//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// FenceAndHouse.java Chris Dang Class: CSCI 1125 Spring 2015

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Program will draw a house behind a fence

//

import java.awt.\* ;

import javax.swing.JApplet;

public class FenceAndHouse extends JApplet{

public void paint (Graphics page)

{

//constants used for fence

final int WIDTH = 25 ;

final int HEIGHT = 250 ;

final int POSTS = 10 ;

final int BOARDS = 3 ;

//constants used for house

final int HOUSE\_SIZE = HEIGHT / 2 ;

final int HOUSE\_HEIGHT = HEIGHT / 3 ;

final int DOOR\_X\_POSITION = HEIGHT / 2 + WIDTH + WIDTH + WIDTH + (WIDTH /2) ;

final int DOOR\_Y\_POSITION = HEIGHT /2 + WIDTH + (WIDTH /2) ;

final int DOOR\_HEIGHT = WIDTH \* 2 ;

final int WINDOW\_POSITION = HEIGHT / 2 + WIDTH ;

final int ROOF\_X\_POSITION = HEIGHT / 2 - WIDTH ;

final int ROOF\_Y\_POSITION = WIDTH \* 4 ;

final int ROOF\_WIDTH = HEIGHT / 2 + WIDTH + WIDTH ;

final int KNOB\_X\_POSITION = HEIGHT / 2 + WIDTH + WIDTH + WIDTH + WIDTH ;

final int KNOB\_Y\_POSITION = HEIGHT - (WIDTH \* 3) ;

final int DOOR\_KNOB = 5 ;

//Draws House----------------------

// House Frame

page.setColor(Color.cyan);

page.fillRect(HOUSE\_SIZE, HOUSE\_SIZE, HOUSE\_SIZE, HOUSE\_HEIGHT );

// Door

page.setColor(Color.gray);

page.fillRect(DOOR\_X\_POSITION, DOOR\_Y\_POSITION, WIDTH, DOOR\_HEIGHT);

// Window

page.setColor(Color.yellow);

page.fillRect(WINDOW\_POSITION, WINDOW\_POSITION, DOOR\_HEIGHT, WIDTH);

// Roof

page.setColor(Color.black);

page.fillRect(ROOF\_X\_POSITION, ROOF\_Y\_POSITION , ROOF\_WIDTH, WIDTH);

// Doorknob

page.fillOval(KNOB\_X\_POSITION, KNOB\_Y\_POSITION, DOOR\_KNOB, DOOR\_KNOB);

//Draws Fence-----------------------

setBackground(Color.green) ;

//creates vertical posts

page.setColor(Color.white) ;

for (int i = 1; i <= POSTS; i++)

page.fillRect(((WIDTH + WIDTH) \* i), WIDTH, WIDTH, HEIGHT) ;

// x-coordinate has twice the width so that it creates the first post

// followed by a space so that the next post is created after the space

//creates horizontal boards

for (int j = 0; j < BOARDS; j++)

page.fillRect(WIDTH, (WIDTH + WIDTH + (j \* (HEIGHT - WIDTH))), (WIDTH + (WIDTH \* POSTS \* 2)), WIDTH) ;

// y-coordinate starts with double the width so that the top board is

// down the width of one post.

// The next board follows at the bottom of the fence

// The width of the boards varies based on the number of posts.

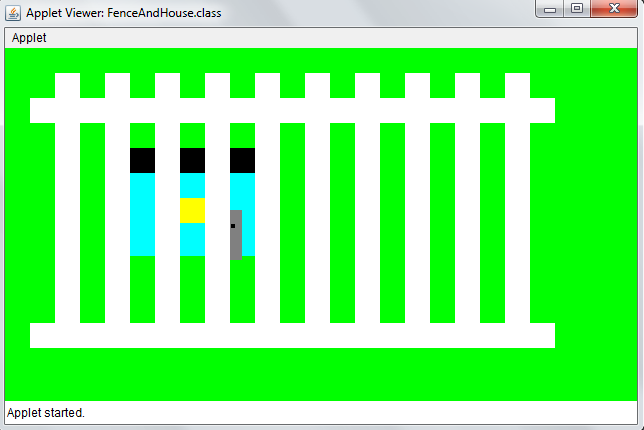
// The width of the boards is twice the number of posts

// (to account for the spaces), plus an additional width of a post to have

// extra space at the end of the boards.

}

}



//-DrawFive.java----------------------------------------------------------------

// Programmer: Chris Dang Class: CSCI 1125 Spring 2015

// Program will draw and display 5 unique cards

//------------------------------------------------------------------------------

public class DrawFive {

public static void main(String[] args) {

final int hand = 5 ;

Deck bikeDeck = new Deck() ;

for (int i = 0; i < hand; i ++) {

int hold = bikeDeck.draw() ; // card number is initially drawn

hold = bikeDeck.checkExists(hold) ;// if the card number has been

bikeDeck.display(bikeDeck.getCard(hold));//drawn, then the number

// is redrawn until a number

} // end for that has not been drawn is generated

} // end static void main

} // end Draw Five driver

import java.util.Random;

public class Deck {

final int deckSize = 52 ;

final int maxSuits = 4 ;

final int maxCards = 13 ;

private class Card {

public int faceVal ; // contains int values 1 - 13

public int suit ; // contains int values 1 - 4

public boolean exist ;// shows if the card has been displayed

public Card(int face, int suitNum){

faceVal = face ;

suit = suitNum ;

exist = false ;

} // end card constructor

} // end Card class

private Card[] Deck ;

// A deck object consists of a Card array

public Deck(){

int cardCount = 0 ;

Deck = new Card[deckSize] ;

for (int i = 0; i < maxSuits; i++){ // sets up 13 cards of each suit

for (int j = 0; j < maxCards; j++){

Deck[cardCount] = new Card(j + 1, i + 1);

++cardCount ;

} // end for

}

} // end Deck constructor

// function: draw

// pre: none

// post: returns an integer of 0 - 51 to represent 1 of 52 cards

public int draw() {

// generates a value 0 - 51 to represent one

// of the 52 cards available

Random generator = new Random() ;

return generator.nextInt(52) ;

} // end draw

// function: getExist

// pre: must pass in Card object

// post: returns a bool from the card to see if card has been displayed

public boolean getExist(Card pick){

return pick.exist ;

}

// function: getSuit

// pre: must pass in Card object

// post: returns a string of containing suit type of card

public String getSuit(Card pick){

switch (pick.suit)

{

case 1: return "Clubs" ;

case 2: return "Hearts" ;

case 3: return "Spades" ;

case 4: return "Diamonds" ;

} // end switch

return "\n\n Suit type does not exist \n\n" ;

} // end getSuit

// function: getFaceVal

// pre: must pass in Card object

// post: returns a string of containing face value of card

public String getFaceVal(Card pick) {

switch (pick.faceVal)

{

case 1: return " Ace" ;

case 2: return " 2" ;

case 3: return " 3" ;

case 4: return " 4" ;

case 5: return " 5" ;

case 6: return " 6" ;

case 7: return " 7" ;

case 8: return " 8" ;

case 9: return " 9" ;

case 10: return " 10";

case 11: return " Jack" ;

case 12: return "Queen" ;

case 13: return " King" ;

} // end switch

return "\n\n Face Value does not exist \n\n" ;

} // end getFaceVal

// function: getCard

// pre: must pass in an integer

// post: returns a a card from the deck object

public Card getCard(int hold) {

return Deck[hold] ;

} // end getCard

// function: checkExists

// pre: must pass in Card object

// post: returns an integer for the card number

public int checkExists(int hold) {

while (this.getExist(this.getCard(hold)))

hold = this.draw() ;

return hold ;

} // end checkExists

// function: display

// pre: must pass in Card object

// post: prints the card with suit and face value

public void display(Card pick) {

System.out.println(this.getFaceVal(pick) + " of " + this.getSuit(pick));

pick.exist = true ;

} // end display

} // end Deck class

