Building Java Programs

Graphics

Reading: Supplement 3G
Objects (briefly)

- **object**: An entity that contains data and behavior.
  - **data**: variables inside the object
  - **behavior**: methods inside the object
    - You interact with the methods; the data is hidden in the object.
    - A **class** is a type of objects.

- Constructing (creating) an object:
  ```
  Type objectName = new Type(parameters);
  ```

- Calling an object's method:
  ```
  objectName.methodName(parameters);
  ```
Graphical objects

We will draw graphics in Java using 3 kinds of objects:

- **DrawingPanel**: A window on the screen.
  - Not part of Java; provided by the authors. See class web site.

- **Graphics**: A "pen" to draw shapes and lines on a window.

- **Color**: Colors in which to draw shapes.
"Canvas" objects that represent windows/drawing surfaces

To create a window:
DrawingPanel name = new DrawingPanel(width, height);

Example:
DrawingPanel panel = new DrawingPanel(300, 200);
"Pen" or "paint brush" objects to draw lines and shapes

- Access it by calling `getGraphics` on your `DrawingPanel`.
  ```java
  Graphics g = panel.getGraphics();
  ```

- Draw shapes by calling methods on the `Graphics` object.
  ```java
  g.fillRect(10, 30, 60, 35);
  g.fillOval(80, 40, 50, 70);
  ```
Java class libraries, import

- **Java class libraries**: Classes included with the JDK (Java Development Kit).
  - organized into groups named **packages**
  - To use a package, put an *import declaration* in your program:

    ```java
    // put this at the very top of your program
    import packageName.*;
    ```

- **Graphics** belongs to a package named **java.awt**

  ```java
  import java.awt.*;
  ```

- To use **Graphics**, you must place the above line at the very top of your program, before the *public class* header.
Coordinate system

- Each \((x, y)\) position is a *pixel* ("picture element").

- Position \((0, 0)\) is at the window's top-left corner.
  - \(x\) increases rightward and the \(y\) increases *downward*.

- The rectangle from \((0, 0)\) to \((200, 100)\) looks like this:
# Graphics methods

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g.drawLine($x_1, y_1, x_2, y_2$);</td>
<td>line between points $(x_1, y_1), (x_2, y_2)$</td>
</tr>
<tr>
<td>g.drawOval($x, y, width, height$);</td>
<td>outline largest oval that fits in a box of size $width \times height$ with top-left at $(x, y)$</td>
</tr>
<tr>
<td>g.drawRect($x, y, width, height$);</td>
<td>outline of rectangle of size $width \times height$ with top-left at $(x, y)$</td>
</tr>
<tr>
<td>g.drawString($text, x, y$);</td>
<td>text with bottom-left at $(x, y)$</td>
</tr>
<tr>
<td>g.fillOval($x, y, width, height$);</td>
<td>fill largest oval that fits in a box of size $width \times height$ with top-left at $(x,y)$</td>
</tr>
<tr>
<td>g.fillRect($x, y, width, height$);</td>
<td>fill rectangle of size $width \times height$ with top-left at $(x, y)$</td>
</tr>
<tr>
<td>g.setColor($Color$);</td>
<td>set Graphics to paint any following shapes in the given color</td>
</tr>
</tbody>
</table>
Color

- Specified as predefined `Color` class constants:

  ```java
  Color.CONSTANT_NAME
  ```

  where `CONSTANT_NAME` is one of:

  - BLACK,
  - BLUE,
  - CYAN,
  - DARK_GRAY,
  - GRAY,
  - GREEN,
  - LIGHT_GRAY,
  - MAGENTA,
  - ORANGE,
  - PINK,
  - RED,
  - WHITE,
  - YELLOW

- Example:

  ```java
  Color.MAGENTA
  ```
Making your own colors

- Create colors using **Red-Green-Blue (RGB)** values of 0-255
  
  ```java
  Color name = new Color(red, green, blue);
  ```

- Example:
  ```java
  Color brown = new Color(192, 128, 64);
  ```

Using colors

- **Pass a** Color *to* Graphics object's **setColor** method
  - Subsequent shapes will be drawn in the new color.

  ```java
  g.setColor(Color.BLACK);
g.fillRect(10, 30, 100, 50);
g.drawLine(20, 0, 10, 30);
g.setColor(Color.RED);
g.fillOval(60, 40, 40, 70);
  ```

- **Pass a** color *to* DrawingPanel's **setBackgroundColor** method
  - The overall window background color will change.

  ```java
  Color brown = new Color(192, 128, 64);
  panel.setBackground(brown);
  ```
Outlined shapes

- To draw a colored shape with an outline, first fill it, then draw the same shape in the outline color.

```java
import java.awt.*;  // so I can use Graphics

public class OutlineExample {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(150, 70);
        Graphics g = panel.getGraphics();

        // inner red fill
        g.setColor(Color.RED);
        g.fillRect(20, 10, 100, 50);

        // black outline
        g.setColor(Color.BLACK);
        g.drawRect(20, 10, 100, 50);
    }
}
```
Superimposing shapes

- When ≥ 2 shapes occupy the same pixels, the last drawn "wins."

```java
import java.awt.*;

public class Car {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(200, 100);
        panel.setBackground(Color.LIGHT_GRAY);
        Graphics g = panel.getGraphics();

        // car body
        g.setColor(Color.BLACK);
        g.fillRect(10, 30, 100, 50);

        // wheels
        g.setColor(Color.RED);
        g.fillOval(20, 70, 20, 20);
        g.fillOval(80, 70, 20, 20);

        // window
        g.setColor(Color.CYAN);
        g.fillRect(80, 40, 30, 20);
    }
}
```
The \( x, y, w, h \) expressions can use the loop counter variable:

```java
panel.setBackground(Color.YELLOW);
g.setColor(Color.RED);
for (int i = 1; i <= 10; i++) {
    // \( x \quad y \quad w \quad h \)
    g.fillOval(100 + 20 * i, 5 + 20 * i, 50, 50);
}
```

Nested loops can be used with graphics:

```java
g.setColor(Color.BLUE);
for (int x = 1; x <= 4; x++) {
    for (int y = 1; y <= 9; y++) {
        g.drawString("Java", x * 40, y * 25);
    }
}
```
Zero-based loops

- Beginning at 0 and using `<` can make coordinates easier.

```java
DrawingPanel panel = new DrawingPanel(150, 140);
Graphics g = panel.getGraphics();

// horizontal line of 5 20x20 rectangles starting // at (11, 18); x increases by 20 each time
for (int i = 0; i < 5; i++) {
    g.drawRect(11 + 20 * i, 18, 20, 20);
}
```

- Exercise: Write a variation of the above program that draws the output at right.
  - The bottom-left rectangle is at (11, 98).

```java
for (int i = 0; i < 5; i++) {
    g.drawRect(11 + 20 * i, 98 - 20 * i, 20, 20);
}
```
Animation exercise

- Modify the following program to draw a "moving" car.

```java
import java.awt.*;

public class Car {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(200, 100);
        panel.setBackground(Color.LIGHT_GRAY);
        Graphics g = panel.getGraphics();

        // car body
        g.setColor(Color.BLACK);
        g.fillRect(10, 30, 100, 50);

        // wheels
        g.setColor(Color.RED);
        g.fillOval(20, 70, 20, 20);
        g.fillOval(80, 70, 20, 20);

        // window
        g.setColor(Color.CYAN);
        g.fillRect(80, 40, 30, 20);
    }
}
```
Parameterized figures

- Modify the car-drawing method so that it can draw cars at different positions, as in the following image.
  - Top-left corners: (10, 30), (150, 10)
  - Increase the drawing panel's size to 260x100 to fit.
Drawing with parameters

- To draw in a method, you must pass the Graphics object to the method.
  - Otherwise, \( g \) is out of scope and cannot be used!

- syntax (declaration):
  ```java
  public static void name(Graphics g, parameters) {
    statement(s);
  }
  ```

- syntax (call):
  ```java
  name(g, values);
  ```
Parameterized answer

import java.awt.*;

public class Car3 {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(260, 100);
        panel.setBackground(Color.LIGHT_GRAY);
        Graphics g = panel.getGraphics();
        drawCar(g, 10, 30);
        drawCar(g, 150, 10);
    }

    public static void drawCar(Graphics g, int x, int y) {
        g.setColor(Color.BLACK);
        g.fillRect(x, y, 100, 50);
        g.setColor(Color.RED);
        g.fillOval(x + 10, y + 40, 20, 20);
        g.fillOval(x + 70, y + 40, 20, 20);
        g.setColor(Color.CYAN);
        g.fillRect(x + 70, y + 10, 30, 20);
    }
}
Java book figure

- Write a program that draws the following figure:
  - drawing panel is size 200x150
  - book is at (20, 35), size 100x100
  - cyan background
  - white "BJP" text at position (70, 55)
  - stairs are (red=191, green=118, blue=73)
  - each stair is 9px tall
    - 1st stair is 10px wide
    - 2nd stair is 20px wide ...
  - stairs are 10px apart (1 blank pixel between)
// Draws a Building Java Programs textbook with DrawingPanel.
import java.awt.*;

public class Book {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(200, 150);
        panel.setBackground(Color.WHITE);
        Graphics g = panel.getGraphics();

        g.setColor(Color.CYAN);
        // cyan background
        g.fillRect(20, 35, 100, 100);

        g.setColor(Color.WHITE);
        // white "bjp" text
        g.drawString("BJP", 70, 55);

        g.setColor(new Color(191, 118, 73));
        for (int i = 0; i < 10; i++) {
            // orange "bricks"
            g.fillRect(20, 35 + 10 * i, 10 + 10 * i, 9);
        }
    }
}
Multiple Java books

- Modify the Java book program so that it can draw books at different *positions* as shown below.
  - book top/left positions: (20, 35), (150, 70), (300, 10)
  - drawing panel's new size: 450x180
Multiple books solution

// Draws many BJP textbooks using parameters.
import java.awt.*;

public class Book2 {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(450, 180);
        panel.setBackground(Color.WHITE);
        Graphics g = panel.getGraphics();

        // draw three books at different locations
        drawBook(g, 20, 35);
        drawBook(g, 150, 70);
        drawBook(g, 300, 10);
    }
}

...
Multiple books, cont'd.

...
Resizable Java books

- Modify the Java book program so that it can draw books at different sizes as shown below.
  - book sizes: 100x100, 60x60, 200x200
  - drawing panel's new size: 520x240
// Draws many sized BJP textbooks using parameters.
import java.awt.*;

public class Book3 {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(520, 240);
        panel.setBackground(Color.WHITE);
        Graphics g = panel.getGraphics();

        // draw three books at different locations/sizes
        drawBook(g, 20, 35, 100);
        drawBook(g, 150, 70, 60);
        drawBook(g, 300, 10, 200);
    }
    ...
}
// Draws a book of the given size at the given position.
public static void drawBook(Graphics g, int x, int y, int size) {
    g.setColor(Color.CYAN); // cyan background
    g.fillRect(x, y, size, size);

    g.setColor(Color.WHITE); // white "bjp" text
    g.drawString("BJP", x + size/2, y + size/5);

    g.setColor(new Color(191, 118, 73)); // orange "bricks"
    for (int i = 0; i < 10; i++) {
        g.fillRect(x, // x
                    y + size/10 * i, // y
                    size/10 * (i + 1), // width
                    size/10 - 1); // height
    }
}
Polygon

Objects that represent arbitrary shapes

• Add points to a Polygon using its `addPoint(<x>, <y>)` method.

• Example:

```java
drawingpanel p = new drawingpanel(100, 100);
graphics g = p.getgraphics();
g.setColor(color.green);

polygon poly = new polygon();
poly.addpoint(10, 90);
poly.addpoint(50, 10);
poly.addpoint(90, 90);
g.fillpolygon(poly);
```
DrawingPanel methods

- `panel.clear();`
  Erases any shapes that are drawn on the drawing panel.

- `panel.setWidth(width);`
  `panel.setHeight(height);`
  `panel.setSize(width, height);`
  Changes the drawing panel's size to the given value(s).

- `panel.save(filename);`
  Saves the image on the panel to the given file (String).

- `panel.sleep(milliseconds);`
  Pauses the drawing for the given number of milliseconds.
Animation with sleep

- DrawingPanel's sleep method pauses your program for a given number of milliseconds.

- You can use sleep to create simple animations.

```java
DrawingPanel panel = new DrawingPanel(250, 200);
Graphics g = panel.getGraphics();

g.setColor(Color.BLUE);
for (int i = 1; i <= 10; i++) {
    g.fillOval(15 * i, 15 * i, 30, 30);
    panel.sleep(500);
}
```

- Try adding sleep commands to loops in past exercises in this chapter and watch the panel draw itself piece by piece.
Animation exercise

- Modify the previous program to draw a "moving" animated car.