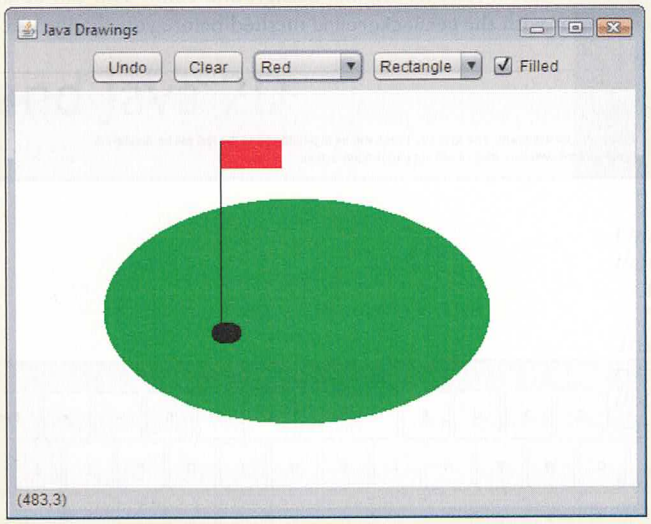

Fig. 14.49 | Interface for drawing shapes.

14.18 *(GUI-Based Version of the ATM Case Study)* Reimplement the ATM Case Study of Chapters 12–13 as a GUI-based application. Use GUI components to approximate the ATM user interface shown in Fig. 12.1. For the cash dispenser and the deposit slot use `JButtons` labeled **Remove Cash** and **Insert Envelope**. This will enable the application to receive events indicating when the user takes the cash and inserts a deposit envelope, respectively.

Making a Difference

14.19 *(Ecofont)* Ecofont (`www.ecofont.eu/ecofont_en.html`)—developed by SPRANQ (a Netherlands-based company)—is a free, open-source computer font designed to reduce by as much as 20% the amount of ink used for printing, thus reducing also the number of ink cartridges used and the environmental impact of the manufacturing and shipping processes (using less energy, less fuel for shipping, and so on). The font, based on sans-serif Verdana, has small circular "holes" in the letters that are not visible in smaller sizes—such as the 9- or 10-point type frequently used. Download Ecofont, then install the font file `Spranq_eco_sans_regular.ttf` using the instructions from the Ecofont website. Next, develop a GUI-based program that allows you to type in a text string to be displayed in the Ecofont. Create **Increase Font Size** and **Decrease Font Size** buttons that allow you to scale up or down by one point at a time. Start with a default font size of 9 points. As you scale up, you'll be able to see the holes in the letters more clearly. As you scale down, the holes will be less apparent. What is the smallest font size at which you begin to notice the holes?

14.20 *(Typing Tutor: Tuning a Crucial Skill in the Computer Age)* Typing quickly and correctly is an essential skill for working effectively with computers and the Internet. In this exercise, you'll build a GUI application that can help users learn to "touch type" (i.e., type correctly without looking at the keyboard). The application should display a *virtual keyboard* (Fig. 14.50) and should allow the user to watch what he or she is typing on the screen without looking at the *actual keyboard*. Use `JButtons` to represent the keys. As the user presses each key, the application highlights the corresponding `JButton` on the GUI and adds the character to a `JTextArea` that shows what the user has typed so far. [*Hint:* To highlight a `JButton`, use its `setBackground` method to change its background

color. When the key is released, reset its original background color. You can obtain the `JButton`'s original background color with the `getBackground` method before you change its color.]

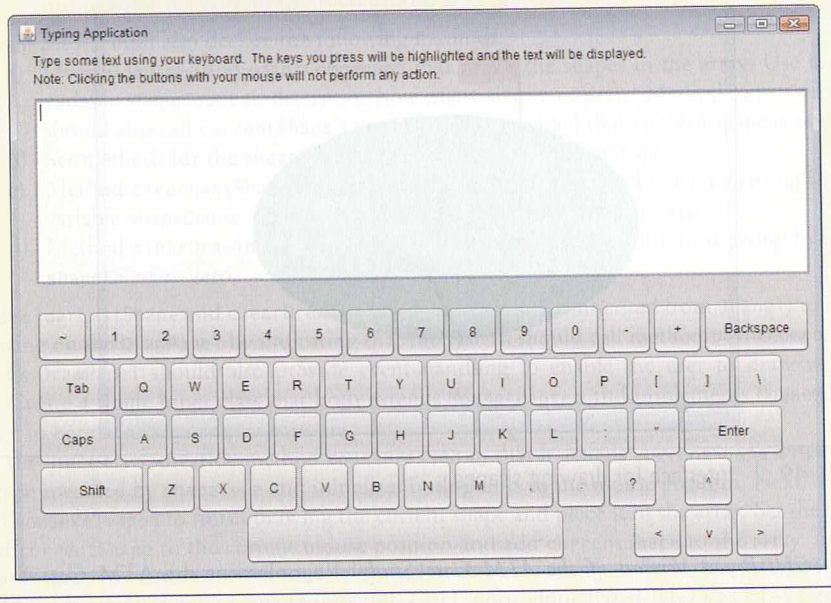

Fig. 14.50 | Typing tutor.

You can test your program by typing a pangram—a phrase that contains every letter of the alphabet at least once—such as "The quick brown fox jumped over a lazy dog." You can find other pangrams on the web.

To make the program more interesting you could monitor the user's accuracy. You could have the user type specific phrases that you've prestored in your program and that you display on the screen above the virtual keyboard. You could keep track of how many keystrokes the user types correctly and how many are typed incorrectly. You could also keep track of which keys the user is having difficulty with and display a report showing those keys.