A GEOMETRIC DRAWING PROGRAM

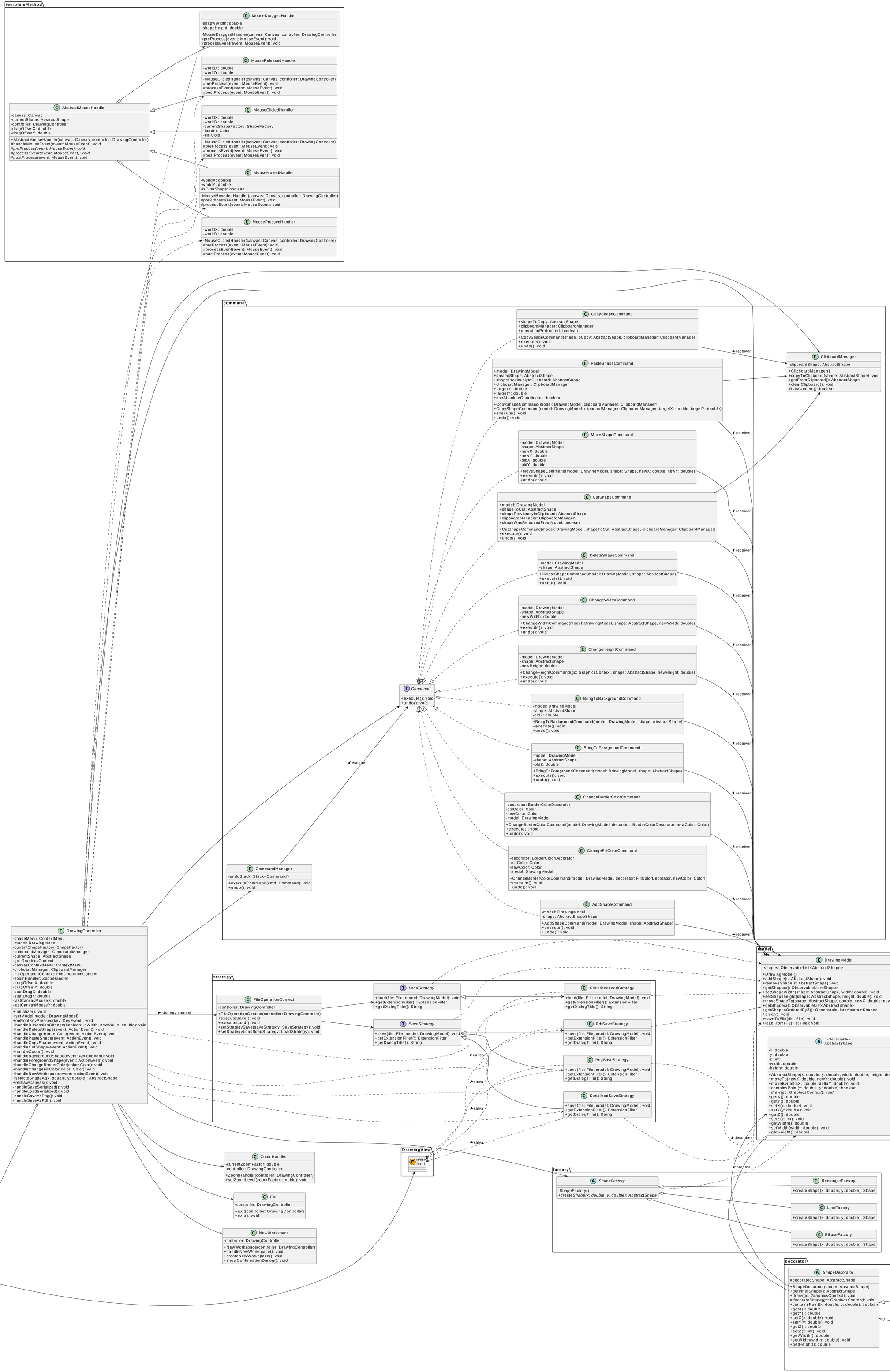
A.A. 2024/2025

Class Diagram

Docente:
Pierluigi Ritrovato

A cura del gruppo 5:

Apicella Antonio 0622702531
Celano Benedetta Pia 0622702558
Cuomo Carmine 0622702688
Guerra Simone 0622702675


templateMethod
MouseDraggedHandler
-shapvWidth: double
-shapeHeight: double
-MouseDraggedHandler(canvas: Canvas, controller: DrawingController)
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
MouseReleasedHandler
-worldX: double
-worldY: double
-MouseClicledHandler(canvas: Canvas, controller: DrawingController)
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
#postProcess(event: MouseEvent): void
AbstractMouseHandler
-canvas: Canvas
-currentShape: AbstractShape
-controller: DrawingController
-dragOffsetX: double
-dragOffsetY: double
+AbstractMouseHandler(canvas: Canvas, controller: DrawingController)
#handleMouseEvent(event: MouseEvent): void
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
#postProcess(event: MouseEvent): void
MouseClicledHandler
-worldX: double
-worldY: double
-currentShapeFactory: ShapeFactory
-border: Color
-fill: Color
-MouseClicledHandler(canvas: Canvas, controller: DrawingController)
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
#postProcess(event: MouseEvent): void
MouseMovedHandler
-worldX: double
-worldY: double
-isOverShape: boolean
-MouseMovededHandler(canvas: Canvas, controller: DrawingController)
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
MousePressedHandler
-worldX: double
-worldY: double
-MouseClicledHandler(canvas: Canvas, controller: DrawingController)
#preProcess(event: MouseEvent): void
#processEvent(event: MouseEvent): void
#postProcess(event: MouseEvent): void
command
CopyShapeCommand
+shapeToCopy: AbstractShape
+clipboardManager: ClipboardManager
+operationPerformed: boolean
+CopyShapeCommand(shapeToCopy: AbstractShape, clipboardManager: ClipboardManager)
+execute(): void
+undo(): void
receiver
ClipboardManager
-clipboardShape: AbstractShape
+ClipboardManager()
+copyToClipboard(shape: AbstractShape): void
+getFromClipboard(): AbstractShape
+clearClipboard(): void
+hasContent(): boolean
PasteShapeCommand
+model: DrawingModel
+pastedShape: AbstractShape
+shapePreviouslyInClipboard: AbstractShape
+clipboardManager: ClipboardManager
+targetX: double
+targetY: double
+useAbsoluteCoordinates: boolean
+CopyShapeCommand(model: DrawingModel, clipboardManager: ClipboardManager)
+CopyShapeCommand(model: DrawingModel, clipboardManager: ClipboardManager, targetX: double, targetY: double)
+execute(): void
+undo(): void
MoveShapeCommand
-model: DrawingModel
-shape: AbstractShape
-newX: double
-newY: double
-oldX: double
-oldY: double
+MoveShapeCommand(model: DrawingModel, shape: Shape, newX: double, newY: double)
+execute(): void
+undo(): void
CutShapeCommand
+model: DrawingModel
+shapeToCut: AbstractShape
+shapePreviouslyInClipboard: AbstractShape
+clipboardManager: ClipboardManager
+shapeWasRemovedFromModel: boolean
+CutShapeCommand(model: DrawingModel, shapeToCut: AbstractShape, clipboardManager: ClipboardManager)
+execute(): void
+undo(): void
DeleteShapeCommand
-model: DrawingModel
-shape: AbstractShape
+DeleteShapeCommand(model: DrawingModel, shape: AbstractShape)
+execute(): void
+undo(): void
ChangeWidthCommand
-model: DrawingModel
-shape: AbstractShape
-newWidth: double
+ChangeWidthCommand(model: DrawingModel, shape: AbstractShape, newWidth: double)
+execute(): void
+undo(): void
ChangeHeightCommand
-model: DrawingModel
-shape: AbstractShape
-newHeight: double
+ChangeHeightCommand(gc: GraphicsContext, shape: AbstractShape, newHeight: double)
+execute(): void
+undo(): void
Command
+execute(): void
+undo(): void
esegue
BringToBackgroundCommand
-model: DrawingModel
-shape: AbstractShape
-oldZ: double
+BringToBackgroundCommand(model: DrawingModel, shape: AbstractShape)
+execute(): void
+undo(): void
BringToForegroundCommand
-model: DrawingModel
-shape: AbstractShape
-oldZ: double
+BringToForegroundCommand(model: DrawingModel, shape: AbstractShape)
+execute(): void
+undo(): void
ChangeBorderColorCommand
-decorator: BorderColorDecorator
-oldColor: Color
-newColor: Color
-model: DrawingModel
+ChangeBorderColorCommand(model: DrawingModel, decorator: BorderColorDecorator, newColor: Color)
+execute(); void
+undo(): void
ChangeFillColorCommand
-decorator: BorderColorDecorator
-oldColor: Color
-newColor: Color
-model: DrawingModel
+ChangeBorderColorCommand(model: DrawingModel, decorator: FillColorDecorator, newColor: Color)
+execute(); void
+undo(): void
AddShapeCommand
-model: DrawingModel
-shape: AbstractShapeShape
+AddShapeCommand(model: DrawingModel, shape: AbstractShape)
+execute(): void
+undo(): void
CommandManager
-undoStack: Stack<Command>
+executeCommand(cmd: Command): void
+undo(): void
DrawingController
-shapeMenu: ContextMenu
-model: DrawingModel
-currentShapeFactory: ShapeFactory
-commandManager: CommandManager
-currentShape: AbstractShape
-gc: GraphicsContext
-canvasContextMenu: ContextMenu
-clipboardManager: ClipboardManager
-fileOperationContext: FileOperationContext
-zoomHandler: ZoomHandler
-dragOffsetX: double
-dragOffsetY: double
-startDragX: double
-startDragY: double
-lastCanvasMouseX: double
-lastCanvasMouseY: double
+initialize(): void
+setModel(model: DrawingModel)
+onRootKeyPressed(key: KeyEvent): void
+handleDimensionChange(boolean: isWidth, newValue: double): void
+handleDeleteShape(event: ActionEvent): void
+handleChangeBorderColor(event: ActionEvent): void
+handlePasteShape(event: ActionEvent): void
+handleCopyShape(event: ActionEvent): void
+handleCutShape(event: ActionEvent): void
+handleZoom(): void
+handleBackgroundShape(event: ActionEvent): void
+handleForegroundShape(event: ActionEvent): void
+handleChangeBorderColor(color: Color): void
+handleChangeFillColor(color: Color): void
+handleNewWorkspace(event: ActionEvent): void
+selecteShapeAt(x: double, y: double): AbstractShape
+redrawCanvas(): void
-handleSaveSerialized(): void
-handleLoadSerialized(): void
-handleSaveAsPng(): void
-handleSaveAsPdf(): void
strategy context
strategy
FileOperationContext
-controller: DrawingController
+FileOperationContext(controller: DrawingController)
+executeSave(): void
+executeLoad(): void
+setStrategySave(saveStrategy: SaveStrategy): void
+setStrategyLoad(loadStrategy: LoadStrategy): void
LoadStrategy
+load(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
SaveStrategy
+save(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
SerializedLoadStrategy
+load(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
PdfSaveStrategy
+save(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
PngSaveStrategy
+save(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
SerializedSaveStrategy
+save(file: File, model: DrawingModel): void
+getExtensionFilter(): ExtensionFilter
+getDialogTitle(): String
carica
salva
salva
salva
DrawingView
«File»
fxml
model
DrawingModel
-shapes: ObservableList<AbstractShape>
+DrawingModel()
+addShape(s: AbstractShape): void
+removeShape(s: AbstractShape): void
+getShapes(): ObservableList<Shape>
+setShapeWidth(shape: AbstractShape, width: double): void
+setShapeheight(shape: AbstractShape, height: double): void
+moveShapeTo(shape: AbstractShape, double: newX, double: newY): void
+getShapes(): ObservableList<AbstractShape>
+getShapesOrderedByZ(): ObservableList<AbstractShape>
+clear(): void
+saveToFile(file: File): void
+loadFromFile(file: File): void
«Serializable»
AbstractShape
-x: double
-y: double
-z: int
-width: double
-height: double
+AbstractShape(x: double, y: double, width: double, height: double)
+moveTo(newX: double, newY: double): void
+moveBy(deltaX: double, deltaY: double): void
+containsPoint(x: double, y: double): boolean
+draw(gc: GraphicsContext): void
+getX(): double
+getY(): double
+setX(x: double): void
+setY(y: double): void
+getZ(): double
+setZ(z: int): void
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
Rectangle
+Rectangle(x: double, y: double, width: double, height: double)
+draw(gc: GraphicsContext): void
Line
+Line(startX: double, startY: double, endX: double, endY: double)
+draw(gc: GraphicsContext): void
+getLenght(): double
+moveBy(deltaX: double, deltaY: double): void
+getEndX(): double
+setEndX(endX: double): void
+getEndY(): double
+setEndY(endY: double): void
Ellipse
+Ellipse(x: double, y: double, radiusX: double, radiusY: double)
+draw(gc: GraphicsContext): void
decorates
creates
factory
ShapeFactory
-ShapeFactory()
+createShape(x: double, y: double): AbstractShape
RectangleFactory
+createShape(x: double, y: double): Shape
LineFactory
+createShape(x: double, y: double): Shape
EllipseFactory
+createShape(x: double, y: double): Shape
decorator
ShapeDecorator
#decoratedShape: AbstractShape
+ShapeDecorator(shape: AbstractShape)
+getInnerShape(): AbstractShape
+draw(gc: GraphicsContext): void
#decorateShape(gc: GraphicsContext): void
+containsPoint(x: double, y: double): boolean
+getX(): double
+getY(): double
+setX(x: double): void
+setY(y: double): void
+getZ(): double
+setZ(z: int): void
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
BorderColorDecorator
-borderColor: Color
-red: double
-green: double
-blue: double
-alpha: double
+BorderColorDecorator(shape: Shape, color: Color)
+decorateShape(gc: GraphicsContext): void
FillColorDecorator
-fillColor: Color
-red: double
-green: double
-blue: double
-alpha: double
+FillColorDecorator(shape: Shape, color: Color)
+decorateShape(gc: GraphicsContext): void
ZoomHandler
-currentZoomFactor: double
-controller: DrawingController
+ZoomHandler(controller: DrawingController)
+setZoomLevel(zoomFactor: double): void
Exit
-controller: DrawingController
+Exit(controller: DrawingController)
+exit(): void
NewWorkspace
-controller: DrawingController
+NewWorkspace(controller: DrawingController)
+handleNewWorkspace(): void
+createNewWorkspace(): void
+showConfirmationDialog(): void
«Application»
GeometricDrawingApp
+start(primaryStage: Stage): void
+main(args: String[]): void

Updated Class Diagram

Durante questa sprint il team ha implementato diverse funzionalità utili per l'applicazione basandosi sullo sprint backlog e rispettando i principi SOLID e i Design Pattern.

Inoltre si deciso di effettuare del refactoring di alcune parti di codice prodotto nella sprint precedente così da recuperare del debito tecnico accumulato. Di conseguenza sono stati applicati due nuovi Design Pattern: **Template Method** e **Strategy,** al fine di separare le responsabilità per la logica di gestione degli eventi del mouse e di salvataggio/caricamento del foglio di lavoro.

Pattern Command

Sono stati implementate nuove classi Command per l'implementazione di nuove funzionalità e dunque nuove possibili interazioni dell'utente con l'applicazione. In particolare sono state implementate le classi: `BringToBackgroundCommand`, `BringToForegroundCommand`, `ChangeBorderColorCommand`, `ChangeFillColorCommand`, `PasteShapeCommand`, `CutShapeCommand`, `CopyShapeCommand`.

Inoltre è stata creata una nuova classe `ClipboardManager` che ha la responsabilità di gestire la clipboard dell'applicazione durante l'esecuzione. Questa classe funge da *receiver* per la classe `CopyShapeCommand` ed è anche una classe usata da altri command.

Classe ZoomHandler

Per implementare la funzionalità di zoom dell'area di lavoro è stata creata una classe apposita a cui è stata assegnata questa singola responsabilità. Il controller rimane focalizzato solo sulla gestione degli eventi mentre utilizza un oggetto `ZoomHandler` delegandogli le operazioni di zoom.

Classe Exit

Questa classe è stata implementata per gestire la logica da eseguire quando l'utente decide di chiudere l'applicazione. Il controller crea internamente una istanza di `Exit` a cui delega la responsabilità di gestire l'uscita dall'applicazione.

Classe NewWorkspace

In modo simile a quanto fatto per la classe `Exit` questa classe è stata creata per gestire la creazione di una nuova area di lavoro durante l'esecuzione dell'applicazione. Il controller crea una istanza di `NewWorkspace` della quale chiama il metodo per gestire la creazione di un nuovo foglio di lavoro quando l'utente seleziona l'opzione durante l'utilizzo dell'applicativo.

Pattern Template Method

Nella prima sprint le logiche degli handler per gli eventi del mouse erano state implementate all'interno del controller, ciò ha comportato un sovraccarico di responsabilità e alla creazione di codice duplicato. In questa sprint è stato applicato il pattern Template Method per la gestione dei mouse event, in particolare sono state create le classi `AbstractMouseHandler` che è la classe astratta che stabilisce quali sono gli steps definendone anche una implementazione di default. Le classi concrete sono: `MouseClicledHandler`, `MousePressedHandler`, `MouseDraggedHandler`, `MouseReleasedHandler`, `MouseMovedHandler`, ciascuna delle quali ha una singola responsabilità differente che viene utilizzata nel controller.

Pattern Strategy

Nella sprint precedente era stata assegnata al controller la responsabilità di caricare e salvare l'area di lavoro e dunque di operare con file di diversi formati. In questa sprint si è deciso di applicare il pattern Strategy per alleggerire le responsabilità del controller isolando il codice ed estraendo i differenti metodi di salvataggio e caricamento in classi apposite. Sono state create due gerarchie di strategie, una per il salvataggio (interfaccia `LoadStrategy`) e una per il caricamento (interfaccia `SaveStrategy`) entrambe basate sullo stesso *context* `FileOperationContext` il quale viene utilizzato dal controller che funge da client del pattern.