Name: Sherif Mohemd Mostafa.

ID:20.

Name: Ahmed Ali Abdelmeguid.

ID:05.

Paint

The required was to implement a drawing application by allowing the user to draw, color, move, delete, resize etc; a line a rectangle a square a circle an ellipse or a triangle.it was also required to implement these shapes using a certain design. The implementation was done using java and swing GUI. It was also required to implement save and load capabilities using both JsON and Xml formats.

Program Design:

The design is based on the following sets of classes:

1. Shapes classes:

This includes one abstract class (general shape) implementing the shapes interface and six concrete classes extending this abstract class for different implementations for drawing and deep cloning. Also each of the classes has certain aspects corresponding to the represented shape (example length and width for the rectangle).The properties of the shape are always updated according to the setters and the constructors where the shapes share certain properties aspects (as color and fill color) and have certain other special properties (as length and width).each shape has a certain number corresponding to its type to be able to implement a factory class.

The abstract class implements:

* Set and get color.
* Set and get fill color.
* Set and get position.
* Set and get properties.
* Clone.

Each of the six classes implements:

* Draw.
* Override the clone method.

Used Data Structures:

* A map for the properties of the shape.

1. The engine class:

This class implements the drawing engine interface. It contains all the methods required for the GUI of the application. This engine allows drawing and storing several shapes. It allows the ability to undo and redo an action in the application (up to last 20 actions). It also allows the ability to remove one of the shapes or update is with another shape (modify its properties) and redrawing the shapes. This class also allows us to look through all the classes in the package and look for those implementing the interface of the shapes. This allows us to interact with shapes out of our scope by simply placing their class in the package. This class also implements the save and load operations according to the required extension (either Xml or JsOn) storing all the properties of each shape and its class name for instantiation.

The engine implements:

* Add and remove a shape.
* Undo and redo.
* Update a shape.
* Get supported shapes.
* Save and load.

Used Data Structures:

* An array list for the current shapes.
* An array list for the last 20 shapes for undo.
* An array list for the operation corresponding to each shape in the undo list (Add or Remove or Update) to reverse it.
* An array list for the last 20 shapes on which undo was made (for redo).
* An array list for the operations corresponding to each shape in the redo list to redo it.
* An array list containing old shape and other containing new shape updated after each shape update operation helps in undo and redo where update is reversed(if old became new then reversing is new becomes old again).
* A list of classes returned in the get supported shapes method containing all the concrete classes implementing the shapes interface.

1. The GUI (paint) Class:

This is the main class which runs when the program starts.

It contains the implementation controlling the application features as:

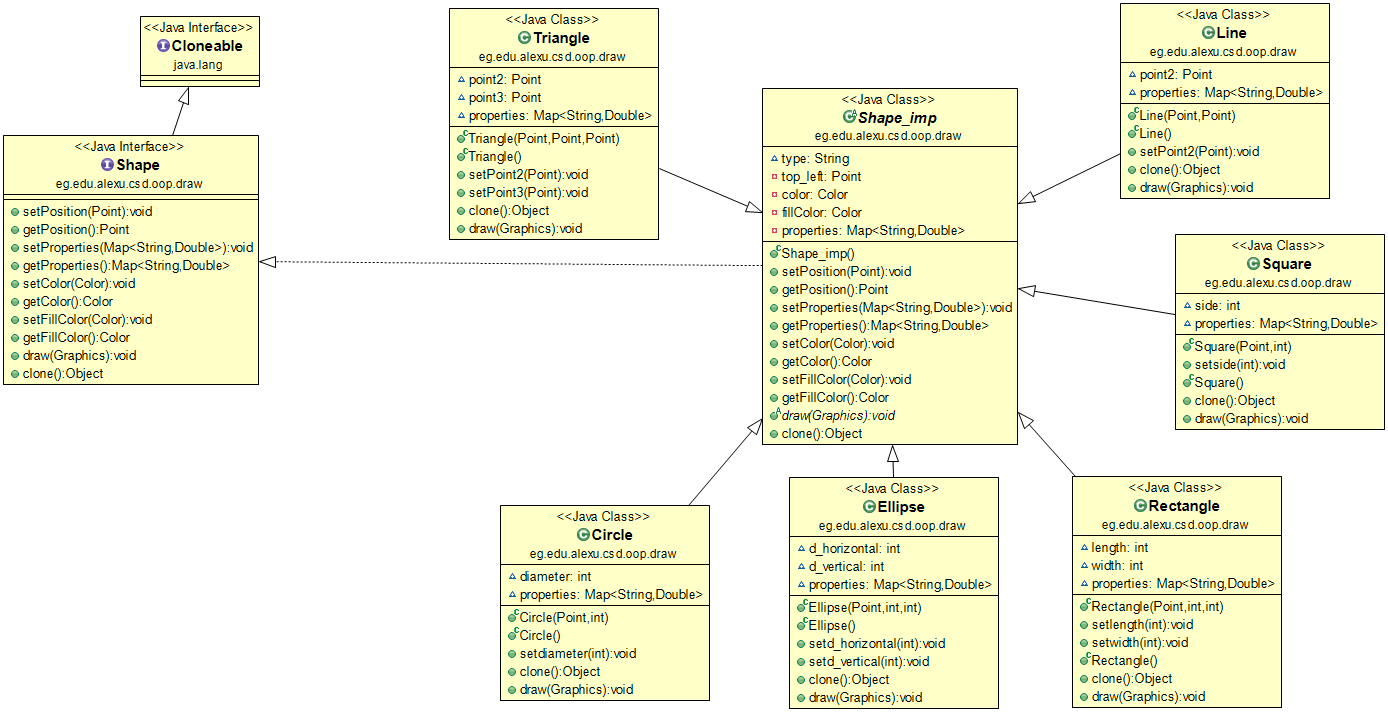
* Drawing shapes
* Modifying them
* Resizing and moving operations
* Removing and copying a shape
* Saving and loading the painting
* Using supported shapes

This class has all the controllers and listeners which allows the user to draw using a friendly interface.

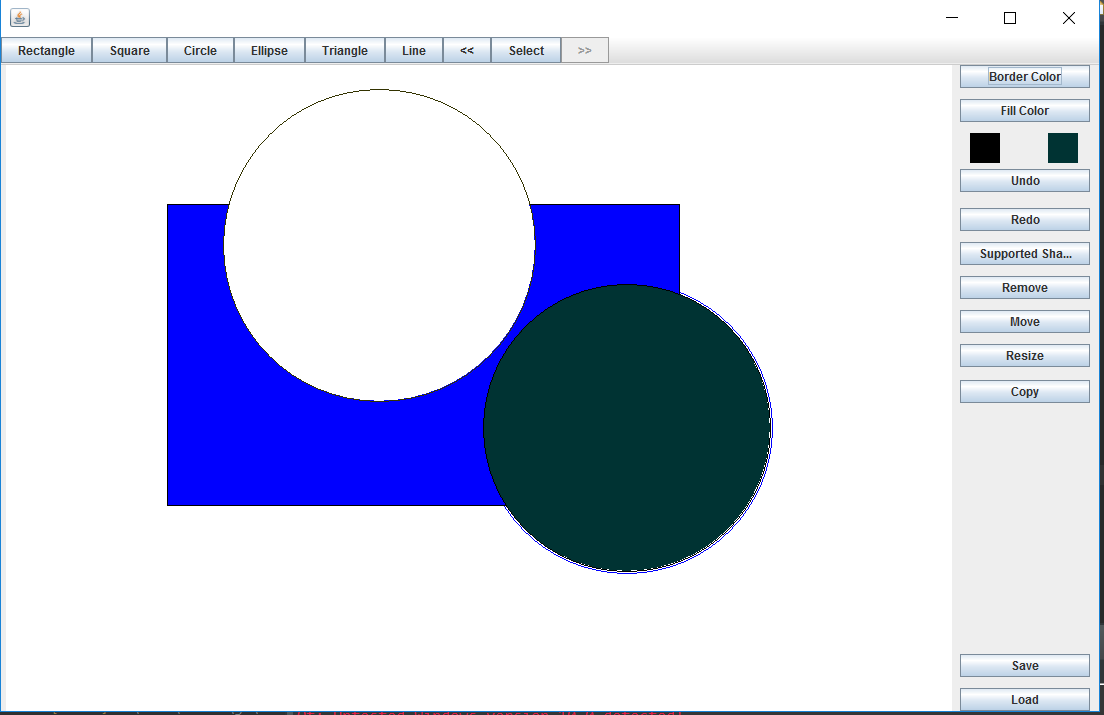
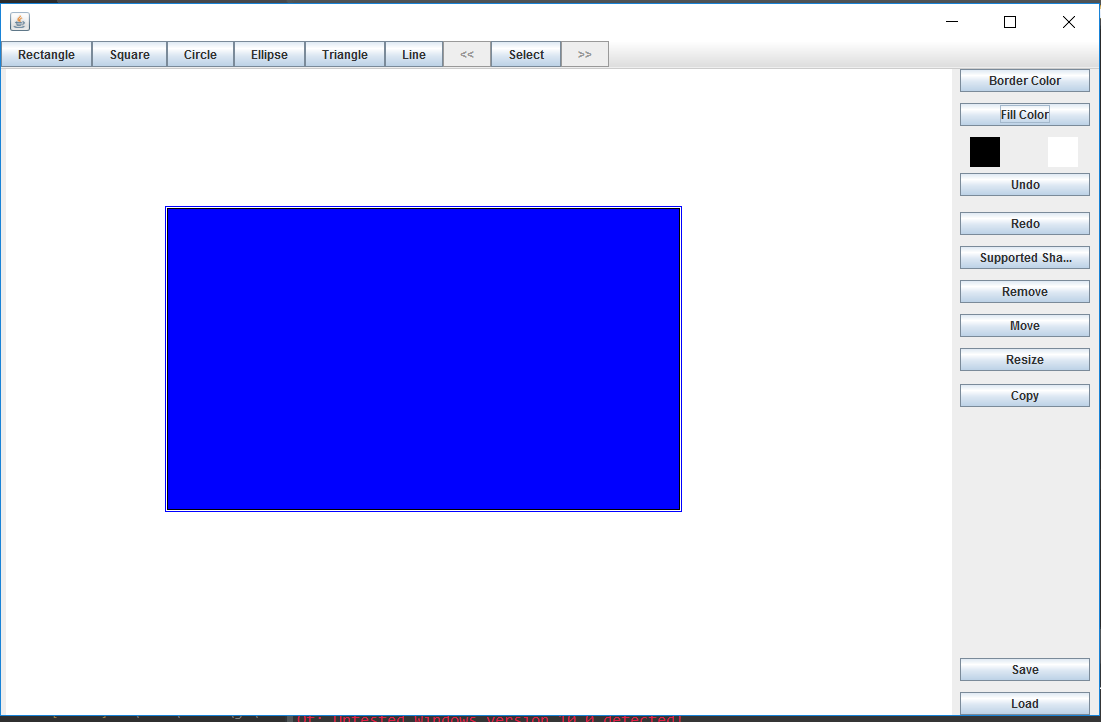
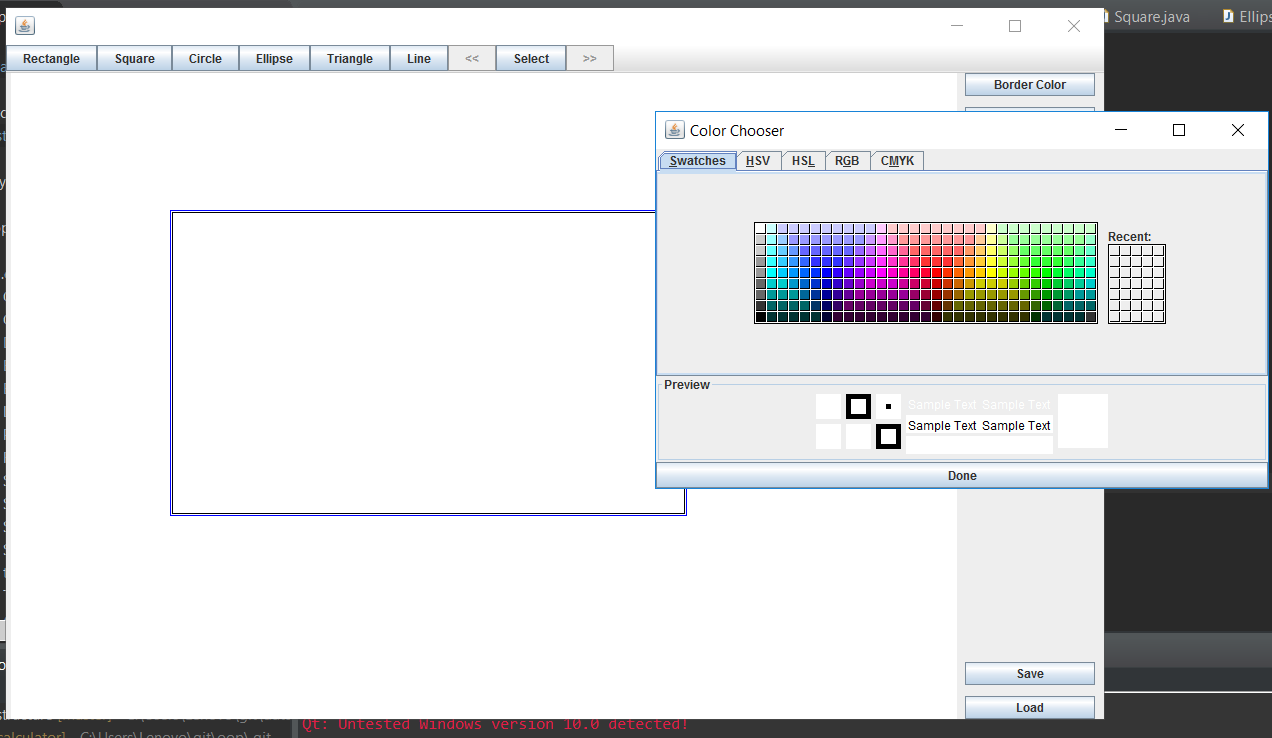
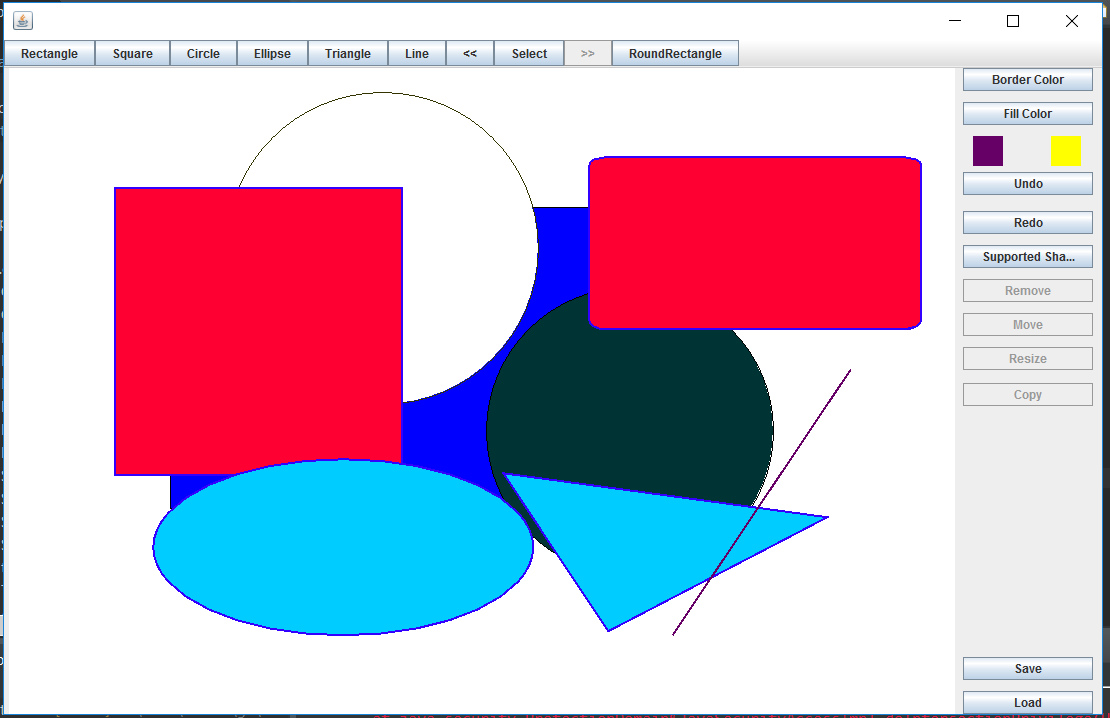
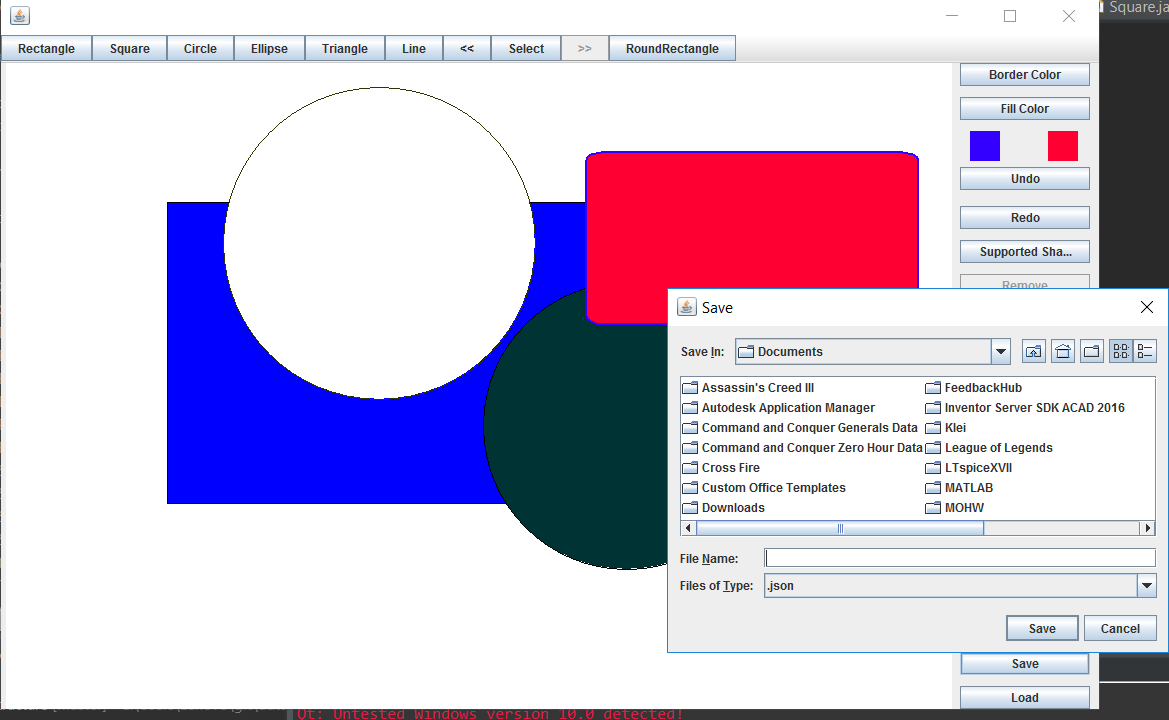
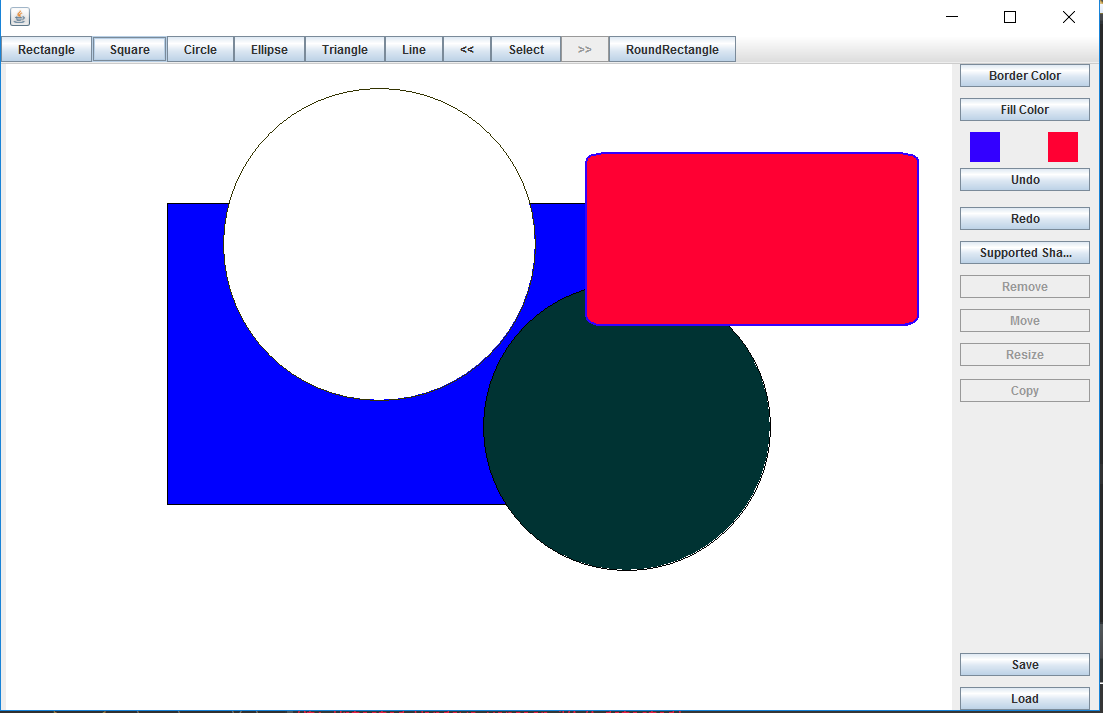
Used Data Structures:

* An array containing the currently drawn shapes: helps in selection and manipulation.
* A list containing all the supported shapes classes.

Uml class diagram for the shapes classes:



GUI Snapshots:

After pressing get supported shapes round rectangle became available:

User Guide:

* Drawing , moving and resizing are done after pressing the button corresponding to the required action by dragging the mouse.
* The triangle is drawn by dragging first to draw its base and another drag to draw its vertex.
* Coloring is via a variety of colors found in the color picker.
* Selection is navigated by pressing a button that selects one of the shapes and navigation through the shapes is via two arrows.
* The selected shape appears shaded by a blue surrounding border.
* After selecting a shape all the manipulation actions are available as copy, remove, move and resize also changing one of the colors after selection changes the color of the selected shape.
* To save or load you must choose the extension , the destination and give the file a name to create it or simply choose an existing file.
* HAVE FUN!

Design Decisions:

* The draw method is the only abstract method in the abstract class implementing the interface.
* The given interface was made to extend the cloneable interface for the clone method.
* The clone method was overridden in each shape class because each has its own map of properties.
* The set and get properties methods only work on the map as a whole but any other setters fill this map (as set color).
* Color and fill color have initial values of black and white respectively.
* The constructor of each shape fills its map too.
* An empty constructor returns a shape with zero dimensions.
* Redo and undo were made using array lists not stacks to ease the manipulation of removing the first index if more than 20 consecutive operations were made.
* The save and load operations were made by our own design of a string having the Xml or JsOn pattern known which is written when saving.
* When loading a method was done especially for the design mentioned manipulating it to obtain the data.
* Drawing is done on a canvas using the mouse where the user can only draw after selecting the shape he wants (pressing the corresponding button).
* When choosing to save the user chooses the extension (.JsOn or .Xml and then the file path and gives it a name).
* When choosing to load the user must choose the correct file to load from.
* The get supported shapes is based on that all the supported classes are in this package any external added class(as round rectangle) must be placed in this package.