The AP Computer Science A Labs

CHAPTER 9

I don't like museums, I like labs.
—Amit Kalantri

Chapter Goals

- The Magpie Lab
- The Elevens Lab
- The Picture Lab

The AP Computer Science A labs were developed as a replacement for the GridWorld case study. Starting in May 2015, there will be no exam questions on GridWorld. And there will be no specific questions that require knowledge of the content of the labs. Instead, new test questions will focus on concepts from the AP Java subset that are emphasized in the labs.

What follows below is a brief summary of the labs, the concepts they illustrate, the concepts they particularly emphasize, and some sample multiple-choice questions based on these concepts.

THE MAGPIE LAB

In this lab, students modify a chatbot, which is a computer program designed to simulate an intelligent conversation between a computer and a human user. Students enter phrases, the computer searches for keywords, then comes up with an intelligent-seeming response.

Student activities include:

- Working through the Magpie code (`if` statements)
- Using `Magpie` and `String` methods (`while` loops, strings, and Javadoc)
- Using an array of possible responses in generating a random response from the computer (arrays, `ArrayLists`, and random integers)
- Improving the search to find keywords that are complete words, not substrings buried in other strings (`String` methods)
- Transforming a computer response based on the format of the statement entered by the user (`String` methods)

Special Emphasis

STRING METHODS

The `String` methods `substring` and `indexOf` are used continually in this lab. Be sure that you recall

- The first index of a `String` is `0`.
- The method call `s.substring(start, end)` returns the substring of `s` starting at index `start` but ending at index `end-1`.
- The method call `s.indexOf(sub)` returns the index of the first occurrence of `sub` in `s`.
- `s.indexOf(sub)` returns `-1` if `sub` is not in `s`.

You should be nimble and well practiced in processing strings.

The following type of code is used repeatedly in the lab to look for multiple occurrences of a substring in a given string:

```
int pos = s.indexOf(someSubstring);
while (pos >= 0)                    //the substring was found
{
    doSomething();
    s = s.substring(pos + 1);  //throw away all characters of s
                               //up to and including someSubstring

    pos = s.indexOf(someSubstring);  //Is there another occurrence
                                     //of someSubstring?
}
```

A modified version of the above code, using some combination of a loop, `indexOf`, and `substring`, can be used to

- count number of occurrences of `substring` in `str`.
- replace all occurrences of `substring` in `str` with `replacementStr`.
- remove all occurrences of `substring` in `str`.

On the AP exam, there will almost certainly be at least one free-response question that requires you to manipulate strings in this way.

RANDOM ELEMENT SELECTION

Another skill that is demonstrated in this lab is returning a random element from an array or `ArrayList`. For example, suppose `responses` is an `ArrayList<String>` of surprised responses the computer may make to a user's crazy input. If the contents of `responses` are currently

0	1	2	3	4	5
Oh my!	Say what?	No!	Heavens!	You're kidding me.	Jumping Jellybeans

You should be able to randomly return one of these responses. The key is to select a random index from `0` to `5`, inclusive, and then return the string in the `responses` list that is at that index.

Recall that the expression `(int)(Math.random()*howMany)` generates a random `int` in the range `0...howMany-1`. In the given example, `howMany` is `6`. The piece of code that returns a random response is:

```
int randIndex = (int) (Math.random() * 6);
String response = responses.get(randIndex);
```

CONDITIONALS: `if...else` STATEMENT

The Magpie lab is loaded with conditionals, searching for keywords that will trigger different responses from the chatbot (computer). Using `if` and `if...else` should be second nature to you.

Example

The user will enter a `sentence` and the chatbot will produce a `chatBotReply`.

```
if (sentence.indexOf ("love") != -1)
{
    if (sentence.indexOf ("you") != -1)
        chatBotReply = "I'm in heaven!";
    else
        chatBotReply = "But do you love me?";
}
else
   chatBotReply = "My heart is in pieces on the floor.";
```

Here are some possible sentences that the user may enter, with the corresponding `chatBoxReply`:

Sentence	chatBoxReply
`I love chocolate cake.`	`But do you love me?`
`I love chocolate cake; do you?`	`I'm in heaven.`
`I hate fudge.`	`My heart is in pieces on the the floor.`

If the substring `"love"` isn't in the sentence, the opening test will be `false`, and execution skips to the `else` outside the braces, producing the `chatBotReply` `"My heart is in pieces on the floor"`. If `sentence` contains both `"love"` and `"you"`, the first test in the braces will be `true`, and the `chatBotReply` will be `"I'm in heaven!"` The middle response `"But do you love me?"` will be triggered by a sentence that contains `"love"` but doesn't contain `"you"`, causing the first test in the braces to be `false`, and the `else` part in the braces to be executed.

THE ELEVENS LAB

In this lab, students simulate a game of solitaire, Elevens, and a related game, Thirteens. A GUI is provided for the labs to make the game interesting and fun to play. You are not required to know about GUIs.

Student activities include:

- Creating a `Card` class (objects, classes, and `Strings`)
- Creating a `Deck` class (arrays, `ArrayLists`, conditionals, loops)
- Shuffling the deck (`Math.random`, list manipulation)
- Writing an `ElevensBoard` class, using an abstract `Board` class (inheritance, abstract classes)
- Testing and debugging
- Playing the game

Special Emphasis

SHUFFLING

Several different algorithms are discussed for shuffling an array of elements. A key ingredient of a good shuffle is generation of random integers. For example, to shuffle a deck of 52 cards in an array may require a random int from 0 to 51:

```
int cardNum = (int) (Math.random() * 52);
```

(Recall that the multiplier in parentheses is the number of possible random integers.)

The following code for shuffling an array of Type elements is used often:

```
for (int k = arr.length - 1; k > 0; k--)
{
    //Pick a random index in the array from 0 to k
    int index = (int) (Math.random() * (k + 1));
    //Swap randomly selected element with element at position k
    Type temp = arr[k];
    arr[k] = arr[index];
    arr[index] = temp;
}
```

WRITING SUBCLASSES

On the AP exam, you will probably be asked to write a subclass of a given class. Don't forget the extends keyword:

```
public class Subclass extends Superclass
```

Recall that constructors are not inherited, and if you use the keyword super in writing a constructor for your subclass, the line containing it should precede any other code in the constructor..

Example

```
public class Dog
{
    private String name;
    private String breed;

    public Dog (String aName, String aBreed)
    {
        name = aName;
        breed = aBreed;
    }
        ...
}

public class Poodle extends Dog
{
    private boolean needsGrooming;

    public Poodle (String aName, String aBreed, boolean grooming)
    {
        super(aName, aBreed);
        needsGrooming = grooming;
    }
        ...
}
```

In the Elevens lab there's extensive discussion about using an abstract class, `Board`, to represent a game board on which the game of Elevens will be played.

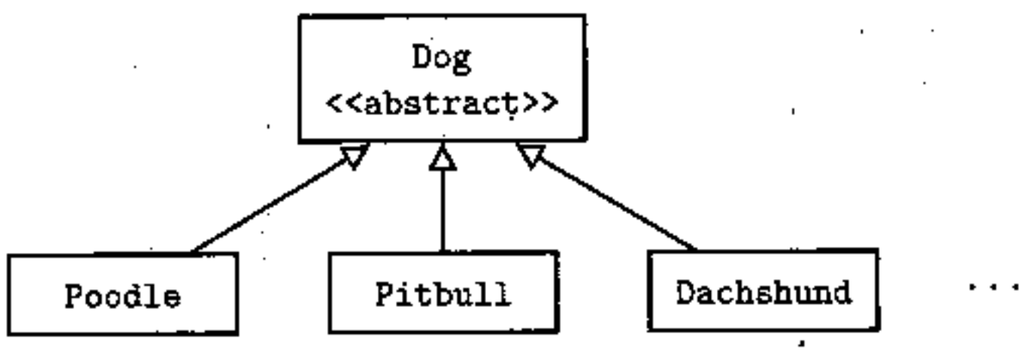
The advantage of the abstract class is that its use can be extended for other solitaire games played on similar boards. In the Elevens lab, the subclass `ElevensBoard` is written, which is specific to the game of Elevens. Further on in the lab, another subclass `ThirteensBoard` is discussed, which applies to a different, but similar game, Thirteens.

The abstract `Board` class contains methods that are common to all games that would be played on a `Board`, like `deal`. But methods that would pertain to moves on the board, like `isLegal`, would be different for each of the specific games. These methods are therefore declared `abstract` in the superclass, but are overridden in the subclasses.

Note: If you're writing a concrete (nonabstract) subclass of an abstract superclass, you must be sure to write an implementation for every abstract method of the superclass.

POLYMORPHISM

Consider this hierarchy of classes, and the declarations that follow it:



Suppose the `Dog` class has this method:

```
public abstract void eat();
```

And each of the subclasses, `Poodle`, `PitBull`, `Dachshund`, etc., has a different, overridden `eat` method. Now suppose that `allDogs` is an `ArrayList<Dog>` where each `Dog` declared above has been added to the list. Each `Dog` in the list will be processed to eat by the following lines of code:

```
for (Dog d: allDogs)
    d.eat();
```

Polymorphism is the process of selecting the correct `eat` method, during run time, for each of the different dogs.

TESTING AND DEBUGGING

In the Elevens lab, a lot of emphasis is placed on testing and debugging code as you write it. Here are some general principles:

- Start simple. For example, if writing a `Deck` class, start with a deck that contains just 2 or 3 cards.
- Always have a driver class (one with a `main` method) to test the current class you're writing.
- In your class, start with a constructor. You want to be sure you can create your object.
- After the constructor, write a `toString` method for clear and easy display. You want to be able to "see" the results of running your code.

SIMULATING RANDOM EVENTS

Flipping a coin, tossing a die, or picking a random card from a deck. Those random numbers again! If there are k possible outcomes, each of them equally likely, be sure you can generate a random int from 0 to k-1.

THE PICTURE LAB

In this lab, students manipulate digital pictures using two-dimensional arrays. Code for the GUI is provided in the lab.

The main concept emphasized is traversal of two-dimensional arrays. Other concepts used are UML diagrams, binary numbers, inheritance, interfaces, abstract methods, constants, and program analysis.

Student activities include:

- Learning how colors are stored in a program.
- Modifying a picture.
- Creating a mirror image of a picture.
- Mirroring part of a picture.
- Creating a collage.
- Detecting the edge of a picture.

Special Emphasis

PROCESSING A 2-D ARRAY

A matrix is stored as an array of rows, each of which is also an array. In the lab, a for-each loop is often used for traversal. Here is an example that traverses an array of int:

```
for (int[] row : matrix)      //for each row array in the matrix
    for (int num : row)       //for each int element in the current row
        doSomething();
```

Here is what doSomething can do:

- Access each element in the matrix (count, add, compare, etc.)

Here is what doSomething cannot do:

- Replace an element with another.

Suppose the matrix is an array of objects that can be changed with mutator methods. The for-each loop can be used not only to access elements, but also to modify them. (No replacing with new elements, however). The following code is OK.

```
for (Clock[] row : clockMatrix)
    for (Clock c : row)
        c.setTime(t);
```

MIRROR IMAGES

A large part of the lab is spent coming up with algorithms that create some kind of mirror image of a matrix. Students are asked to reflect across mirrors placed somewhere in the center of the matrix, horizontally, vertically, or diagonally.

Note that if a vertical mirror is placed down the center of a matrix, so that all elements to the left of the mirror are reflected across it, the element `mat[row][col]` reflects across to element `mat[row][numCols-col-1]`.

You should teach yourself to trace the following type of code:

```
public static void matrixMethod(int[][] mat)
{
    int height = mat.length;
    int numCols = mat[0].length;
    for (int col = 0; col < numCols; col++)
        for (int row = 0; row < height/2; row++)
            mat[height - row - 1][col] = mat[row][col];
}
```

What does it do? How does it transform the matrix below?

```
2 3 4
5 6 7
8 9 0
1 1 1
```

Solution: The algorithm reflects the matrix from top to bottom across a horizontal mirror placed at its center.

```
height = 4, numCols = 3
col takes on values 0, 1, and 2
row takes on values 0 and 1
```

Here are the replacements that are made:

```
col = 0, row = 0: mat[3][0] = mat[0][0]
         row = 1: mat[2][0] = mat[1][0]

col = 1, row = 0: mat[3][1] = mat[0][1]
         row = 1: mat[2][1] = mat[1][1]

col = 2, row = 0: mat[3][2] = mat[0][2]
         row = 1: mat[2][2] = mat[1][2]
```

This transforms the matrix into

```
2 3 4
5 6 7
5 6 7
2 3 4
```

Note that a for-each loop was not used in the traversal, because elements in the matrix are being replaced.

BASE 2, BASE 8, BASE 16

Binary (base 2) and hexadecimal (base 16) numbers are discussed in the Picture lab as they apply to storage of colors. You need to be able to convert a given number in base *b* to a decimal (base 10) number (see p. 62).

Example

Convert the following octal (base 8) number to a decimal number: 123_{oct}
Solution:

$$\begin{aligned} 123_{oct} &= (1)(8^2)+(2)(8^1)+(3)(8^0) \\ &= 64+16+3 \\ &= 83_{dec} \end{aligned}$$

For a base 16 conversion, you'll be given that A, B, C, D, E, and F represent 10, 11, 12, 13, 14, and 15, respectively.

Example

Convert $A2B_{hex}$ to a decimal number.
Solution:

$$\begin{aligned} A2B_{hex} &= (A)(16^2)+(2)(16^1)+(B)(16^0) \\ &= 2560+32+11 \\ &= 2603_{dec} \end{aligned}$$

Can you go in the other direction, namely, start with a decimal number and convert it to a different base?

Example

Convert 25 to a binary (base 2) number.
Solution:
Pull out the highest power of 2 less than or equal to 25, which is 16. Thus,

$$25 = 16+9$$

Now pull out the highest power of 2 less than or equal to 9, which is 8. Thus,

$$25 = 16+8+1$$

Now write 25 as a sum of powers of 2, making sure to include the missing powers that are less than the highest power in the first step of the solution:

$$\begin{aligned} 25 &= 16+8+1 \\ &= (1)(2^4)+(1)(2^3)+(0)(2^2)+(0)(2^1)+(1)(2^0) \\ &= 11001_{bin} \end{aligned}$$

Note: you must be careful to use 0's as place holders for the missing powers of 2.

Chapter Summary

String manipulation and matrix processing are the two big topics you should master. Review the meanings and boundary conditions of the parameters in the `String` methods `substring` and `indexOf`. For matrices, you should nail down both the row-column and for-each traversals. Remember, you cannot use a for-each loop for the replacement of elements.

Be sure you can hand-execute tricky matrix algorithms, like those used for modifying matrices using mirror images.

A matrix is an array of row-arrays, so familiarize yourself with the the use of a method with an array parameter to process the rows of a matrix.

Array manipulation is another big topic. Be sure you know how to shuffle the elements of an array.

Other concepts emphasized in the labs are inheritance and polymorphism, writing subclasses, abstract classes, simulation of events using random numbers, multi-base numbers, and conditional (`if...else`) statements. You should have all of these at your fingertips.

MULTIPLE-CHOICE QUESTIONS ON THE LAB CONCEPTS

1. For ticket-selling purposes, there are three categories at a certain theater:

Age	Category
65 or above	Senior
From 18 to 64 inclusive	Adult
Below 18	Child

Which of the following code segments will assign the *correct* string to `category` for a given integer age?

I
```
if (age >= 65)
    category = "Senior";
if (age >= 18)
    category = "Adult";
else
    category = "Child";
```

II
```
if (age >= 65)
    category = "Senior";
if (18 <= age <= 64)
    category = "Adult";
else
    category = "Child";
```

III
```
if (age >= 65)
    category = "Sénior";
else if (age >= 18)
    category = "Adult";
else
    category = "Child";
```

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

2. What is the output of the following code segment?

```
String s = "How do you do?";
int index = s.indexOf("o");
while (index >= 0)
{
    System.out.print(index + " ");
    s = s.substring(index + 1);
    index = s.indexOf("o");
}
```

(A) 1 3 2 3
(B) 2 4 3 4
(C) 1 5 8 12
(D) 1 5 8 11
(E) No output because of an `IndexOutOfBoundsException`

3. Consider the following method `removeAll` that creates and returns a string that has stripped its input phrase of all occurrences of its single-character `String` parameter `ch`.

```
Line 1: public static String removeAll(String phrase, String ch)
Line 2: {
Line 3:     String str = "";
Line 4:     String newPhrase = phrase;
Line 5:     int pos = phrase.indexOf(ch);
Line 6:     if (pos == -1)
Line 7:         return phrase;
Line 8:     else
Line 9:     {
Line 10:        while (pos >= 0)
Line 11:        {
Line 12:            str = str + newPhrase.substring(0, pos - 1);
Line 13:            newPhrase = newPhrase.substring(pos + 1);
Line 14:            pos = newPhrase.indexOf(ch);
Line 15:            if (pos == -1)
Line 16:                str = str + newPhrase;
Line 17:        }
Line 18:        return str;
Line 19:    }
Line 20: }
```

The method doesn't work as intended. Which of the following changes to the `removeAll` method will make it work as specified?

(A) Change Line 10 to
```
while (pos >= -1)
```
(B) Change Line 12 to
```
str = str + newPhrase.substring(0, pos);
```
(C) Change Line 13 to
```
newPhrase = newPhrase.substring(pos);
```
(D) Change Line 14 to
```
pos = phrase.indexOf(ch);
```
(E) Change Line 16 to
```
str = str + newPhrase.substring(pos + 1);
```

4. A programmer has written a program that "chats" to a human user based on statements that the human inputs. The program contains a method `findKeyWord` that searches an input statement for a given keyword. The `findKeyWord` method contains the following line of code:

```
pos = statement.indexOf(word);
```

Suppose `pos` has a value `>= 0`, that is, `word` was found. The programmer now wants to test that an actual word was found, not part of another word. For example, if "cat" is the keyword, the programmer needs to check that it's not part of "catch" or "category." Here is the code that tests if `word` is a stand-alone word. (You may assume that `statement` is all lowercase and contains only letters and blanks.)

```
pos = statement.indexOf(word);
//Check for first or last word
if (pos == 0 || pos + word.length() == statement.length())
{
    before = " ";
    after = " ";
}
else
{
    before = statement.substring(pos - 1, pos);
    after = statement.substring(pos + word.length(),
            pos + word.length() + 1);
    if (/* test */)
        //then a stand-alone word was found ...
    else
        //word was part of a larger word
}
```

Which replacement for `/* test */` will give the desired result?

(A) `(before < "a" || before > "z") && (after < "a" || after > "z")`

(B) `(before > "a" || before < "z") && (after > "a" || after < "z")`

(C) `(before.compareTo("a") < 0 && before.compareTo("z") > 0) ||`
`(after.compareTo("a") > 0 && after.compareTo("z") < 0)`

(D) `(before.compareTo("a") > 0 && before.compareTo("z") < 0) &&`
`(after.compareTo("a") > 0 && after.compareTo("z") < 0)`

(E) `(before.compareTo("a") < 0 || before.compareTo("z") > 0) &&`
`(after.compareTo("a") < 0 || after.compareTo("z") > 0)`

5. A program that simulates a conversation between a computer and a human user generates a random response to a user's comment. All possible responses that the computer can generate are stored in an array of `String` called `allResponses`. The method given below, `getResponse`, returns a random response string from the array.

```
/** Precondition:  array allResponses is initialized with strings.
 *  Postcondition: returns a random response from allResponses.
 */
public String getResponse();
{ /* implementation */ }
```

Which is a correct `/* implementation */`?

(A)
```
int i = (int) (Math.random() * allResponses.length);
return allResponses[i];
```

(B)
```
return (String) (Math.random() * allResponses.length);
```

(C)
```
int i = Math.random() * allResponses.length;
return allResponses[i];
```

(D)
```
int i = (int) (Math.random() * (allResponses.length - 1));
return allResponses[i];
```

(E)
```
return (int) (Math.random() * allResponses.length);
```

Questions 6 and 7 refer to the `Deck` class described below.

A `Deck` class contains an array `cards` with an even number of `Card` values and a `final` variable `NUMCARDS`, which is an odd integer.

6. Here are two possible algorithms for shuffling the deck.

Algorithm 1

Initialize an array of `Card` called `shuffled` of length `NUMCARDS`.
Set `k` to `0`.
For `j=0` to `NUMCARDS/2-1`
- Copy `cards[j]` to `shuffled[k]`
- Set `k` to `k+2`

Set `k` to `1`.
For `j=NUMCARDS/2` to `NUMCARDS-1`
- Copy `cards[j]` to `shuffled[k]`
- Set `k` to `k+2`

Algorithm 2

Initialize an array of `Card` called `shuffled` containing `NUMCARDS` slots.
For `k=0` to `NUMCARDS-1`
- Repeatedly generate a random integer `j` from `0` to `NUMCARDS-1`, until `cards[j]` contains a card not marked as empty
- Copy `cards[j]` to `shuffled[k]`
- Set `cards[j]` to empty

Which is a *false* statement concerning Algorithms 1 and 2?

(A) A disadvantage of Algorithm 1 is that it won't generate all possible deck permutations.
(B) For Algorithm 2, to determine the last element shuffled requires an average of `NUMCARDS` calls to the random number generator.
(C) Algorithm 2 will lead to more permutations of the deck than Algorithm 1.
(D) In terms of run time, Algorithm 2 is more efficient than Algorithm 1.
(E) If Algorithm 1 is repeated several times, it may return the deck to its original state.

7. The following `shuffle` method is used to shuffle the cards in the `Deck` class.

```
Line 1: public void shuffle()
Line 2: {
Line 3:     for (int k = NUMCARDS; k > 0; k--)
Line 4:     {
Line 5:         int randPos = (int) (Math.random() * (k + 1));
Line 6:         //swap randomly selected card with card at position k
Line 7:         Card temp = cards[k];
Line 8:         cards[k] = cards[randPos];
Line 9:         cards[randPos] = temp;
Line 10:    }
Line 11: }
```

The method does not work as intended. Which of the following changes should be made to correct the method?

(A) Replace Line 3 with

```
for (int k = NUMCARDS; k >= 0; k--)
```

(B) Replace Line 3 with

```
for (int k = NUMCARDS - 1; k > 0; k--)
```

(C) Replace Line 3 with

```
for (int k = 1; k <= NUMCARDS; k++)
```

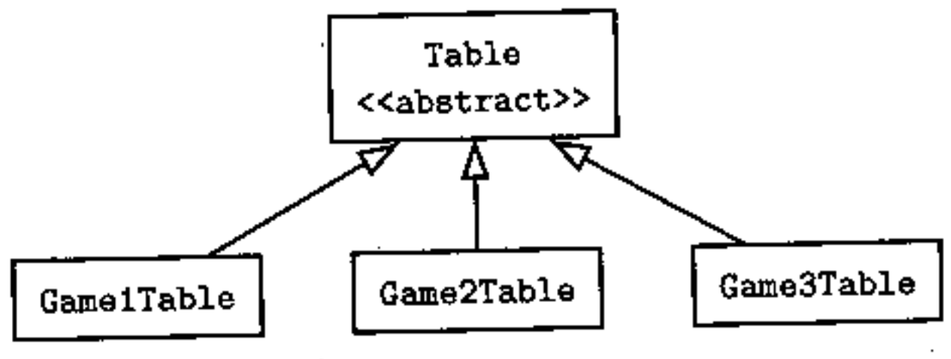
(D) Replace Line 5 with

```
int randPos = (int) (Math.random() * k);
```

(E) Replace Lines 7 – 9 with

```
Card temp = cards[randPos];
cards[randPos] = cards[k];
cards[k] = temp;
```

8. A programmer wants to simulate several different but related solitaire games, each of which uses a standard 52-card deck. Each game starts with 10 cards dealt face up on a table. The game types differ in the rules that allow groups of cards to be discarded from the table and new cards to be dealt from the remainder of the deck. For each of the solitaire games, a winning game occurs if there are no cards left in the pile or on the table. To represent the table for the games, the programmer will use the inheritance relationship shown below.



Which of the following methods should be abstract in the `Table` class?

I `dealNextCard`, which provides the next card for the table

II `checkForMove`, which returns `true` if a move is possible, `false` otherwise

III `isWinningGame`, which returns `true` if both the table and deck of cards are empty

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

Questions 9 and 10 refer to the following.

A word creation game uses letter tiles, where each tile has a letter and a point value for scoring purposes. A `Tile` class is used to represent a letter tile.

```
public class Tile
{
    private String letter;
    private int pointValue;

    //Constructors and other methods are not shown.
}
```

9. The `Tile` class contains a `toString` method that creates a `String` containing the letter and point value of a `Tile`. The string should be in the following format:

   `Letter` *`letter`* `(point value =` *`pointValue`*`)`

   For example,

   ```
   Letter A (point value = 1)
   Letter Z (point value = 10)
   ```

   Consider the `toString` method below:

   ```
   public String toString()
   {
       return /* code */
   }
   ```

   Which `/* code */` leads to correct output?

   (A) `Letter + "letter " + "(point value = " + pointValue + ")";`
   (B) `"Letter " + letter + ("point value = " + pointValue);`
   (C) `Letter + this.letter + " (point value = " + pointValue + ")";`
   (D) `"Letter " + letter + " (point value = " + (String) pointValue + ")";`
   (E) `"Letter " + letter + " (point value = " + pointValue + ")";`

10. Any two tiles in the word game that have the same letter also have the same point value, but the opposite is not necessarily true. For example, all the vowels have a point value of 1. Two tiles are said to match if they have the same letter. Consider the following `matches` method for the `Tile` class.

    ```
    /** @return true if the letter on this tile equals the letter
     *  on otherTile */
    public boolean matches(Tile otherTile)
    { return /* code */; }
    ```

    Which replacements for `/* code */` return the desired result? Note: You may *not* assume that the `Tile` class has its own `equals` method.

    I `letter == otherTile.letter`

    II `this.equals(otherTile)`

    III `letter.equals(otherTile.letter)`

    (A) I only
    (B) II only
    (C) III only
    (D) II and III only
    (E) I and III only

11. Consider the following method.

```
public static void alterArray(int[] arr)
{
    int mid = arr.length/2;
    for (int i = 0; i < mid; i++)
    {
        int temp = arr[i];
        arr[i] = arr[arr.length - i - 1];
        arr[arr.length - i - 1] = temp;
    }
}
```

If the current state of a matrix `mat` is

```
2 7 9 5
8 1 4 3
6 5 0 9
```

which matrix will result from the method call `alterArray(mat[2])`?

(A)
```
2 7 9 5
3 4 1 8
6 5 0 9
```

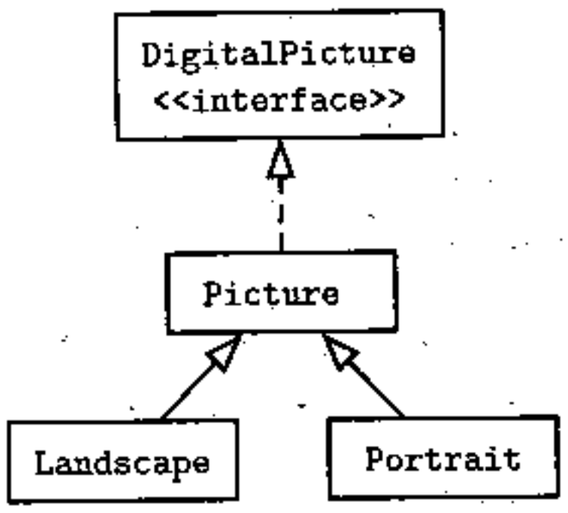
(B)
```
2 7 0 5
8 1 4 3
6 5 9 9
```

(C)
```
5 9 7 2
3 4 1 8
9 0 5 6
```

(D)
```
2 7 9 5
8 1 4 3
9 0 5 6
```

(E)
```
5 9 7 2
8 1 4 3
6 5 0 9
```

12. Consider a program to manipulate images. The inheritance hierarchy is as follows:



You may assume that `Picture` has a default (no-argument) constructor, but that `Landscape` and `Portrait` do not have any constructors. Which of the following declarations will compile?

I `DigitalPicture p = new Portrait();`

II `Landscape p = new Picture();`

III `DigitalPicture p = new DigitalPicture();`

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

13. The color of a pixel can be represented using the RGB (Red, Green, Blue) color model, which stores integer values for red, green, and blue, each ranging from 0 to 255. A value of 0 represents none of that color, while 255 represents the maximum. Consider a `Pixel` class that, among other methods, contains methods `getRed`, `getGreen`, and `getBlue`. These methods return integer values of those colors for that `Pixel`. There are also methods `setRed`, `setGreen`, and `setBlue`, which allow these values to be changed. For example, `setBlue(250)` would set the amount of blueness for that pixel to 250.

Consider a `Picture` class and a private instance variable `pixels`, where `pixels` is a two-dimensional array of `Pixel` objects. A method `removeRed` in the `Picture` class sets the red value of every pixel to zero:

```
public void removeRed()
{
    for (int row = 0; row < numRows; row++)
        for (int col = 0; col < numCols; col++)
        {
            /* code to set red value to 0 */
        }
}
```

Which is a correct replacement for */* code to set red value to 0 */*?

I
```
Pixel p = pixels[row][col];
p.setRed(0);
```

II
```
pixels[row][col].setRed(0);
```

III
```
pixels[row][col].getRed() = 0;
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II, and III

14. Consider a class `MatrixStuff` that has a private instance variable

```
private int[][] mat;
```

The following method uses a vertical mirror down the center of a matrix to reflect the left half of the matrix onto the right. The following two examples show the result of mirroring a two-dimensional array of numbers from left to right vertically. (Another way of saying this is that the right half of the matrix is replaced by a vertical mirror image of the left half.)

Example 1:

mat					mat after mirroring				
1	2	3	4	5	1	2	3	2	1
6	7	8	9	10	6	7	8	7	6
11	12	13	14	15	11	12	13	12	11

Example 2:

mat				mat after mirroring			
1	2	3	4	1	2	2	1
5	6	7	8	5	6	6	5
9	10	11	12	9	10	10	9

```
public static void mirrorVerticalLeftToRight(int[][] mat)
{
    int width = mat[0].length;
    int numRows = mat.length;
    for (int row = 0; row < numRows; row++)
        for (int col = 0; col < width/2; col++)
            /*  element assignments */
}
```

Which replacement for `/* element assignments */` will make the method work as intended?

(A) `mat[row][col] = mat[row][width - col];`
(B) `mat[row][width - col] = mat[row][col];`
(C) `mat[row][width - 1 - col] = mat[row][col];`
(D) `mat[row][col] = mat[row][width - 1 - col];`
(E) `mat[row][width - 1 - col] = mat[col][row];`

15. Consider a square matrix in a class that has a private instance variable `mat`:

```
private int[][] mat;
```

Method `alter` in the class changes `mat`:

```
public void alter()
{
    for (int row = 1; row < mat.length; row++)
        for (int col = 0; col < row; col++)
            mat[col][row] = mat[row][col];
}
```

If `mat` has current value

```
{{1, 2, 3},
{4, 5, 6},
{7, 8, 9}}
```

what are the contents of `mat` after method `alter` has been executed?

(A)
```
{{1, 4, 7},
 {4, 5, 8},
 {7, 8, 9}}
```

(B)
```
{{1, 4, 7},
 {2, 5, 8},
 {3, 6, 9}}
```

(C)
```
{{1, 2, 3},
 {2, 5, 6},
 {3, 6, 9}}
```

(D)
```
{{9, 6, 3},
 {8, 5, 6},
 {7, 8, 9}}
```

(E)
```
{{1, 2, 3},
 {4, 5, 2},
 {7, 4, 1}}
```