setAlignment(copyright, Pos.CENTER_RIGHT);

        calcolatrice.setMargin(left, new Insets(40,40,40,40));
        calcolatrice.setMargin(clear, new Insets(40,40,40,40));

        //aggiungo gli elementi
        calcolatrice.setCenter(input);
        calcolatrice.setLeft(left);
        calcolatrice.setRight(clear);
        calcolatrice.setBottom(copyright);

        //creo gli eventhandler: lo faccio ora perchè mi servivano i textfield
        piu.addEventHandler(ActionEvent.ACTION, new EventHandler(){@Override
                            public void handle(Event e){
                                if(controllaTextField(n1) ||controllaTextField(n2) || n1.getText().isEmpty() || n2.getText().isEmpty()){
                                        Alert alert = new Alert(AlertType.ERROR);
                                        alert.setTitle("ERRORE");
                                        alert.setHeaderText("Ci sono dei valori non accettati!");
                                        alert.showAndWait();
                                }
                                else{
                                        double ris = Double.parseDouble(n1.getText())+Double.parseDouble(n2.getText());
                                        n3.setText(Double.toString(ris));
                                }
                            }
        });
        per.addEventHandler(ActionEvent.ACTION, new EventHandler(){@Override
                            public void handle(Event e){if(controllaTextField(n1) ||controllaTextField(n2) || n1.getText().isEmpty() ||
    n2.getText().isEmpty()){Alert alert = new Alert(AlertType.ERROR);alert.setTitle("ERRORE");alert.setHeaderText("Ci sono dei valori non
    accettati!");alert.showAndWait();}else{double ris = Double.parseDouble(n1.getText())*Double.parseDouble(n2.getText());n3.setText(Double.toString(ris));}}});
        meno.addEventHandler(ActionEvent.ACTION, new EventHandler(){@Override
                            public void handle(Event e){if(controllaTextField(n1) ||controllaTextField(n2) || n1.getText().isEmpty() ||
    n2.getText().isEmpty()){Alert alert = new Alert(AlertType.ERROR);alert.setTitle("ERRORE");alert.setHeaderText("Ci sono dei valori non
    accettati!");alert.showAndWait();}else{double ris = Double.parseDouble(n1.getText())-Double.parseDouble(n2.getText());n3.setText(Double.toString(ris));}}});
        diviso.addEventHandler(ActionEvent.ACTION, new EventHandler(){@Override
                            public void handle(Event e){if(controllaTextField(n1) ||controllaTextField(n2) || n1.getText().isEmpty() ||
    n2.getText().isEmpty()){Alert alert = new Alert(AlertType.ERROR);alert.setTitle("ERRORE");alert.setHeaderText("Ci sono dei valori non
    accettati!");alert.showAndWait();}else{double ris = Double.parseDouble(n1.getText())/Double.parseDouble(n2.getText());n3.setText(Double.toString(ris));}}});

        Scene scene = new Scene(calcolatrice);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    //ritorna "true" se il textfield contiene lettere
    //non gestisce il campo vuoto, che funziona solo DENTRO l'eventhanlder
    private boolean controllaTextField(TextField text){
        String testo = text.getText();
        for(char c:testo.toCharArray()){
            if(Character.isLetter(c)){
                System.out.println("Lettera: "+c);
                return(true);
            }
        }
        return(false);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```