Documentazione del progetto “ASCIIArt”

Gruppo “Mediglia”:

Patrizia Fornoni, Federico Marcolongo

02/2020

Indice

Modifiche e aggiunte

In questo file di documentazione (e in particolare nei diagrammi UML) le classi, gli attributi e i metodi aggiunti oppure modificati rispetto alla traccia dell'esame sono evidenziati in verde. In particolare, le scelte principali da noi operate nello sviluppo del progetto sono elencate qui sotto.

Classe Canvas

Canvas possiede una classe annidata (statica e privata) Frame, le cui istanze sono responsabili del mantenimento in memoria delle griglie di caratteri. Un canvas deve implementare un metodo undo(k: int) che permetta di ritornare a uno stato precedente della griglia. Per implementare questa funzionalità abbiamo associato a ogni canvas una pila di Frame, registroFrame. Il metodo undo opera chiamando il metodo pop() di Stack k volte consecutive, mentre dell'aggiunta al registro del frame corrente si occupa il metodo addToHistory() di Canvas. Il frame corrente si trova sempre in cima alla pila ed è puntato dall'attributo "corrente" di Canvas. La classe Frame è privata perché le funzionalità sono proprie della classe Canvas e non è desiderabile che sia visibile all'esterno di essa. Frame è anche statica perché non necessita di accedere a membri di Canvas, e questa caratterizzazione rende più efficiente il suo funzionamento.

Consequenzialità degli eventi di Drag

Nel testo dell'esame si richiede che agli eventi di tipo DragStart e DragEnd sia imposto un controllo: a un DragStart devono seguire necessariamente dei MouseMove e un DragEnd, così come un DragEnd può essere chiamato soltanto dopo un DragStart seguito da MouseMove. L'applicazione ASCIIArt fornita implementa già questo modo di funzionare, e non permette una consequenzialità tra gli eventi che non sia compatibile con quanto scritto sopra. Tuttavia, un'ulteriore richiesta dell'esame è che la gestione degli eventi sia indipendente dall'utilizzo di ASCIIArt, e per questo motivo abbiamo implementato tale funzionamento direttamente nelle sottoclassi di EventoDiMouse. Prima di tutto, abbiamo introdotto due nuove eccezioni non controllate: DragStartWithoutDragEndException e DragEndWithoutDragStartException. Abbiamo definito un attributo statico "dragging", di tipo boolean, di EventoDiMouse. Il costruttore di DragStart controlla che dragging non sia già true, ovvero non sia già attivo un "processo di drag". Se così è, lancia una DragStartWithoutDragEndException, poiché il processo precedente non si è concluso con un DragEnd. Viceversa, se EventoDiMouse.dragging è inizialmente falso, il funzionamento è regolare e DragStart imposta dragging a true. Se durante un processo di drag avviene qualcosa che non sia un MouseMove o un DragEnd, ovvero, nel nostro caso, un MouseClick, il costruttore di tale evento lancia la medesima eccezione, poiché il processo non si è concluso con un DragEnd. Se infine un DragEnd viene istanziato al di fuori di un processo di drag (EventoDiMouse.dragging == false), il suo costruttore lancia una DragEndWithoutDragStartExcpetion. Eventi di tipo MouseMove non influenzano e non sono influenzati da tale processo.

Gerarchia di StrumentoDiDisegno

Abbiamo apportato modifiche alla gerarchia degli delle sottoclassi di Strumento: il testo dell'esame richiede di ridefinire il metodo ricevi nelle sottoclassi concrete di StrumentoDiDisegno. Abbiamo preferito, dove opportuno, separare la gestione degli input da mouse dall'effettivo disegno delle figure geometriche. Abbiamo dunque introdotto tre classi astratte, StrumentoDiDisegnoDrag, StrumentoDiDisegnoCentro e StrumentoDiDisegnoLibero, nelle quali viene ridefinito il metodo void ricevi(e: EventoDiMouse) di Strumento in modo consistente alla gestione degli input desiderata da ogni strumento concreto. Ognuna di queste classi dichiara inoltre un metodo astratto disegna, con la

segnatura opportuna per il particolare stile di input desiderato per il disegno di una figura. Uno strumento concreto che eredita da una di queste classi si impegna a ridefinire il metodo disegna, che determina la stampa sulla griglia della figura corrispondente al particolare strumento, date in input le necessarie informazioni. Abbiamo introdotto le classi concrete Quadrato (che eredita da StrumentoDiDisegnoCentro) e Pennello (che eredita da StrumentoDiDisegnoLibero) per dimostrare la facilità nell'introdurre nuove classi avendo separato a priori la gestione degli input dal disegno. Lo strumento Poligonale è l'unico ad essere sottoclasse (concreta) diretta di StrumentoDiDisegno. Un diagramma UML con la gerarchia delle sottoclassi di Strumento è fornito nella sezione "Dettaglio delle classi".

Evidenziatori e riquadro di selezione

Per facilitare l'utilizzo dell'applicazione ASCIIArt per il disegno di figure geometriche abbiamo deciso di implementare una funzionalità non richiesta dal testo dell'esame. Abbiamo introdotto gli attributi statici e costanti (final) di tipo char EVIDENZIATORE in StrumentoDiDisegno e TRATTOSELEZIONE in StrumentoDiSelezione. Il primo è utilizzato dagli strumenti concreti di tipo StrumentoDiDisegnoCentro per evidenziare temporaneamente il centro della figura da disegnare (il luogo del primo click). Il secondo è utilizzato dagli strumenti di selezione per disegnare temporaneamente i bordi del riquadro di selezioni. Entrambe le modifiche al canvas vengono annullate sia nel caso in cui l'utilizzo dello strumento vada a buon fine, sia nel caso in cui lo strumento venga ripristinato allo stato iniziale. La scelta del porre final i due attributi è dovuta al fatto che si suppone che nell'utilizzo dell'applicazione la scelta dei tratti sopra menzionata non spetti all'utente, quanto allo sviluppatore o al fornitore del servizio.

Metodo reset di Strumento

Abbiamo introdotto un metodo astratto reset() di Strumento, le cui sottoclassi concrete si impegnano a ridefinire e implementare in modo opportuno, per la gestione del reset dello stato dello strumento. In alcuni casi l'implementazione è compiuta da una classe astratta intermedia (es.: StrumentoDiDisegnoDrag), nel caso in cui lo stato dello strumento sia completamente determinato da attributi delle sue superclassi. Il metodo reset viene sempre chiamato dal metodo attiva(char) di Toolbox. In questo modo, quando si cambia lo strumento selezionato, lo strumento precedente torna sempre allo stato neutro. In alcuni casi questo si traduce anche in azioni sul canvas. Per esempio, gli strumenti di selezione disegnano sul canvas un riquadro di selezione. La chiamata di reset fa tornare allo stato precedente del canvas, in cui il riquadro di selezione non appare.

Metodi ricevi e azione di StrumentoDiSelezione

Come nel caso di StrumentoDiDisegno, abbiamo separato la gestione degli input (la definizione del riquadro di selezione e la cattura dell'action point) dall'effettiva azione dello strumento. Abbiamo dunque ridefinito il metodo astratto ricevi di Strumento direttamente in StrumentoDiSelezione, dove è dichiarato un metodo astratto disegna, che viene ridefinito da ciascun particolare strumento di selezione.

Strumento Salva

Abbiamo introdotto un nuovo strumento che eredita direttamente da Strumento per due motivi: non solo appare come l'utilizzo più naturale che si possa fare degli stream di Java per questo tipo di progetto, ma abbiamo inoltre ritenuto che per un'applicazione utilizzata da un utente per disegnare sia desiderabile il fatto di poter salvare le proprie opere sul disco. Una volta selezionato lo strumento, un click destro sul canvas salva il frame corrente come "ASCIIArt (X).txt", dove X è un numero intero ("ASCIIArt.txt" se X==0). Un click sinistro fa la stessa cosa dopo aver incrementato X, in modo da non

sovrascrivere un file salvato in precedenza. X, che nel codice è rappresentato dall'attributo statico fileCount, parte da -1 e viene incrementato a 0 già dal primo utilizzo di Salva, sia con click sinistro, sia con click destro.

Metodi equals, hashCode e toString

Abbiamo ritenuto che, ad eccezione del caso di Canvas, per ogni classe che abbiamo implementato fosse funzionale l'utilizzo dell' equals (e quindi dell'hashCode) ereditato dalla classe Object: per Evento, Strumento e Toolbox, infatti, riteniamo che ogni relativa istanza debba essere distinguibile dalle altre della stessa classe, anche nel caso in cui ne condivida lo stato (che è soggetto a continue modifiche durante l'esecuzione del programma). Abbiamo ritenuto che ridefinire il metodo equals di Frame non fosse necessario poiché tale classe è visibile solo all'interno di Canvas. La classe Canvas stessa è l'unica a ridefinire i metodi equals e hashCode, in modo che due canvas risultino indistinguibili da tali metodi nel caso in cui le relative griglie correnti contengano gli stessi caratteri nelle stesse posizioni, ignorando la differenza tra i registri dei due canvas. Questo è un comportamento desiderabile per l'utilizzo da parte di una classe esterna a Canvas per la quale il canvas sia descritto completamente dalla griglia corrente. Questo NON è un comportamento desiderabile nel caso in cui si voglia, da qualche parte, costruire e gestire per esempio una HashMap di Canvas. Abbiamo ritenuto tuttavia tale eventualità molto improbabile in un'applicazione di questo tipo, e abbiamo agito di conseguenza.

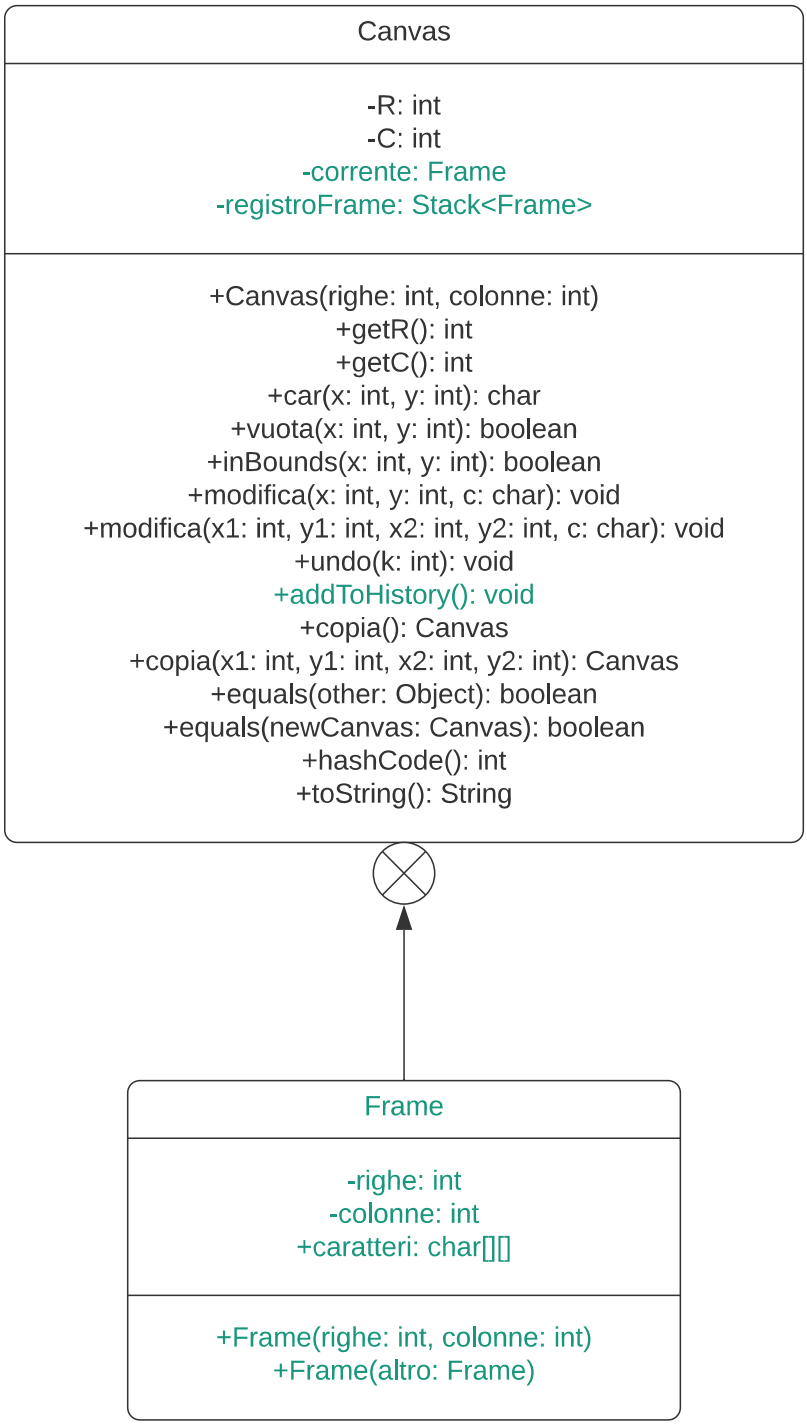
Il metodo hashCode di Canvas è stato ridefinito in modo da restituire la somma dei valori interi dei caratteri presenti in ogni cella della griglia. In questo modo canvas indistinguibili da parte di equals risultano indistinguibili anche da parte di hashCode (come desiderato). Inoltre piccole modifiche al canvas (persino la modifica di una sola cella) comportano la mutazione dell'hashCode del canvas.

I metodi toString di ogni classe (eccetto Frame, per motivi analoghi a quanto scritto sopra) sono stati ridefiniti in modo da identificare il tipo di oggetto che chiami tale metodo. Ciò dovrebbe fornire un buono strumento nel caso in cui sia necessario del debugging in un momento successivo nello sviluppo del codice.

Il metodo toString di Canvas è invece stato ridefinito in modo da soddisfare le richieste dell'esame (ed essere compatibile con la stampa su schermo dell'applicazione ASCIIArt).

Dettaglio delle classi

Classe Canvas



Il Canvas è una griglia di caratteri di forma rettangolare su cui possono agire diversi strumenti.

I campi di Canvas sono costituiti da due interi, corrispondenti al numero di righe (***R***) e colonne (***C***) del rettangolo, un Frame che rappresenta una singola griglia e fa riferimento all'attuale finestra di lavoro (***corrente***) e infine una pila di Frame che tiene traccia di tutte le eventuali modifiche apportate alla griglia (***registroFrame***). La pila è costruita attraverso la classe Stack presente nella libreria java.util.

Canvas contiene una classe annidata (statica e privata) Frame. Gli attributi del Frame sono due interi, **righe** e **colonne**, e un array di array di char, **caratteri**. La classe presenta due costruttori. Il primo costruttore riceve in ingresso due interi e inizializza **righe** e **colonne** rispettivamente. Inoltre inizializza caratteri ad un array di array di dimensioni righe x colonne contenente il carattere spazio. Il secondo costruttore riceve in ingresso un frame e ne costruisce un altro identico al frame in ingresso.

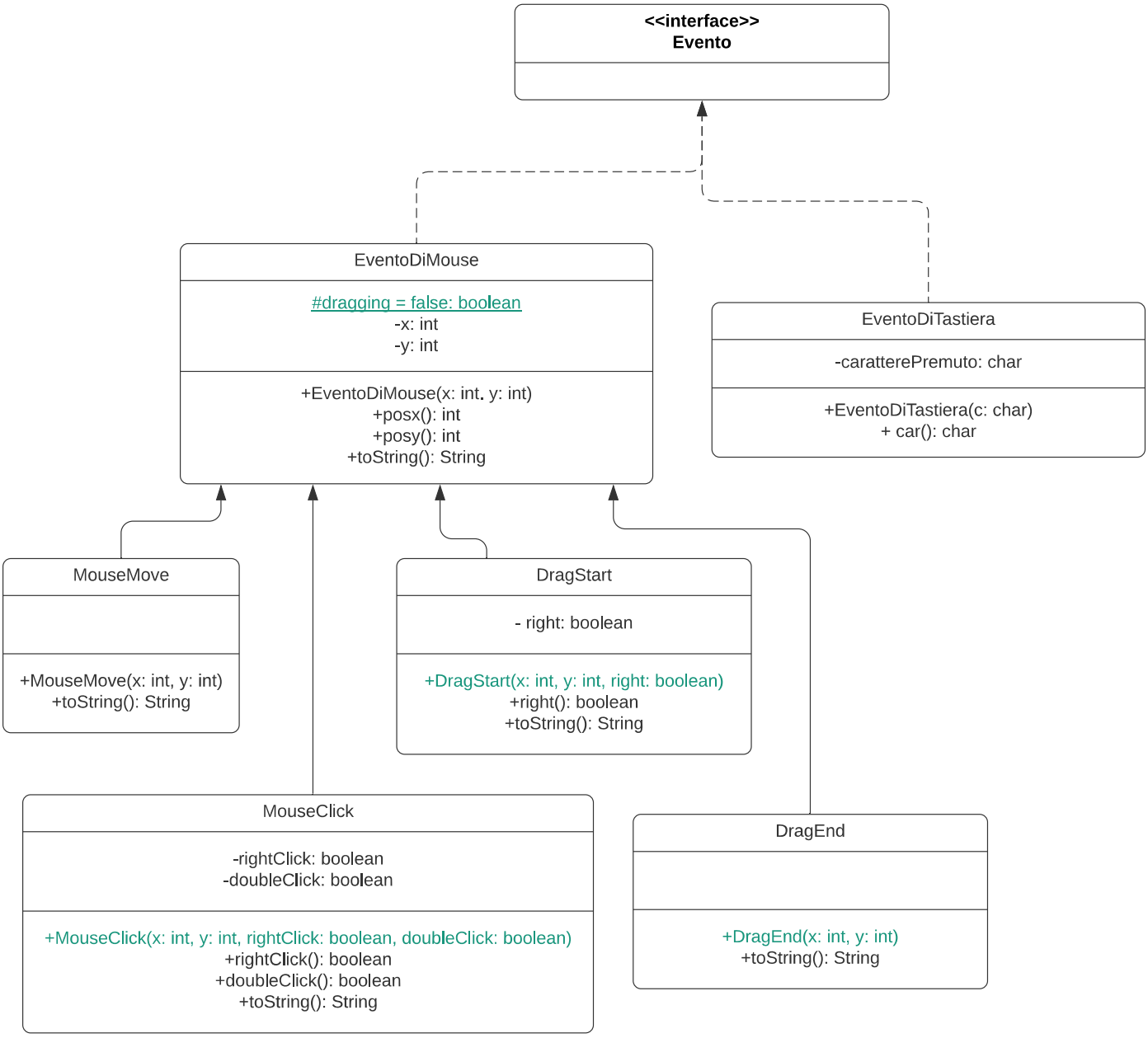
Il costruttore di Canvas riceve come parametri due interi, rispettivamente il numero di righe e di colonne della griglia. Quindi inizializza ***R*** e ***C*** con i parametri in ingresso, inizializza ***registroFrame*** attraverso il costruttore vuoto di Stack e gli aggiunge in *cima* un nuovo Frame di dimensioni R e C. Infine restituisce un puntatore al Frame che si trova in *cima* al ***registroFrame*** che viene memorizzato in ***corrente.***

I metodi di Canvas:

- **public int getR()** : restituisce il numero di righe del canvas su cui viene applicato il metodo.
- **public int getC()** : restituisce il numero di colonne del canvas su cui viene applicato il metodo.
- **public char car(int x, int y)** : restituisce il carattere corrispondente alla cella (x,y) di questo canvas. Nel caso in cui x e y non rispettino le dimensioni del canvas, 0<x<R e 0<y<C, viene visualizzato un opportuno messaggio di errore.
- **public boolean vuota(int x, int y)** : restituisce true se la cella (x,y) è vuota, cioè contiene il carattere spazio.
- **public boolean inBounds(int x, int y)** : restituisce true se e solo se i parametri x e y rispettano le dimensioni del canvas.
- **public void modifica(int x, int y, char c)** : sostituisce il carattere in posizione (x,y) con il carattere c in ingresso se e solo se i parametri x e y rispettano le dimensioni del canvas.
- **public void modifica(int x1, int y1, int x2, int y2, char c)** : traccia un segmento di estremi (x1,y1) e (x2,y2). Il metodo disegna solo la parte del segmento che rispetta le dimensioni del canvas. Per l'implementazione del metodo si confrontano la larghezza e l'altezza tra i due punti. Nel caso sia maggiore la prima le ascisse vengono incrementate di uno mentre l'incremento delle ordinate è riscalato secondo il coefficiente angolare della retta che passa per i due punti. Analogamente nel caso in cui sia maggiore l'altezza. Nel caso in cui i due punti coincidano il metodo si comporta come il metodo precedente per il punto in ingresso.
- **public void undo(int k)** : ripristina il canvas allo stato in cui si trovava prima delle ultime k modifiche. L'implementazione si basa sul metodo **pop()** della classe Stack che ad ogni chiamata rimuove il Frame che si trova in cima al **registroFrame**. Nel caso in cui **registroFrame** sia rimasto vuoto aggiungo un nuovo Frame di dimensioni pari a **R** e **C** del canvas su cui agisce il metodo. Infine **corrente** viene sovrascritto con il riferimento al Frame che si trova in cima alla pila.
- **public void addToHistory()** : questo metodo serve per memorizzare il Frame corrente prima che venga modificato. Il metodo agisce creando una copia del frame corrente, aggiungendola al registro e infine aggiornando **corrente** con tale Frame, che si trova in cima alla pila.
- **public Canvas copia()** : restituisce un nuovo canvas, copia di quello che chiama il metodo. Il registro del canvas restituito contiene solo il frame corrente e, sotto a questo, il frame vuoto.

- **public Canvas copia(int x1, int y1, int x2, int y2)** : Restituisce un nuovo canvas, copia del rettangolo di estremi (x1, y1), (x2,y2) del canvas che invoca il metodo. Il registro del canvas restituito contiene solo il frame corrente e, sotto questo, il frame vuoto.
- **public boolean equals (Object other)** : restituisce false se other non è istanza di Canvas. Nel caso in cui lo sia, chiama il metodo che segue per verificare l'uguaglianza tra i due oggetti di tipo Canvas.
- **public boolean equals(Canvas newCanvas)** : restituisce true se i frame correnti dei due Canvas sono uguali (stesso numero di righe, stesso numero di colonne e stessi caratteri nelle medesime posizioni).
- **public int hashCode() :** restituisce un intero pari alla somma dei caratteri del Frame corrente.
- **public String toString()** : restituisce una stringa corrispondente ai caratteri presenti nel Frame. Il carattere "a capo" serve per distinguere le varie righe del Frame.

Interfaccia Evento



Evento è una marker interface ed è implementata da due classi: EventoDiMouse ed EventoDiTastiera.

Classe EventoDiMouse

EventoDiMouse rappresenta il fatto che ci sia stata un'interazione tra il mouse e l'utente.

La classe presenta come attributi un boolean statico denominato dragging inizializzato a false che serve per controllare gli eventi di DragStart e DragEnd, infatti quest'ultimo si può verificare solo in seguito a un DragStart (intervallato da eventi MouseMove). Gli altri due attributi sono due campi interi, **x** e **y**, che indicano la posizione dell'evento all'interno del Canvas.

EventoDiMouse ha un unico costruttore che inizializza **x** e **y** con i due interi ricevuti in ingresso.

I metodi di EventoDiMouse sono:

- **public int posx()**: restituisce la coordinata x della posizione in cui si è verificato l'evento.
- **public int posy()**: restituisce la coordinata y della posizione in cui si è verificato l'evento.
- **public String toString()** : restituisce una stringa che esprime che si è verificato un evento di mouse in una determinata posizione.

Classe MouseMove

MouseMove rappresenta il fatto che il mouse è stato spostato e ora si trova in una data posizione.

MouseMove in quanto sottoclassse di EventoDiMouse, ha come unico costruttore quello ereditato dalla sua superclasse. Inoltre lascia invariato il campo dragging.

L'unico metodo che viene riscritto è **toString,** il quale non fa altro che specificare che l'evento di mouse in questione è di tipo MouseMove. Per il resto si comporta in modo analogo al metodo ereditato da EventoDiMouse.

Classe MouseClick

MouseClick rappresenta il fatto che l'utente ha premuto uno dei due tasti del mouse in una data posizione.

Oltre ai campi ereditati da EventoDiMouse presenta altri due campi di tipo boolean: **rightClick** e **doubleClick**. Questi due tasti rappresentano rispettivamente il fatto che è stato premuto il tasto destro e il fatto che ci sia stato un doppio click.

La classe ha unico costruttore che riceve in ingresso due interi che si riferiscono alla posizione e due boolean che si riferiscono agli eventi sopra descritti. Quello che fa il costruttore è inizializzare tutti gli attributi con i parametri ricevuti. Se al momento della costruzione EventoDiMouse.dragging è true, lancia una DragStartWithoutDragEndException (vedi Modifiche e aggiunte all'inizio del documento).

I metodi di MouseClick sono:

- **public boolean rightClick()**: restituisce true se e solo se è stato premuto il tasto destro del mouse.

- **public boolean doubleClick()** : restituisce true se e solo se si è verificato un doppio click.
- **public String toString()**: restituisce un'opportuna stringa che descrive il fatto che ci sia stato un click col mouse e il tipo di click, semplice(non dice nulla) o doppio, e destro o sinistro.

Classe DragStart

DragStart corrisponde al fatto che l'utente ha premuto uno dei due tasti del mouse, e tenendolo premuto, sta spostando il mouse.

La classe è sottoclasse di EventoDiMouse e per questo eredita tutti i suoi attributi. Oltre a questi contiene un campo di tipo boolean, **right.** Quest'ultimo vale true se e solo se l'evento di drag è stato chiamato col tasto destro del mouse.

Il costruttore di DragStart riceve come parametri due interi (indicanti la posizione) e un boolean che si riferisce al tasto che è stato premuto (destro o sinistro). Il costruttore inizializza tutti i campi con i parametri in ingresso e cambia il valore di **dragging** a true. Se però al momento della costruzione EventoDiMouse.dragging è già true, lancia una DragStartWithoutDragEndException (vedi Modifiche e aggiunte all'inizio del documento).

I metodi aggiunti o riscritti in DragStart sono:

- **public boolean right():** restituisce true se e solo se l'evento è stato chiamato col tasto destro del mouse.
- **public String toString()**: restituisce una stringa che descrive il fatto che c'è stato un evento di tipo DragStart e con quale tasto del mouse si è verificato.

Classe DragEnd

DragEnd rappresenta la fine di un evento di drag (momento in cui viene rilasciato il tasto del mouse).

Il costruttore riceve in ingresso i due interi che descrivono la posizione di fine drag. Inizializza quindi la posizione e ripristina il **dragging** a false. Se però al momento della costruzione EventoDiMouse.dragging è già false, lancia una DragEndWithoutDragStartException (vedi Modifiche e aggiunte all'inizio del documento).

L'unico metodo riscritto è:

- **public String toString()** : viene riscritto specificando che si tratta di un evento di tipo DragEnd.

Classe EventoDiTastiera

EventoDiTastiera rappresenta il fatto che l'utente abbia premuto un tasto sulla tastiera.

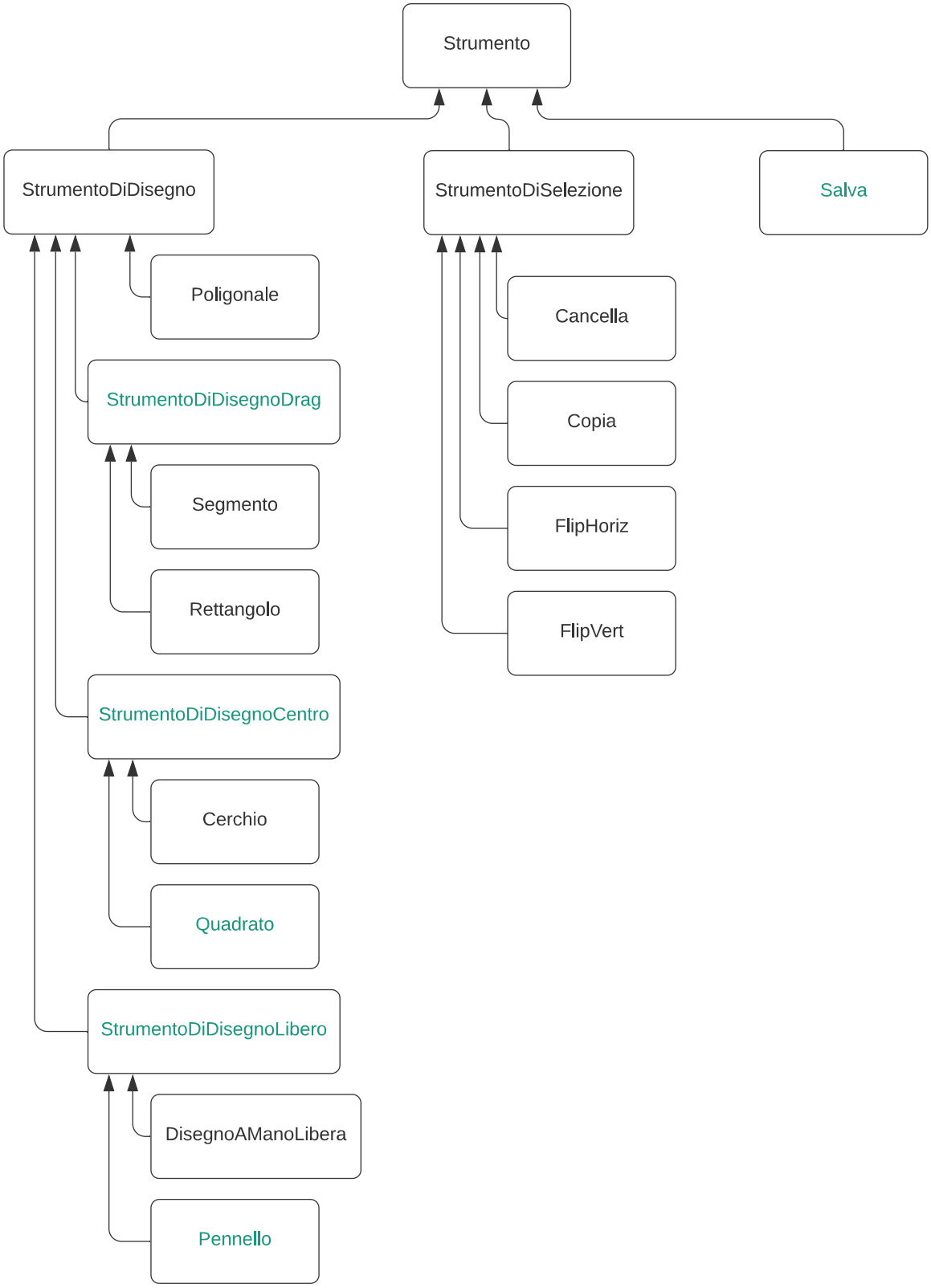
La classe presenta come unico attributo, **caratterePremuto,** di tipo char, che rappresentata il tasto della tastiera premuto dall'utente.

Il costruttore prende in ingresso un carattrere e inizializza **caratterePremuto**.

I metodi di EventoDiTastiera sono:

- **public char car()**: restituisce il carattere premuto.
- **public String toString()**: restituisce una stringa che descrive il fatto che si è trattato di un evento di tastiera e il carattere digitato dall'utente

Classe astratta Strumento



La classe Strumento reagisce ad alcuni eventi di mouse e come conseguenza modifica opportunamente il canvas.

Strumento ha un unico attributo **canvas** di tipo Canvas che sarà la base di lavoro per ogni strumento concreto definito come sottoclasse di Strumento.

Il costruttore riceve come parametro un canvas e inizializza di conseguenza **canvas**. Inoltre resetta sempre lo strumento allo stato iniziale grazie al metodo **reset**(), definito appunto in questa classe.

La classe risulta essere astratta dal momento che presenta due metodi astratti che verranno poi ridefiniti nelle varie sottoclassi.

I metodi di Strumento sono:

- **public abstract void ricevi(EventoDiMouse e)** : riceve un evento di tipo EventoDiMouse e modifica opportunamente il canvas su cui agisce strumento.
- **protected abstract void reset()** : resetta lo strumento allo stato iniziale.
- **public String toString()**: restituisce una stringa che descrive il tipo di strumento che sta lavorando sul canvas.

Strumento ha come sottoclassi: StrumentoDiDisegno, StrumentoDiSelezione e Salva.

Classe astratta StrumentoDiDisegno

Uno StrumentoDiDisegno è uno Strumento che serve per disegnare.

Questa classe presenta due attributi: **trattoDiDisegno** e **EVIDENZIATORE**. trattoDiDisegno è di tipo char e viene dichiarato static in modo tale da essere indipendente dall'oggetto stesso. Questo rappresenta il carattere utilizzato dallo strumento in questione per disegnare. EVIDENZIATORE è di tipo char e viene dichiarato anche lui static con l'aggiunta della clausola final, in modo tale che il suo valore non può essere mai modificato. EVIDENZIATORE viene utilizzato da alcuni strumenti di disegno per mostrare il punto del canvas in cui è avvenuto un determinato evento. Viene utilizzato per esempio dalla classe Cerchio per evidenziare la scelta del punto da utilizzare come centro della figura.

StrumentoDiDisegno ha come unico costruttore quello ereditato da Strumento.

I metodi sono:

- **public static void setTratto(char c)**: modifica il valore di trattoDiDisegno con c.
- **public static char getTratto()** : restituisce il carattere che si sta utilizzando per il disegno.
- **protected abstract void reset()** : resetta lo strumento alo stato iniziale.
- **public String toString()** :restituisce una stringa che descrive lo strumento che sta lavorando sul canvas, compreso del tratto utilizzato durante il disegno.

StrumentoDiDisegno ha come sottoclassi: StrumentoDiDisegnoDrag, Poligonale, StrumentoDiDisegnoCentro e StrumentoDiDisegnoLibero.

Classe astratta StrumentoDiDisegnoDrag

StrumentoDiDisegnoDrag è uno StrumentoDiDisegno che risponde solo a eventi di DragStart e DragEnd; e solo nel caso in cui questi eventi si verifichino con il tasto sinistro del mouse.

E' una classe astratta che viene estesa da due classi concrete: Segmento e Riquadro.

Ha come attributi un DragStart di nome **pIniziale**, che è il primo EventoDiMouse da registrare, e un DragEnd di nome **pFinale**, che corrisponde al secondo EventoDiMouse da registrare.

Il suo costruttore viene ereditato direttamente da Strumento.

I metodi di questa classe sono:

- **protected abstract void disegna(DragStart pIniziale, DragEnd pFinale)** : disegna la figura per trascinamento dato un punto iniziale e finale.
- **public void ricevi(EventoDiMouse e)** : si occupa di rispondere agli input di mouse. Nel caso **e** si tratti di un DragStart inizializza pIniziale e, dopo il controllo che questo si sia verificato effettivamente col tasto sinistro, modifica il canvas evidenziando il punto di inizio. Prima di modificare il canvas salva il Frame nel registro. Nel caso, invece, **e** sia istanza di DragEnd e ci sia stato prima un evento di DragStart si può precedere con il disegno effettivo della figura attraverso **disegna().** Alla fine vengono ripristinati DragStart a DragEnd a null.
- **protected void reset()** : ripristina DragStart e DragEnd a null e riporta il canvas allo stato iniziale.

- **public String toString()** : utilizza il toString della superclasse con l'aggiunta della stringa che si è trattato di uno strumento per trascinamento.

Classe segmento

Un segmento è uno StrumentoDiDisegnoDrag.

La classe è dichiarata di tipo final in modo tale che non può essere estesa da altre classi.

Il costruttore è ereditato da Strumento.

I metodi di Segmento sono:

- **protected void disegna(DragStart pIniziale, DragEnd pFinale)** : costruisce il segmento di estremi pIniziale e pFinale (corrispondenti ai due eventi di drag).
- **public String toString() :** restituisce una stringa che specifica che si è trattato dello strumento Segmento.

Classe Rettangolo

Rettangolo è uno strumento concreto di StrumentoDiDDisegnorag.

Il costruttore è ereditato da Strumento.

I metodi di Rettangolo sono:

- **protected void disegna(DragStart pIniziale, DragEnd pFinale)** : disegna un riquadro che ha come vertici contrapposti il pIniziale e il pFinale
- **public String toString()**: restituisce una stringa che specifica che si è trattato dello strumento Riquadro.

Classe Poligonale

Poligonale è uno StrumentoDiDisegno che serve per disegnare una sequenza di segmenti.

Poligonale ha come attributi, oltre a quelli ereditati, due MouseClick di nomi **primoClick** e **secondoClick** che determinano rispettivamente l'inizio della poligonale e ogni click successivo al primo. Questi click devono essere sempre semplici e con il tasto sinistro. Un eventuale doppio click sinistro riunisce l'ultimo punto con quello iniziale e termina la poligonale. Invece, un qualsiasi click col tasto destro resetta lo stato dello strumento. primoClick e secondoClick sono inizializzati a null.

Gli altri campi di Poligonale sono un boolean di nome **start** inizializzato a false che descrive l'inizio della poligonale. Infine sono presenti due campi interi per tenere traccia del punto di partenza della poligonale.

Il costruttore è ereditato da strumento.

I metodi di Poligonale sono:

- **public void ricevi(EventoDiMouse e)** : si occupa di gestire gli eventi di mouse in ingresso come sopra descritto e di agire modificando il Canvas. Nel caso in cui **e** sia un MouseClick destro viene richiamato il metodo **reset()**.
- **protected void reset()**: resetta lo strumento allo stato iniziale.
- **public String toString()**: restituisce una stringa che specifica che si è trattato dello strumento Poligonale.

Classe astratta StrumentoDiDisegnoCentro

StrumentoDiDisegnoCentro è uno StrumentoDiDisegno che serve per disegnare figure che rispondono a MouseClick effettuati col tasto sinistro. Il primo Click corrisponde al centro della figura mentre il secondo clik a un punto appartenente al perimetro della figura.

Le classi concrete che estendono StrumentoDiDisegnoCentro sono Quadrato e Cerchio.

Gli attributi sono due eventi di tipo MouseClick di nomi **centro** e **altro**, inizializzati a null.

Il costruttore è ereditato da Strumento.

I metodi di questa classe sono:

- **protected abstract void disegna(MouseClick centro, MouseClick altro)** : disegna la figura corrispondente allo strumento, dati il centro e un punto del perimetro.
- **public void ricevi(EventoDiMouse e)**: controlla se l'evento è effettivamente istanza di ClickMouse, e nel caso in cui centro sia stato effettivamente registrato, aggiunge il canvas corrente al registro e disegna la figura chiamando il metodo disegna. Nella fase di cancellazione, che si verifica sempre se il secondo click è destro, viene aggiunto un controllo nel caso in cui **centro** e **altro** coincidano. La fase di cancellazione richiama il metodo reset.
- **protected void reset()**: resetta lo strumento allo stato iniziale.
- **public String toString()** : restituisce una stringa che specifica che si è trattato di uno strumento con input di tipo centro.

Classe Quadrato

Quadrato è uno StrumentoDiDisegnoCentro concreto. Esso rappresenta una figura rettangolare avente come centro il primo click di mouse e passante per il secondo click di mouse. Quadrato è definito con il modificatore *final* in quanto non può essere esteso da nessuna classe.

Il costruttore è ereditato da Strumento.

L'unico aspetto sostanziale che cambia rispetto a StrumentoDiDisegnoCentro è l'implementazione del metodo disegna.

I metodi di quadrato sono:

- **protected void disegna(MouseClick centro, MouseClick altro)** : costruisce il rettangolo con centro il primo click e passante per altro.

- **public String toString()** :restituisce una stringa che specifica che si è trattato dello strumento Quadrato.

Classe Cerchio

Cerchio è uno StrumentoDiDisegnoCentro concteto. Esso rappresenta una circonferenza con centro il primo click e raggio pari alla distanza tra il centro e il secondo click. Cerchio è definito con il modificatore *final* in quanto non può essere esteso da nessuna classe.

Il costruttore è ereditato da Strumento.

I metodi di cerchio sono:

- **protected void disegna(MouseClick centro, MouseClick altro)** : il metodo disegna un quadrante di circonferenza alla volta utilizzando proprio l'equazione geometrica della circonferenza.
- **public String toString()** : restituisce una stringa che specifica che si è trattato dello strumento Cerchio.

Classe astratta StrumentoDiDisegnoLibero

StrumentoDiDisegnoLibero è uno StrumentoDiDisegno che traccia una figura seguendo i movimenti del mouse. Lo stato di StrumentoDiDisegnoLibero può essere attivo o non attivo. Nel primo caso risponde a eventi del tipo MouseMove, nel secondo caso non fa nulla. Lo stato di questo strumento si cambia facendo click semplice col tasto sinistro.

Gli attributi sono un boolean di nome **stato** che vale true se e solo se lo strumento è attivo; un MouseClick di nome click e un MouseMove di nome **move**. Gli eventi di mouse sono inizializzati a null mentre dtato è inizializzato a false.

Il costruttore è ereditato da Strumento.

I metodi di questa classe sono:

- **public void ricevi(EventoDiMouse e)** : riceve in ingresso un EventoDiMouse **e** e controlla se questo è istanza di MouseClick; nel caso in cui effettivamente lo sia e **stato** vale false prima di qualsiasi tipo di modifica salva il frame corrente nel registro. Lo stato dello strumento viene modificato secondo le modalità sopra descritte. Infine se lo strumento si trova nelle condizioni per poter disegnare (stato vale **true** e **e** è istanza di MouseMove) viene chiamato il metodo astratto **disegna(move)** dopo che l'informazione di **e** è stata salvata in **move**.
- **protected abstract void disegna(MouseMove move)** : disegna una linea continua seguendo il movimento del mouse.
- **protected void reset()** : resetta lo strumento alle condizioni iniziali.
- **public String toString()** : restituisce una stringa che specifica che si è trattato dello strumento con input di disegno libero.

Classe DisegnoAManoLibera

DisegnoAManoLibera è uno StrumentoDiDisegnoLibero che reagisce a movimenti di mouse disegnando nella cella in cui si trova. E' una classe concreta, definita con la clausola *final* in modo tale da non poter essere estesa da nessun'altra classe.

Il costruttore è ereditato da Strumento.

L'unica modifica sostanziale rispetto alla sua superclasse è l'implementazione del metodo **disegna(MouseMove move)** .

I suoi metodi sono:

- **protected void disegna(MouseMove move)** : disegna nella cella in cui si trova con il tratto corrente.
- **public String toString()** : restituisce una stringa che specifica che si tratta di uno strumento di disegno a mano libera.

Classe Pennello

Pennello è uno StrumentoDiDisegnoLibero che reagisce a movimenti di mouse disegnando dei cerchi centrati nella casella in cui si trova l'evento di mouse. E' una classe concreta, definita con la clausola *final* in modo tale da non poter essere estesa da nessun'altra classe.

L'attributo di Pennello è un intero di nome **raggio** e con modificatore *final.* raggio è inizializzato al valore due.

Il costruttore è ereditato da Strumento.

I metodi di questa classe sono:

- **protected void disegna(MouseMove move)** : disegna cerchi di raggio pari a **raggio** centrati nella cella in cui si trova il mouse.
- **public String toString()** : restituisce una stringa che descrive il fatto che si è trattato di uno strumento di tipo Pennello.

Classe astratta StrumentoDiSelezione

StrumentoDiSelezione è uno strumento che inizialmente reagisce solo a eventi di DragStart e DragEnd. Questi due eventi individuano un'areaSelezionata che rappresenta una porzione di canvas di forma rettangolare, i cui vertici sono dati dai due eventi di drag. Se dopo aver individuato un ' areaSelezionata si verifica qualsiasi evento che non sia un move o un click di mouse col tasto sinistro, la selezione viene persa e lo strumento ritorna allo stato iniziale.

Se invece si verifica un click col tasto sinistro, viene invocato il metodo astratto **azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint)**.

Gli attributi sono:

- **pIniziale** di tipo DragStart inizializzato a null.
- **pFinale** di tipo DragEnd inizializzato a null.
- **actionPoint** di tipo MouseClick inizializzato a null ,che descrive il momento in cui l'utente fa click col tasto sinistro dopo aver selezionato una determinata area del canvas.
- **areaSelezionata** di tipo canvas inizializzata a null.
- **TRATTODISELEZIONE** di tipo char che delimita il perimetro di areaSelezionata. E' inizializzato col carattere '.' e il suo valore non può essere modificato.

Il costruttore è ereditato da Strumento.

I metodi di StrumentoDiSelezione sono:

- **public void ricevi(EventoDiMouse e)** : gestisce gli input di mouse. Disegna l'areaSelezionata se si sono susseguiti correttamente gli eventi di DragStart e DragEnd. Infine, se **e** è istanza di MouseClick (tasto sinistro), richiama il metodo **azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint),** dopo aver tolto il riquadro di selezione e aver aggiunto il frame corrente al registro.
- **public abstract void azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint)** : svolge una qualche azione che sarà implementata nelle sottoclassi. La segnatura è stata modificata ma il tipo di funzionamento mantenuto.
- **protected void reset()**: resetta lo stato dello strumento in seguito a un click destro di mouse o alla fine di un'azione.
- **public String toString()**: restituisce una stringa che descrive il fatto che si è trattato di uno strumento di selezione.

Classe Cancella

Cancella è uno StrumentoDiSelezione che cancella il contenuto di areaSelezionata. Non è rilevante l'actionPoint.

Il costruttore è ereditato da strumento.

I metodi di Cancella sono:

- **public void azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint)**: modifica il contento di areaSelezionata con il carattere spazio.

- **public String toString()**: restituisce un'opportuna stringa per far capire che è stato applicato lo strumento Cancella.

Classe Copia

Copia è uno StrumentoDiSelezione che copia il contenuto di areaSelezionata nel rettangolo che ha l'actionPoint come vertice in alto a sinistra.

Il costruttore è ereditato da strumento.

I metodi di Cancella sono:

- **public void azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint)**: modifica il canvas sostituendo a partire dall'actionPoint, visto come vertice in alto a sinistra del rettangolo, il contenuto di areaSelezionata.
- **public String toString()**: restituisce un'opportuna stringa per far capire che è stato applicato lo strumento Copia.

Classi FlipHoriz e FlipVert

FlipHoriz e FlipVert sono due sottoclassi di StrumentoDiSelezione che applicano opportune simmetrie al rettangolo individuato da areaSelezionata. In particolare FlipHoriz applica una simmetria rispetto all'asse verticale che taglia il rettangolo in due parti uguali; analogamente FlipVert rispetto a quello orizzontale.

Il costruttore è ereditato da strumento.

I metodi di FlipHoriz e FlipVert sono:

- **public void azione(DragStart pIniziale, DragEnd pFinale, MouseClick actionPoint)**: modifica il canvas secondo quanto appena descritto.
- **public String toString()**: restituisce un'opportuna stringa per far capire che è stato applicato lo strumento FlipHoriz o, eventualmente, lo strumento FlipVert.

Classe Salva

Salva è uno Strumento che si occupa di salvare il Frame corrente in un file di testo di nome “ASCIIArt (X).txt, dove X corrisponde a un intero. Salva viene attivato da un MouseClick e, nel caso in cui questo sia sinistro, crea sempre un file nuovo per salvare la schermata, incrementando X; se destro, sovrascrive il file precedente.

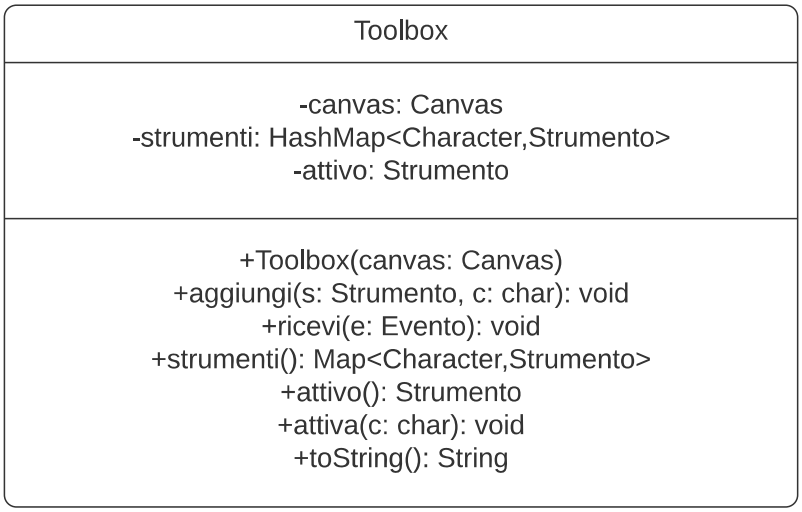
Gli attributi sono costituiti da un MouseClick di nome **click**, inizializzato a null, che serve per registrare il click dell’utente; e un intero di nome **fileCount**, il cui valore non viene modificato se lo strumento sovrascrive il file precedente.

Il costruttore è ereditato da strumento.

I metodi di Salva sono:

- **public void ricevi(EventoDiMouse e)** : riceve un evento di mouse e controlla se questo è istanza di MouseClick. Nel caso in cui lo sia, distingue i casi destro e sinistro. Se il click è destro, ma non è mai stato creato un file di testo (fileCount=-1), crea comunque un nuovo file perché non c’è nulla da sovrascrivere. Se il click è sinistro crea sempre un file nuovo su cui copiare il canvas corrente.
- **protected void reset()** : resetta lo strumento facendo puntare click a null.

Classe Toolbox



Toolbox rappresenta un insieme di strumenti in cui al più uno strumento è in ogni istante attivo. Ogni strumento è univocamente associato a un carattere.

I campi di Toolbox sono:

- **canvas** di tipo Canvas su cui gli strumenti di toolbox opereranno.
- **strumenti** di tipo HashMap<Character, Strumento> che associa uno Strumento a un'etichetta rappresentata da un carattere.
- **attivo** di tipo Strumento che rappresenta lo strumento attualmente in uso.

Il costruttore inizializza **canvas** con un canvas ricevuto in ingresso e crea una HashMap **strumenti** (di lunghezza e fattore di caricamento standard: 16 e 0.75).

I metodi di questa classe sono:

- **public void aggiungi(Strumento s, char c)** : aggiunge lo strumento **s** con etichetta **c** a strumenti, controllando se l'etichetta è già presente nell'HashMap. Nel caso in cui effettivamente lo sia, rimpiazza lo strumento già contenuto con il nuovo strumento passato come argomento.
- **public void attiva(char c)** : se strumenti contiene l'etichetta **c** ,attiva il corrispondente strumento, dopo aver resettato lo strumento precedentemente attivato nel caso in cui ci fosse. Quindi **attivo** contiene ora un riferimento al nuovo strumento.
- **public Strumento attivo()** :restituisce lo strumento attivo.
- **public Map<Character,Strumento> strumenti()** : restituisce l'HashMap strumenti.
- **public void ricevi(Evento e)** : riceve un evento.

  Se l'evento è di tastiera, controlla se il carattere premuto è presente o meno tra le etichette dell'HashMap,; nel caso in cui lo sia attiva lo strumento associato al carattere in questione. Se il carattere vale 'u', il canvas viene riportato allo stato precedente ad una singola azione.

Infine, se non si verifica nessuna di queste situazioni e lo strumento attivo è di tipo StrumentoDiDisegno, viene modificato il carattere di disegno con il carattere premuto.

Se l'evento è di mouse e c'è uno strumento attivo, viene passato l'evento di mouse allo strumento in questione che agirà secondo il proprio metodo **ricevi(EventoDiMouse e)**.

- **public String toString()** :restituisce una stringa opportuna che descrive gli strument**i** presenti nella ToolBox con la corrispettiva etichetta. Inoltre, riferisce lo strumento attualmente attivo nel caso ci sia.