Blue Pelican Computer Science
Three AP Labs

Teacher Version 1.01

Edited by: Tanner Wright

This page left blank on purpose.

Getting Started (a word from the author)

Don't skip this:
If you are like me, you may be tempted to skip over much of this section and go straight to the labs. Take the time to read this, especially the part about the 20 hour requirement, the sequence of presentation, and downloading the zip files. For your **planning and preparation**, **those are important!**

The three labs:
So, GridWorld is finished and it's time for the "Three AP Labs". Is that good or bad? GridWorld made for a nice motivator and learning opportunity for student late in the course, but did little to engage or motivate beginning students early on. On the other hand, parts of the The Three AP Labs can be presented much earlier in the course and students should quickly become aware of how relevant and practical they are. The labs range from the fairly simple (Magpie ChatBot Lab), to the more complex (Elevens Lab, a card game). Right in the middle is Picture Lab, a fascinating lab that lets student manipulate digital images.

A down side:
Besides being so engaging and motivating, and even though these labs contribute greatly to the learning process**, there is a danger** and down side. They could easily become so time consuming, they could crowd out some precious instruction time. If only the material sent from the College Board for these labs is used, a teacher will typically need to spend many hours in filtering through it all in trying to decide how and what to present as well the sequence of presentation.

Philisophy:
Hopefully, what I have here addresses all of these needs and pitfalls. I have approached the organization of these labs with a philosophy of "Simpler is Better." Much like the organization of my Blue Pelican Math curriculum and the lessons in my Java textbook, Blue Pelican Java, you will find here a day-by-day, march-right-through, "get it done" attitude.

Tying it all together, keeping it simple:
The original three labs were written by three different individuals with three different styles and organizations. I have tried to unify these three labs into presentations that are both consistent and concise. Again if you expect to navigate through the original labs as they come from the College Board, expect a

lengthy process in deciding what to leave out and what to keep for your students. It was an arduous process in making these decisions and producing new presentation of the three labs; however, I think you will find them to be very much like the lessons in the Blue Pelican Java textbook; direct, simple, and to-the-point.

The 20 hour requirement:
Now, about that minimum of 20 hours requirement that your student must spend on the labs. You still will have some decisions to make regarding what to leave in and what to take out. I have some definite suggestions and the following should be of assistance:

> You will notice that I have the labs organized into Day-1, Day-2, etc. progression where the assumption is that each "day" is an actual hour of instruction time.
>
> - Magpie Lab has 5 days of instruction.
> - Picture Lab has 5 days of instruction.
> - Elevens Lab has 11 days of instruction; however, 4 of those are optional (which I recommend be skipped) 11 – 4 = 7
>
>   5 + 5 + 7 = 17 hours of instruction
>
> We need three more hours. The fact is, a few lesson won't exactly fit one hour (1 class, day) These could be carried over into homework or be finished by the student in an off-period. However, they would best be extended over two class periods. Following is a list of the lessons that would need more than one class period to finish. The ones with the asterisks are the most in need of extra time.
>
> - Magpie, Day-3
> - Magpie, Day-4*
> - Picture, Day-1*
> - Elevens Day-2*
> - Elevens Day-3
> - Elevens Day-9

Meeting the minimum requirement:
Choose the three with the asterisks to make each of those lesson actually span 2

days (2 hours) . That will add 3 hours to the 17 we obtained above to give a grand total of 20 (the minimum required).

Exceeding the minimum requirements:
My recommendation is to make each of the six lessons in the list above span 2 days. That will give a total of 23 hours with these three labs.

The sequence of presentation:
When to present a lab really depends on the point in the java curriculum to which the students have progressed. The order of complexity (from simplest to the most advanced) of the labs is: **Magpie, Picture, Elevens. Therefore, the labs should be presented in that order.** The following shows what java concepts are needed to understand the labs. Once the students have reached that point, the lab could be presented. **Just don't wait too long**. Part of what these labs are to accomplish is to rescue those students that are struggling with the tedium of the syntax of the language and not seeing any practical application or relevance to the world around them.

Magpie Lab: *if*, *else if* statements, *String* methods, one dimensional arrays. Only the last lesson uses arrays and if your students haven't had arrays yet, this could be used to introduce arrays.

Picture Lab: *if else*, loops, classes, objects, *String* methods, and two dimensional arrays, inheritance. Even if your students have had arrays of one dimension only, they should be able to adapt easily. Don't worry too much about inheritance; just a cursory explanation will do. Don't wait too long to get to this lab. **Student will love it**.

Elevens: All the fundaments (conditional statements and loops), *ArrayList*, *String* methods, classes, objects, random numbers, inheritance. You might want to wait until 2nd semester on this one until most of the advanced topics have been covered.

Download files and prepare student computers:
The beginning of each lab will instruct the student to download, unzip, and store certain folders and files on his computer. It will be a **tremendous time saver** if you as the teacher would download all of these files and already have them stored on

the student computers.

After being told to download the files, the student will be instructed to bring certain classes into a java project for that particular lab. All the students will need to know at that point is where you stored the files on their computer.

Download these zip files, unzip, and then store the resulting folder structure on student computers:

www.bluepelicanjava.com/threeLabs/magpieLab.zip.

www.bluepelicanjava.com/threeLabs/pictureLab.zip.

www.bluepelicanjava.com/threeLabs/elevensLab.zip.

In the lessons, references are made to folder names as they exist now, so after unzipping these files, do not change the names of the folders.

Charles Cook / Author Blue Pelican Java

Three AP Labs – Table of Contents

Blue Pelican Computer Science
AP Magpie Lab

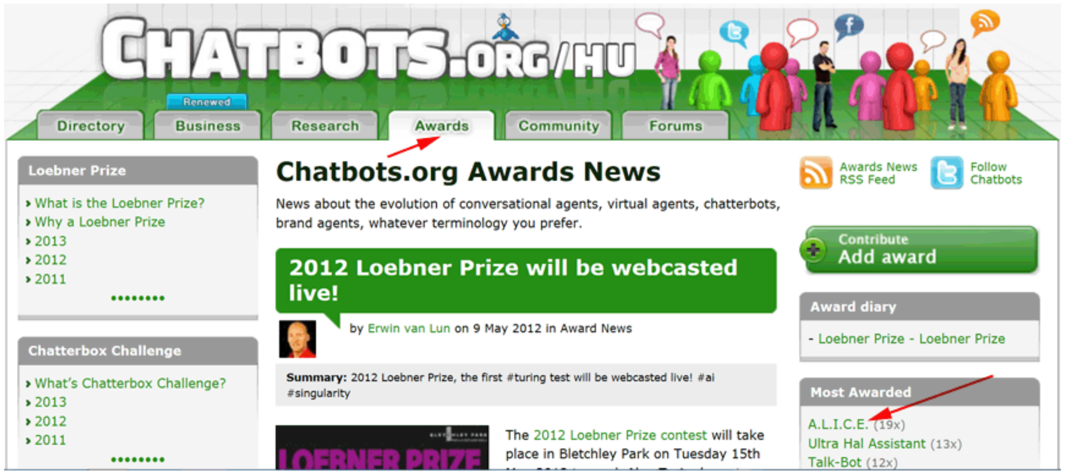
Teacher Version 1.01

Edited by: Tanner Wright

Blue Pelican Magpie Lab – Day-1

Talking to a computer:
In this lab we will learn about Natural Language Processing (NLP). This is similar to a chat room where two people converse with each other in written form using computers over a network. In our studies here, instead of two people, there is only one with a "smart computer" on the other end participating in the conversation. The software on that "smart computer" is known as a **chatbot**.

Before working with the code provided in this lab that implements a chatbot, use your internet browser to explore some existing chatbots. Go to www. chatbot.org, click on the Awards tab, and then click on A.L.I.C.E.



Loebner Prize
› What is the Loebner Prize?
› Why a Loebner Prize
› 2013
› 2012
› 2011

Chatterbox Challenge
› What's Chatterbox Challenge?
› 2013
› 2012
› 2011

Chatbots.org Awards News

News about the evolution of conversational agents, virtual agents, chatterbots, brand agents, whatever terminology you prefer.

2012 Loebner Prize will be webcasted live!

by Erwin van Lun on 9 May 2012 in Award News

Summary: 2012 Loebner Prize, the first #turing test will be webcasted live! #ai #singularity

The 2012 Loebner Prize contest will take place in Bletchley Park on Tuesday 15th

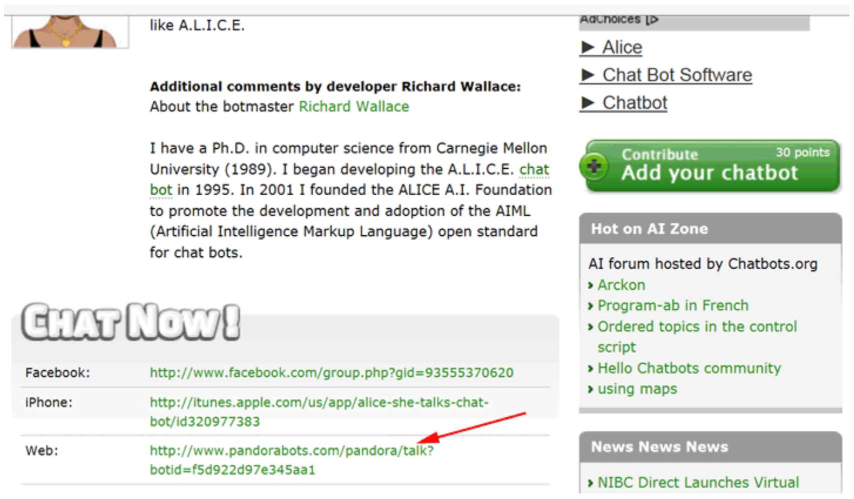
Awards News RSS Feed
Follow Chatbots

Contribute
Add award

Award diary
- Loebner Prize - Loebner Prize

Most Awarded
A.L.I.C.E. (19x)
Ultra Hal Assistant (13x)
Talk-Bot (12x)

To start a chat session with an actual chatbot, click on the link indicated here.



like A.L.I.C.E.

Additional comments by developer Richard Wallace:
About the botmaster Richard Wallace

I have a Ph.D. in computer science from Carnegie Mellon University (1989). I began developing the A.L.I.C.E. chat bot in 1995. In 2001 I founded the ALICE A.I. Foundation to promote the development and adoption of the AIML (Artificial Intelligence Markup Language) open standard for chat bots.

CHAT NOW!

Facebook:	http://www.facebook.com/group.php?gid=93555370620
iPhone:	http://itunes.apple.com/us/app/alice-she-talks-chat-bot/id320977383
Web:	http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1

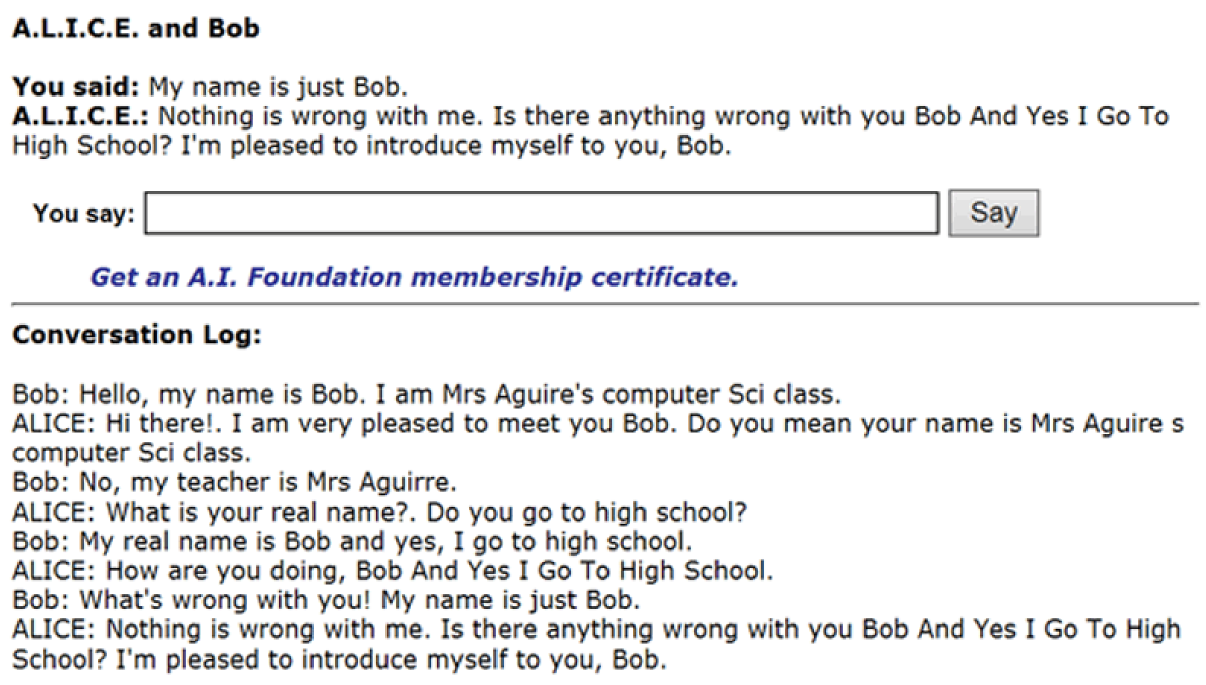
AdChoices
▶ Alice
▶ Chat Bot Software
▶ Chatbot

Contribute 30 points
Add your chatbot

Hot on AI Zone
AI forum hosted by Chatbots.org
› Arckon
› Program-ab in French
› Ordered topics in the control script
› Hello Chatbots community
› using maps

News News News
› NIBC Direct Launches Virtual

Shown here is an example of a chat session. Initiate your own by entering something in the text box and clicking on "Say."



A.L.I.C.E. and Bob

You said: My name is just Bob.
A.L.I.C.E.: Nothing is wrong with me. Is there anything wrong with you Bob And Yes I Go To High School? I'm pleased to introduce myself to you, Bob.

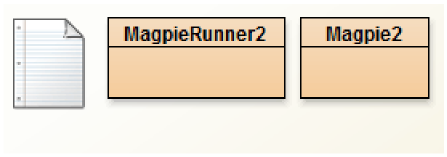
Conversation Log:

Bob: Hello, my name is Bob. I am Mrs Aguire's computer Sci class.
ALICE: Hi there!. I am very pleased to meet you Bob. Do you mean your name is Mrs Aguire s computer Sci class.
Bob: No, my teacher is Mrs Aguirre.
ALICE: What is your real name?. Do you go to high school?
Bob: My real name is Bob and yes, I go to high school.
ALICE: How are you doing, Bob And Yes I Go To High School.
Bob: What's wrong with you! My name is just Bob.
ALICE: Nothing is wrong with me. Is there anything wrong with you Bob And Yes I Go To High School? I'm pleased to introduce myself to you, Bob.

Download the necessary code:
The AP Magpie Lab (as well as the other three labs) can be downloaded at www.cccc.org . For your convenience Blue Pelican offers the equivalent download for the Magpie Lab only at www.bluepelicanjava.com/threeLabs/magpieLab.zip. Download and store in a convenient place on your computer.

In this lab series there are five discussions/activities where each is designed to exactly fit one class period:

The provided classes:
Make sure the *Magpie Code* folder is copied to your computer. Make a java project out of all of the classes that are within the *activity2* folder that is just under the *Magpie Code* folder. Here's how to do it with the IDE, BlueJ.



Click on Projects | New Projects and navigate to the folder on your computer where you normally keep you java projects, for example, *myJavaProjects* and then enter *myMagpieLab* as the folder name. Next, click on Projects | Import… , navigate to the *Magpie Code/activity2* folder, and then click Import. This will display all of the classes stored there. Compile both classes.

At this point it is assumed that the student is somewhat familiar with some of the methods of the *String* class, *if* statements, and loops. Examine the code in the *MagpieRunner2* class. It has a *main* method. To run any program in java, we must always run a *main* method.

Explore the code:
Notice in *main* that the *nextLine* method will retrieve whatever is entered via the keyboard. Also notice that the *while* loop has provisions for an "escape" that will end the chat session ("Bye").

Next, look at the code in the *Magpie2* class. Notice the *if* and *else if* statements. Also notice how the call to the *getRandomResponse* method chooses a response from a group of *String* objects.

Run *main* in the *MagpieRunner2* class and begin a chat session by entering something like, "I like pickles but I can't swim." After exchanging a few statements with the chatBot, enter "Bye" to end the session. Notice that our chatbot is fairly weak at this point.

Blue Pelican Magpie Lab – Day-2

Inserting some of our own code:

In the *Magpie2* class between the *else if* and the *else* near the top of the code, enter the following two *else if*s to make the chatbot recognize that you are talking about your pet or your teacher.

```
else if( (statement.indexOf(“cat”) >= 0) || (statement.indexof(“dog”) >= 0) )
        response = “Tell me more about your pets.”;
else if(statement.indexof(“Mr. Manson”) >= 0 )  //use your teachers name
        response = “He sounds like a wonderful teacher.”;
```

Test it:

Begin a chat session by running *main*. Be sure to use the key words, “cat”, “dog”, and “Mr. Manson” (or your teacher’s name) in the conversation.

To make your chatBot a little more powerful, think of several more key-words and an appropriate response for each. Three suggested key-word response pairs are shown below.

Key word	Response
“math”	“So, are you taking a math course?”
“hit”	“I don’t like violence. Let’s change the subject.”
“mother”	“Oh, do you have problems with your mother?”

Again, insert these into the *Magpie2* class in the form of *else if* statements as was done with “cat” and “dog”. Test by running *main*.

Guard against an empty *String*:

During a chat session, just press *Enter* without actually entering any characters and see what response you get. Try entering several spaces but no visible characters and see what response is returned.

As an assignment, insert yet another *if else* statement that will cause the response, "Please say something." to be returned and then test it during a chat session. Use the *String* methods *trim()* and *length()*. Recall that *trim* removes all leading and trailing white space from a *String*.

```
else if(statement.trim().length() == 0)
    response = "Please say something.";
```

Blue Pelican Computer Science
AP Picture Lab

Teacher Version 1.01

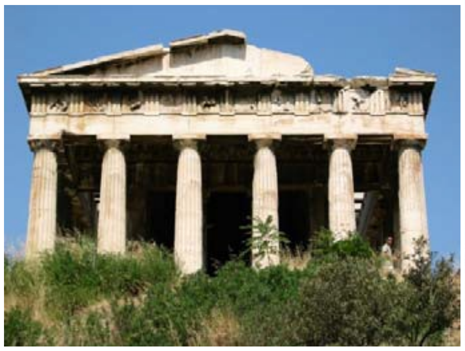
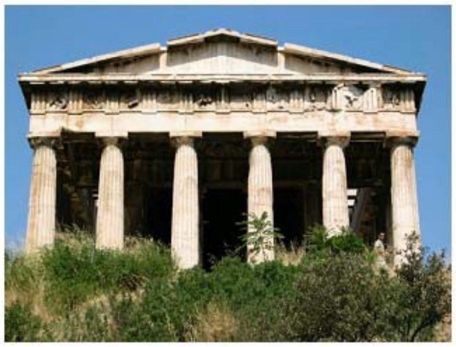
Edited by: Tanner Wright

Blue Pelican Picture Lab – Day-5

Repair a Greek Temple

Use *Picture Explorer* to display *temple.jpg*. In this lesson we will take the intact portion of the roof on the left and mirror image it across the center of the building so as to make it appear that the building has been repaired. Only a portion of the original picture needs to be mirrored. This will be somewhat similar to a method we created in an earlier lesson, *mirrorVertical*; however, the left side of **only the roof area** will be mirrored and **not** the entire left side of the picture.

Clicking on the picture in *PictureExplorer* will reveal that the center of the temple is at column 275. The portion of the roof on the left side necessary for the repair involves rows 27 – 96 and columns 13 – 275.

Within the *PictureTester* class, create the *void* method *repairTemple*. It receives no parameters.

- Create a *Picture* object called *pic* from the *temple.jpg* picture.
- Display *pic* using the *explore()* method.
- Create a *pixels* array from *pic* using *getPixels2D( )*.
- Set up two nested **traditional** for-loops.
  - Let the outer row-loop go through rows 27 – 96 inclusive.
  - Let the inner col-loop go through columns 13–275 inclusive.
- Set the Color of the pixel in its appropriate mirror image position.
- Call *explore()* to display *pic*.

Test this method by calling it from the *main* method in *PictureTester*.

At this point, the student is invited to explore the creation of collages and edge detection on his/her own.

Creating a collage (a composite picture consisting of several small pictures.

An important method of the *Picture* class that will be used to create a collage is:

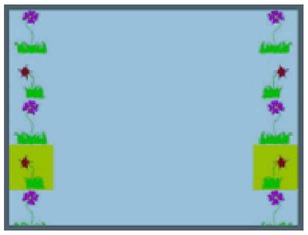
public void copy(Picture fromPic, int startRow, int startCol)

Below is an example of a typical call to this method using the *Picture* object *pic*.

pic.copy(smallPic, 40, 20);
Where smallPic is the small picture to be copied into the larger *pic* beginning at upper right coordinates row 40,column 20.

To see how this all works, study the following method in the *Picture* class:

```
public void createCollage()
 {
   Picture flower1 = new Picture("flower1.jpg");
   Picture flower2 = new Picture("flower2.jpg");
   this.copy(flower1,0,0);  //this is the picture object used to call this
                            //method.
   this.copy(flower2,100,0);
   this.copy(flower1,200,0);
   Picture flowerNoBlue = new Picture(flower2);
   flowerNoBlue.zeroBlue();
   this.copy(flowerNoBlue,300,0);
   this.copy(flower1,400,0);
   this.copy(flower2,500,0);
   this.mirrorVertical();
   this.write("collage.jpg");  //Creates a new disk file.
 }
```



Test with *testCollage* in *PictureTester*.

Edge Detection:
Many software products (like Photoshop$^{\text{TM}}$), devices including digital cameras, and the camera function in smart phones use edge detection.

A common way to look for an edge in a picture is compare the “color difference” between the picture in question and adjacent pixels.

The color difference will be called **color distance**:
Color distance is similar to the way distance between two 3D points is calculated in Algebra.

$$\mathrm{d} = \sqrt{(\mathrm{x}_2 - \mathrm{x}_1)^2 + (\mathrm{y}_2 - \mathrm{y}_1)^2\ (\mathrm{z}_2 - \mathrm{z}_1)^2}$$

Similarly, if r_1, g_1, and b_1 stand for the respective color values of pixel1 and r_2, g_2, and b_2 stand for the respective color values of pixel2, then the following gives the color distance between the two pixels:

$$\mathrm{d} = \sqrt{(\mathrm{r}_2 - \mathrm{r}_1)^2 + (\mathrm{g}_2 - \mathrm{g}_1)^2\ (\mathrm{b}_2 - \mathrm{b}_1)^2}$$

If *d* exceeds some predetermined value, then an edge has been detected.

Call the *Picture* method *colorDistance* to get the color distance between the current *Pixel* object used to call this method and a passed *Pixel* object.

Test this with the *testEdgeDetection* method in the *PictureTester* class.

Code for *repairTemple*

```
public void repairTemple()
{
       Picture pic = new Picture(temple.jpg);
       pic.explore();
       Pixel[][] pixels = pic.getPixels2D();
       int mirrorCol = 276;
       Pixel leftPixel = null;
       Pixel rightPixel = null;
       for (int row = 27; row < 97; row++)
       {
            for (int col = 13; col < mirrorCol; col++)
            {
                 leftPixel = pixels[row][col];
                 rightPixel = pixels[row][2* mirrorCol - col];
                 rightPixel.setColor(leftPixel.getColor());
            }
       }
       pic.explore();
 }
```

Quick Reference

```
DigitalPicture Interface
Pixel[][] getPixels2D()        // implemented in SimplePicture
void explore()                 // implemented in SimplePicture
boolean write(String fileName) // implemented in SimplePicture
```

```
SimplePicture Class (implements Digital Picture)
public SimplePicture()
public SimplePicture(int width, int height)
public SimplePicture(SimplePicture copyPicture)
public SinplePicture(String fileName)
public Pixel[][] getPixels2D()
public void explore()
public boolean write(String fileName)
```

```
Picture Class (extends SimplePicture)
public Picture()
public Picture(int height, int width)
public Picture(Picture copyPicture)
public Picture(String fileName)
public Pixel[][] getPixels2D()        // from SimplePicture
public void explore()                 // from SimplePicture
public boolean write(String fileName) // from SimplePicture
```

```
Pixel Class
public double colorDistance(Color testColor)
public double getAverage()
public int getRed()
public int getGreen()
public int getBlue()
public Color getColor()
public int getRow()
public int getCol()
public void setRed(int value)
public void setGreen(int value)
public void setBlue(int value)
public void setColor(Color newColor)
```

```
java.awt.Color Class
public Color(int r, int g, int b)
public int getRed()
public int getGreen()
public int getBlue()
```

Blue Pelican Computer Science
AP Elevens Lab

Teacher Version 1.01

Edited by: Tanner Wright

Blue Pelican Elevens Lab – Day-6

The game of Eleven is a solitaire game that uses a deck of 52 standard playing cards. The point values of numbered cards are the actual numbers on the card. The remaining cards are valued as follows:

Ace = 1, Jack = 11, Queen= 12, King = 13.

The rules of the game are:

1. The deck is shuffled, and nine cards are dealt "face up" from the deck to the board.

2. Then the following sequence of steps is repeated:

   a. The player removes each pair of cards (A, 2, … , 10) that total 11, e.g., an 8 and a 3, or a 10 and an A. An ace is worth 1, and suits are ignored when determining cards to remove.

   b. Any triplet consisting of a J, a Q, and a K is also removed by the player. Suits are also ignored when determining which cards to remove.

   c. Cards are dealt from the deck if possible to replace the cards just removed.

The game is won when the deck is empty and no cards are left on the table.

Following is a sample game:

Cards on the Table	**Explanation**
K♠ 10♦ J♣ 2♣ 2♥ 9♦ 3♥ 5♠ 5♦	initial deal
K♠ 10♦ J♣ 7♦ 2♥ Q♠ 3♥ 5♠ 5♦	remove 2♣ (either 2 would work) and 9♦

A♠ 10♦ 9♣ 7♦ 2♥ 7♣ 3♥ 5♠ 5♦	remove J♣ Q♠ K♠
A♠ 10♦ 10♠ 7♦ 3♣ 7♣ 3♥ 5♠ 5♦	remove 9♣ and 2♥ (removing A♠ and 10♦ would have been legal here too)
2♠ 10♦ 9♠ 7♦ 3♣ 7♣ 3♥ 5♠ 5♦	remove A♠ and 10♠ (10♦ could have been removed instead)
A♣ 10♦ K♦ 7♦ 3♣ 7♣ 3♥ 5♠ 5♦	remove 2♠ and 9♠
6♦ K♣ K♦ 7♦ 3♣ 7♣ 3♥ 5♠ 5♦	remove A♣ and 10♦
2♦ K♣ K♦ 7♦ 3♣ 7♣ 3♥ 5♠ Q♦	remove 6♦ and one of the 5s; no further plays are possible; game is lost.

The GUI:
Navigate through the files on your computer (using Windows Explorer if you are using a PC) and find the *Elevens.jar* file in the *Eleven Code/Activity Starter/Activity6StarterCode* folder. Do not try to bring this file into your java project; rather, just double click on it and the file should execute.

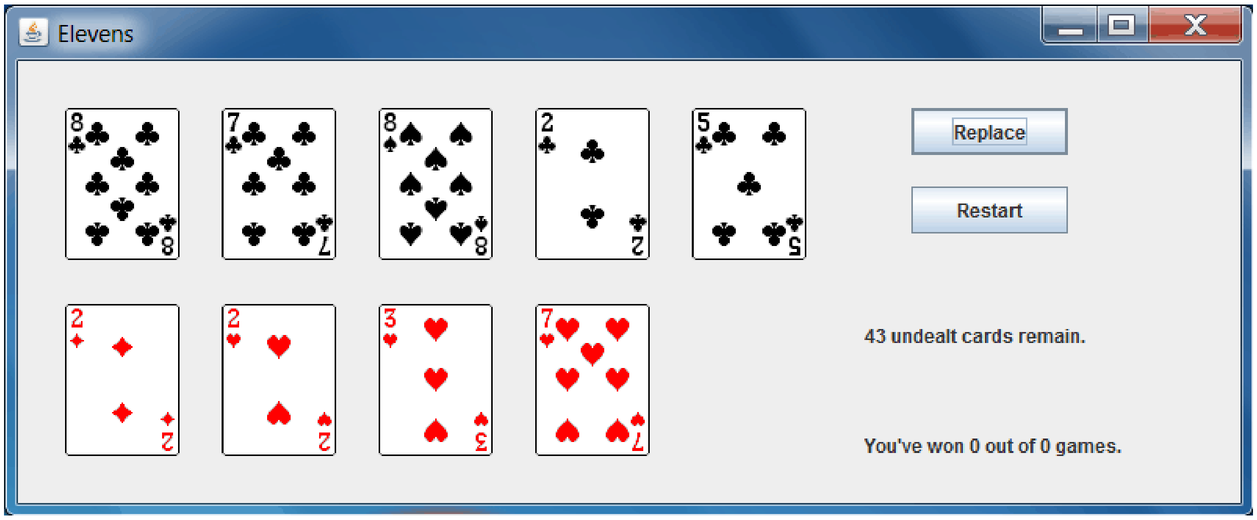
If the file does not directly execute, try running it from a command line by doing the following:

Copy the Elevens.jar file to the root of the C:\ folder.

Bring up a command line prompt and enter the following:

```
cd c:\
java –jar Elevens.jar
```

The result should look something like this:



43 undealt cards remain.

You've won 0 out of 0 games.

To use this interactive GUI (graphical user interface):

- Click on a card to select/deselect it.
- After making selections, click **Replace**. (It does a check to see if the selection was legal. If so, new cards fill the selected positions.)
- Click on **Restart** to start a new game.

Play a few games and think about what code is executing in the background to make all it all happen. We have already written some of that code and will soon be writing more.