```
// This program is copyright VUW.
// You are granted permission to use it to construct your answer to a COMP102 assignment.
// You may not distribute it in any other way without permission.

/* Code for COMP102 - 2024T3, Assignment 4
 * Name:
 * Username:
 * ID:
 */

import ecs100.*;
import java.util.*;
import java.awt.Color;

/**
 *  Lets a player play a simple Solitaire dominoes game.
 *  Dominoes are rectangular tiles with two numbers from 0 to 6 on
 *  them (shown with dots).
 *  The player has a "hand" which can contain up to five dominoes.
 *  They can
 *  - reorder the dominoes in their hand,
 *  - place dominoes from their hand onto the table,
 *  - pick up more dominoes from a bag to fill the gaps in their "hand".
 *  - replace a domino from their hand back into the bag.
 *  The core and completion do not involve any of the matching and scoring
 *  of real dominoes games.
 *
 *  PROGRAM DESIGN
 *  The dominoes are represented by objects of the Domino class.
 *  The Domino constructor will construct a new, random domino.
 *  Dominos have a draw(double x, double y, boolean horiz) method that will draw the
 *    Domino centered at the specified position, either vertically or horizontally.
 *
 *  The program has two key fields:
 *    hand:  an array that can hold 5 Dominos.
 *    table: an ArrayList of the Dominos that have been placed on the table.
 *
 *  The hand should be displayed near the top of the Graphics pane with a
 *   rectangular border and each domino drawn at its place in the hand.
 *  Empty spaces in the hand should be represented by nulls and displayed as empty.
 *
 *  The user can select a position on the hand using the mouse.
 *  The selected domino (or empty space) should be highlighted with
 *  a border around it.
 *
 *  The user can use the "Left" or "Right" button to move the selected domino
 *  (or the space) to the left or the right, in which case the domino is
 *  swapped with the contents of the adjacent position in the hand.
 *  If the selected position contains a domino, the user
 *  can use the "Place" button to move the selected domino to the table.
 *
 *  If there are any empty positions on the hand, the user can use the
 *  "Pickup" button to get a new (random) domino which will be added to
 *  the hand at the leftmost empty position.
 *
 *  The table is represented by an ArrayList of dominos.
 *  At the beginning of the game the table should be empty.
 *  Dominos should be added to the end of the table.
 *  The table should be displayed in rows at the top of the graphics pane.
 */

public class DominoGame{
    public static final int NUM_HAND = 5;    // Number of dominos in hand

    // Fields: hand, table and selectedPos
    private Domino[] hand;;            // the hand (fixed size array of Dominos)
    private ArrayList<Domino> table;   // the table (variable sized list of Dominos)

    private int selectedPos = 0;      //  selected position in the hand.
```

```
    // (You shouldn't add any more fields for core or completion)

    /**
     * Restart the game:
     *  set the table to be empty,
     *  set the hand to have no dominos
     */
    public void restart(){
        /*# YOUR CODE HERE */
        hand = new Domino[NUM_HAND]  ;
        table = new ArrayList<Domino>();



        this.redraw();

    }

    /**
     * If there is at least one empty position on the hand, then
     * create a new random domino and put it into the first empty
     * position on the hand.
     * (needs to search along the array for an empty position.)
     */
    public void pickup(){
        /*# YOUR CODE HERE */
        for(int i=0;i < hand.length;i++){
            if(hand[i] == null){
                hand[i] = new Domino();
                break;

                }
        }

        this.redraw();
    }

    /**
     * Remove domino from selected position on hand (if there is domino there)
     * (to go back into the "bag")
     */
    public void removeDomino(){
        /*# YOUR CODE HERE */
        if(hand[selectedPos] !=null){
            hand[selectedPos] = null;
        }

        this.redraw();
    }


    /**
     * Move domino from selected position on hand (if there is domino there) to the table
     * The selectedPos field contains the index of the selected domino.
     */
    public void placeDomino(){
        /*# YOUR CODE HERE */
        if(hand[selectedPos] !=null){
            table.add(this.hand[selectedPos]);
            this.hand[selectedPos] = null;


        }

        this.redraw();
    }

    /**
```

```
     * If there is a domino at the selected position in the hand,
     * flip it over.
     */
    public void flipDomino(){
        /*# YOUR CODE HERE */
        if(this.hand[selectedPos] != null){
            this.hand[selectedPos].flipNums();
        }
        this.redraw();
    }

    /**
     * Move the contents of the selected position in the hand
     * to the leftmost position, moving all the items on its left
     * (if there are any) one step to the right
     */
    public void moveToLeftEnd(){
        /*# YOUR CODE HERE */
         if (this.hand[selectedPos] != null) {

        Domino selectedDomino = this.hand[selectedPos];


        for (int i = selectedPos; i > 0; i--) {
            this.hand[i] = this.hand[i - 1];
        }


        this.hand[0] = selectedDomino;
    }
        this.redraw();
    }

    /**
     * Move the contents of the selected position in the hand
     * to the rightmost position, moving all the items on its right
     * (if there are any) one step to the left
     */
    public void moveToRightEnd(){
        /*# YOUR CODE HERE */
        if (this.hand[selectedPos] != null) {

        Domino selectedDomino = this.hand[selectedPos];


        for (int i = selectedPos; i < this.hand.length - 1; i++) {
            this.hand[i] = this.hand[i + 1];
        }


        this.hand[this.hand.length - 1] = selectedDomino;
    }
        this.redraw();
    }

    /**
     * If the table is empty, only a double (first and second are the same) can be suggested.
     * If the table is not empty, see if one domino has a number that matches the right
     *    number of the last domino on the table.
     */
    public void suggestDomino(){
        /*# YOUR CODE HERE */

    }

    //=========================================================
    // Methods for GUI and display.  ALL WRITTEN FOR YOU
    //=========================================================
```

```
    /**
     * Set up the GUI with buttons and mouse.
     * Start the first game.
     */
    public void setupGUI(){
        UI.setMouseListener( this::doMouse );
        UI.addButton("Pickup", this::pickup);
        UI.addButton("Put back", this::removeDomino);
        UI.addButton("Place", this::placeDomino);
        UI.addButton("Flip",  this::flipDomino);
        UI.addButton("Left", this::moveToLeftEnd);
        UI.addButton("Right", this::moveToRightEnd);
        UI.addButton("Suggest", this::suggestDomino);
        UI.addButton("Restart", this::restart);
        UI.addButton("Quit", UI::quit);

        UI.setWindowSize(1100,500);
        UI.setDivider(0.0);  // graphics pane only
        this.restart();
    }

    // constants for the layout
    public static final int HAND_LEFT = 80; // x-position of the center of the leftmost Domino in
 the hand
    public static final int HAND_Y = 60;   // y-Position of all the Dominos in the hand

    public static final int TABLE_LEFT = 60;
    public static final int TABLE_Y = 170;

    public static final int SPACING = 4;  // distance between Dominos when laid out.

    /**
     *  Redraw the table and the hand.
     */
    public void redraw(){
        UI.clearGraphics();
        UI.printMessage("");
        this.drawHand();
        this.drawTable();
    }

    /**
     * Draws the outline of the hand,
     * draws all the Dominos in the hand,
     * highlights the selected position in some way
     */
    public void drawHand(){
        for (int t=0; t<this.hand.length; t++){
            if (this.hand[t] != null){
                int x = HAND_LEFT + t*(Domino.WIDTH + SPACING);
                this.hand[t].draw(x, HAND_Y, false);
            }
        }
        // outline the hand and the selected position
        UI.setLineWidth(2);
        UI.setColor(Color.black);
        double border = 4;
        UI.drawRect(HAND_LEFT-Domino.WIDTH/2-border, HAND_Y-Domino.LENGTH/2-border,
            (Domino.WIDTH+SPACING)*NUM_HAND+2*border, Domino.LENGTH+2*border);
        UI.setLineWidth(2);
        UI.setColor(Color.green);
        int selLeft = HAND_LEFT-Domino.WIDTH/2 + (this.selectedPos * (Domino.WIDTH+SPACING)) - 2;
        UI.drawRect(selLeft, HAND_Y-Domino.LENGTH/2 - 2, Domino.WIDTH+SPACING, Domino.LENGTH+4);

    }

    /**
     * Draws the list of Dominos on the table, 7 to a row
     * Note, has to wrap around to a new row when it gets to the
```

```
        * edge of the table
        */
    public void drawTable(){
        int x = TABLE_LEFT;
        int y = TABLE_Y;
        int count = 0;
        for (Domino domino : this.table){
            domino.draw(x, y, true);
            x = x + Domino.LENGTH+SPACING;
            count++;
            if (count == 7){
                x = TABLE_LEFT;
                y = y + Domino.WIDTH+3*SPACING;
                count = 0;
            }
        }

    }

    /**
     * Allows the user to select a position in the hand using the mouse.
     * If the mouse is released over the hand, then sets  selectedPos
     * to be the index into the hand array.
     * Redraws the hand and table */
    public void doMouse(String action, double x, double y){
        if (action.equals("released")){
            if (Math.abs(y - HAND_Y) <= Domino.LENGTH/2 &&
                x >= HAND_LEFT-Domino.WIDTH/2 &&
                x <= HAND_LEFT + (NUM_HAND-0.5)*(Domino.WIDTH+SPACING)) {
                this.selectedPos = (int) ((x-(HAND_LEFT-Domino.WIDTH/2))/(Domino.WIDTH+SPACING));
                this.redraw();
            }
        }
    }

    /**
     * Creates an object, calls setupGUI, and calls restart.
     */
    public static void main(String[] args){
        DominoGame obj = new DominoGame();
        obj.setupGUI();
        obj.restart();
    }
}
```