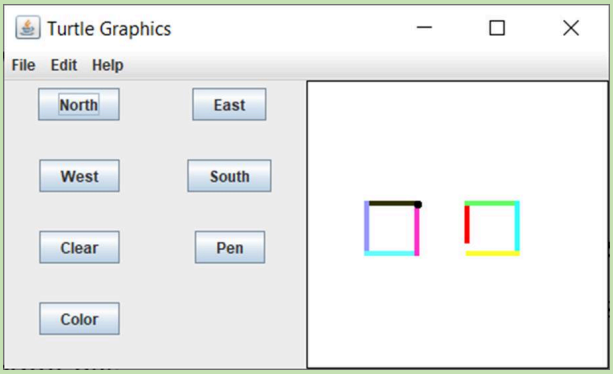
Turtle Graphics

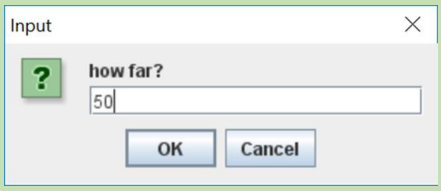
Specification

Similar to the Etch-a-Sketch toy of the 1950s, Turtle Graphics is a desktop application that allows users to command a virtual turtle to move around a virtual canvas. If the turtle's pen is down, then it leaves a colored path.

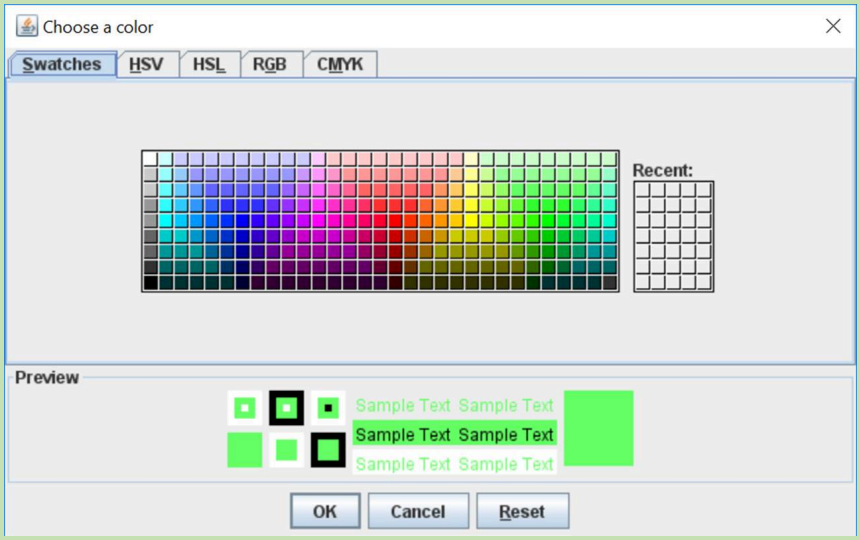


- A "turtle" has
  - a path (a list of points where the turtle stopped),
  - ~~a location (same as the last point on the path),~~
  - ~~a color (of the next line to be drawn)~~
  - ~~a pen status (true if the pen is down, else false).~~
- A point has
  - fixed integer x and y screen coordinates
  - a modifiable color
  - and a modifiable pen status (if true, then a line should be drawn to the next point or the path).
- The Pen button toggles the pen up (don't draw) and down (draw) property of the last point on the turtle's path. This also changes the graphical representation of the turtle from a green circle (pen up) to a green disc (pen down).

- The Clear button erases the drawing by emptying the turtle's path except for its current position.
- The North, East, West, and South buttons prompts the user for the number of steps the turtle should take, then moves the turtle the specified number of steps in the specified direction.



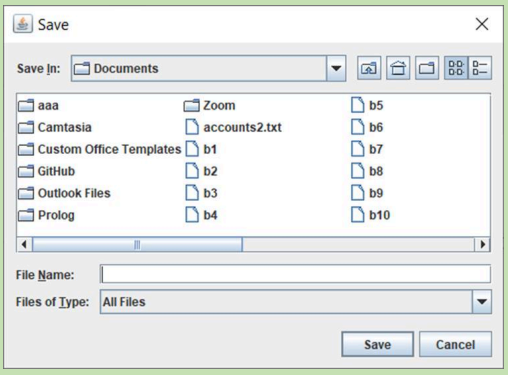
- The Color button allows the user to set the color the turtle uses to specify the color of visited points (assuming the pen is down):



- In addition, the user can issue these commands from the Edit menu.

The File Menu

- The File menu contains the options New, Save, SaveAs, Open, and Quit.
- Save, SaveAs, and Open prompt the user for a file name using the File Chooser Dialog:



- Save remembers the file name entered by the user so that it doesn't need to be entered each time.
- Open and Quit warn the user of unsaved changes in the current turtle.

The Help Menu

The Help menu contains the items Help and About.

- About should display an information dialog containing your name and the date.

- The Help item should display an information dialog with one line of explanation for each command on the Edit menu.

Design

- The design of Turtle Graphics follows the Model-View-Controller design pattern, with the turtle playing the role of model and the canvas playing the role of view.
- Look for other opportunities to use design patterns.
- You may consider introducing controllers for each menu, or a single controller that listens to all of the menus.

Hints

- The name on a button or the name of a menu item is the name of the command it includes with the action event it generates. This means that you don't need separate handlers for the buttons and menu items:

```
void actionPerformed(ActionEvent e) {
   String cmmd = e.getActionCommand();
   if (cmmd == "North") {
     // Edit.North and North button handled here
   }
   // etc.
}
```

- Turtles will need to be serializable as well as any points they contain.
- Ask ChatGPT about prompting users with dialog boxes.
- Here's how to work the color chooser:

```
Color newColor = JColorChooser.showDialog(null, "Choose a color",
turtle.getColor());
```

- Java's graphics coordinate system is weird. The positive x-axis runs across the top of the view panel and the positive y-axis runs down the left side of the view panel. This means that as the turtle moves south, its y-coordinate increases.
- The canvas is a square subregion of the view panel. I included the dimensions of the square as a static field inside the Turtle class:

```
public static Integer WORLD_SIZE = 250;
```

- Consider using the Graphics method drawLine.
- If the user asks the turtle to move off the canvas, the turtle throws an exception or stops at the edge of the canvas.
- Create a reusable tools package containing any classes that might be useful in future applications. For example, you could include your own Publisher-Subscriber implementation.
- Start by drawing a class diagram showing your design of Turtle Graphics.
- You will be graded on the design of your code. Ask yourself how much of your code could be reused in another application. In other words, is the turtle totally independent of the UI? How dependent is the UI on the turtle? Does it know too much about the turtle's internal structure and operation?

Project

Implement Turtle Graphics. Submit your source code in a zip file.