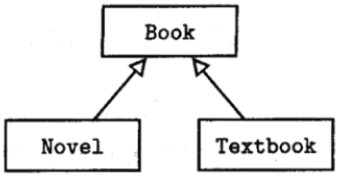
1. Consider this inheritance hierarchy, in which `Novel` and `Textbook` are subclasses of `Book`.



Which of the following is a *false* statement about the classes shown?

(A) The `Textbook` class can have private instance variables that are neither in `Book` nor `Novel`.

(B) Each of the classes—`Book`, `Novel`, and `Textbook`—can have a method `computeShelfLife`, whose code in `Book` and `Novel` is identical, but different from the code in `Textbook`.

(C) If the `Book` class has private instance variables `myTitle` and `myAuthor`, then `Novel` and `Textbook` inherit them but cannot directly access them.

(D) Both `Novel` and `Textbook` inherit the constructors in `Book`.

(E) If the `Book` class has a private method called `readFile`, this method may not be accessed in either the `Novel` or `Textbook` classes.

2. A programmer is designing a program to catalog all books in a library. He plans to have a `Book` class that stores features of each book: `author`, `title`, `isOnShelf`, and so on, with operations like `getAuthor`, `getTitle`, `getShelfInfo`, and `setShelfInfo`. Another class, `LibraryList`, will store an array of `Book` objects. The `LibraryList` class will include operations such as `listAllBooks`, `addBook`, `removeBook`, and `searchForBook`. The programmer plans to implement and test the `Book` class first, before implementing the `LibraryList` class. The programmer's plan to write the `Book` class first is an example of

(A) top-down development.

(B) bottom-up development.

(C) procedural abstraction.

(D) information hiding.

(E) a driver program.

Questions 3–5 refer to the `Card` and `Deck` classes shown below.

```
public class Card
{
    private String mySuit;
    private int myValue;     //0 to 12

    public Card(String suit, int value)
    { /* implementation */ }

    public String getSuit()
    { return mySuit; }

    public int getValue()
    { return myValue; }

    public String toString()
    {
        String faceValue = "";
        if (myValue == 11)
            faceValue = "J";
        else if (myValue == 12)
            faceValue = "Q";
        else if (myValue == 0)
            faceValue = "K";
        else if (myValue == 1)
            faceValue = "A";
        if (myValue >= 2 && myValue <= 10)
            return myValue + " of " + mySuit;
        else
            return faceValue + " of " + mySuit;
    }
}

public class Deck
{
    private Card[] myDeck;
    public final static int NUMCARDS = 52;

    public Deck()
    { ...

    //Simulate shuffling the deck.
    public void shuffle()
    { ...

    //other methods not shown ...
}
```

3. Which of the following represents correct /* *implementation* */ code for the constructor in the Card class?

(A)
```
mySuit = suit;
myValue = value;
```

(B)
```
suit = mySuit;
value = myValue;
```

(C)
```
Card = new Card(mySuit, myValue);
```

(D)
```
Card = new Card(suit, value);
```

(E)
```
mySuit = getSuit();
myValue = getValue();
```

4. Consider this description of the Deck constructor:
A Deck object will be constructed as follows:
myDeck[0]...myDeck[12] will contain the spade suit
myDeck[13]...myDeck[25] will contain the heart suit
myDeck[26]...myDeck[38] will contain the diamond suit
myDeck[39]...myDeck[51] will contain the club suit
In each suit the card values range from 0 to 12. (These are converted to actual card values in the toString method of the Card class.) Here is the constructor for the Deck class:

```
public Deck()
{
    < declaration of the myDeck array >

    for (int i = 0; i < NUMCARDS; i++)
    {
        /* code to insert the spade cards into myDeck */
        /* code to insert the heart cards into myDeck */
        /* code to insert the diamond cards into myDeck */
        /* code to insert the club cards into myDeck */
    }
}
```

Which of the following is a correct replacement for /* *code to insert the heart cards into* myDeck */, so that the specification for the myDeck array is satisfied?

(A)
```
if (i / 13 == 1)
    myDeck[i / 13] = new Card("hearts", i % 13);
```

(B)
```
if (i >= 13 && i <= 25)
    myDeck[i % 13] = new Card("hearts", i % 13);
```

(C)
```
if (i / 13 == 1)
    myDeck[i] = new Card("hearts", i % 13);
```

(D)
```
if (i >= 13 && i <= 25)
    myDeck[i] = new Card("hearts", i / 13);
```

(E)
```
if (i / 13 == 1)
    myDeck[i % 13] = new Card("hearts", i % 13);
```

5. Consider the implementation of a `writeDeck` method that is added to the `Deck` class.

```
//Write the cards in myDeck, one per line.
public void writeDeck()
{
    /* implementation code */
}
```

Which of the following is correct `/* implementation code */`?

I
```
System.out.println(myDeck);
```

II
```
for (Card card : myDeck)
    System.out.println(card);
```

III
```
for (Card card : myDeck)
    System.out.println((String) card);
```

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

Refer to the following class for Questions 6 and 7.

```
public class Tester
{
    private int[] testArray = {3, 4, 5};

    //Add 1 to n.
    public void increment (int n)
    { n++; }

    public void firstTestMethod()
    {
        for (int i = 0; i < testArray.length; i++)
        {
            increment(testArray[i]);
            System.out.print(testArray[i] + " ");
        }
    }

    public void secondTestMethod()
    {
        for (int element : testArray)
        {
            increment(element);
            System.out.print(element + " ");
        }
    }
}
```

6. What output will be produced by invoking `firstTestMethod` for a `Tester` object?
   (A) `3 4 5`
   (B) `4 5 6`
   (C) `5 6 7`
   (D) `0 0 0`
   (E) No output will be produced. An `ArrayIndexOutOfBoundsException` will be thrown.

7. What output will be produced by invoking `secondTestMethod` for a `Tester` object, assuming that `testArray` contains `3,4,5`?
   (A) `3 4 5`
   (B) `4 5 6`
   (C) `5 6 7`
   (D) `0 0 0`
   (E) No output will be produced. An `ArrayIndexOutOfBoundsException` will be thrown.

8. Consider the following loop, where n is some positive integer.

```
for (int i = 0; i < n; i += 2)
{
    if (/* test */)
        /* perform some action */
}
```

In terms of n, which Java expression represents the maximum number of times that
/ perform some action */* could be executed?

(A) `n / 2`
(B) `(n + 1) / 2`
(C) `n`
(D) `n - 1`
(E) `(n - 1) / 2`

9. A method is to be written to search an array for a value that is larger than a given item and return its index. The problem specification does not indicate what should be returned if there are several such values in the array. Which of the following actions would be best?

(A) The method should be written on the assumption that there is only one value in the array that is larger than the given item.
(B) The method should be written so as to return the index of every occurrence of a larger value.
(C) The specification should be modified to indicate what should be done if there is more than one index of larger values.
(D) The method should be written to output a message if more than one larger value is found.
(E) The method should be written to delete all subsequent larger items after a suitable index is returned.

10. When will method `whatIsIt` cause a stack overflow (i.e., cause computer memory to be exhausted)?

```
public static int whatIsIt(int x, int y)
{
    if (x > y)
        return x * y;
    else
        return whatIsIt(x - 1, y);
}
```

(A) Only when $x < y$
(B) Only when $x \leq y$
(C) Only when $x > y$
(D) For all values of x and y
(E) The method will never cause a stack overflow.

11. The boolean expression `a[i] == max || !(max != a[i])` can be simplified to
(A) `a[i] == max`
(B) `a[i] != max`
(C) `a[i] < max || a[i] > max`
(D) `true`
(E) `false`

12. Suppose the characters $0, 1, \ldots, 8, 9, A, B, C, D, E, F$ are used to represent a hexadecimal (base-16) number. Here $A = 10$, $B = 11, \ldots, F = 15$. What is the largest base-10 integer that can be represented with a two-digit hexadecimal number, such as 14 or 3A?
(A) 32
(B) 225
(C) 255
(D) 256
(E) 272

13. Consider a `Clown` class that has a default constructor. Suppose a list `ArrayList<Clown> list` is initialized. Which of the following will *not* cause an `IndexOutOfBoundsException` to be thrown?

(A)
```
for (int i = 0; i <= list.size(); i++)
    list.set(i, new Clown());
```

(B)
```
list.add(list.size(), new Clown());
```

(C)
```
Clown c = list.get(list.size());
```

(D)
```
Clown c = list.remove(list.size());
```

(E)
```
list.add(-1, new Clown());
```

Questions 14–16 refer to the `Point`, `Quadrilateral`, and `Rectangle` classes below:

```
public class Point
{
    private int xCoord;
    private int yCoord;

    //constructor
    public Point(int x, int y)
    {
        ...
    }

    //accessors

    public int get_x()
    {
        ...
    }

    public int get_y()
    {
        ...
    }

    //other methods not shown ...

}

public abstract class Quadrilateral
{
    private String myLabels;    //e.g., "ABCD"

    //constructor
    public Quadrilateral(String labels)
    { myLabels = labels; }

    public String getLabels()
    { return myLabels; }

    public abstract int perimeter();
    public abstract int area();
}
```

```
public class Rectangle extends Quadrilateral
{
    private Point myTopLeft;   //coords of top left corner
    private Point myBotRight;  //coords of bottom right corner

    //constructor
    public Rectangle(String labels, Point topLeft, Point botRight)
    { /* implementation code */ }

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }

    //other methods not shown ...
}
```

14. Which statement about the `Quadrilateral` class is *false*?
    (A) The `perimeter` and `area` methods are abstract because there's no suitable default code for them.
    (B) The `getLabels` method is not abstract because any subclasses of `Quadrilateral` will have the same code for this method.
    (C) If the `Quadrilateral` class is used in a program, it *must* be used as a superclass for at least one other class.
    (D) No instances of a `Quadrilateral` object can be created in a program.
    (E) Any subclasses of the `Quadrilateral` class *must* provide implementation code for the `perimeter` and `area` methods.

15. Which represents correct `/* implementation code */` for the `Rectangle` constructor?

    I
    ```
    super(labels);
    ```

    II
    ```
    super(labels, topLeft, botRight);
    ```

    III
    ```
    super(labels);
    myTopLeft = topLeft;
    myBotRight = botRight;
    ```

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

16. Refer to the `Parallelogram` and `Square` classes below.

```
public class Parallelogram extends Quadrilateral
{
    //private instance variables and constructor not shown
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}

public class Square extends Rectangle
{
    //private instance variables and constructor not shown
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}
```

Consider an `ArrayList<Quadrilateral>` `quadList` whose elements are of type `Rectangle`, `Parallelogram`, or `Square`.
Refer to the following method, `writeAreas`:

```
/* Precondition:  quadList contains Rectangle, Parallelogram, or
 *                Square objects in an unspecified order. */
public static void writeAreas(ArrayList quadList)
{
    for (Quadrilateral quad : quadList)
        System.out.println("Area of " + quad.getLabels()
            + " is " + quad.area());
}
```

What is the effect of executing this method?

(A) The area of each `Quadrilateral` in `quadList` will be printed.
(B) A compile-time error will occur, stating that there is no `area` method in abstract class `Quadrilateral`.
(C) A compile-time error will occur, stating that there is no `getLabels` method in classes `Rectangle`, `Parallelogram`, or `Square`.
(D) A `NullPointerException` will be thrown.
(E) A `ClassCastException` will be thrown.

17. Refer to the doSomething method:

```
// postcondition
public static void doSomething(ArrayList<SomeType> list, int i, int j)
{
    SomeType temp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, temp);
}
```

Which best describes the *postcondition* for doSomething?

(A) Removes from list the objects indexed at i and j.
(B) Replaces in list the object indexed at i with the object indexed at j.
(C) Replaces in list the object indexed at j with the object indexed at i.
(D) Replaces in list the objects indexed at i and j with temp.
(E) Interchanges in list the objects indexed at i and j.

18. Consider the NegativeReal class below, which defines a negative real number object.

```
public class NegativeReal
{
    private Double myNegReal;

    //constructor. Creates a NegativeReal object whose value is num.
    //Precondition: num < 0.
    public NegativeReal(double num)
    { /* implementation not shown */ }

    //Postcondition: Returns the value of this NegativeReal.
    public double getValue()
    { /* implementation not shown */ }

    //Postcondition: Returns this NegativeReal rounded to the nearest integer.
    public int getRounded()
    { /* implementation */ }
}
```

Here are some rounding examples:

Negative real number	Rounded to nearest integer
−3.5	−4
−8.97	−9
−5.0	−5
−2.487	−2
−0.2	0

Which /* *implementation* */ of getRounded produces the desired postcondition?

(A) `return (int) (getValue() - 0.5);`
(B) `return (int) (getValue() + 0.5);`
(C) `return (int) getValue();`
(D) `return (double) (getValue() - 0.5);`
(E) `return (double) getValue();`

19. Consider the following method.

```
public static void whatsIt(int n)
{
    if (n > 10)
        whatsIt(n / 10);
    System.out.print(n % 10);
}
```

What will be output as a result of the method call `whatsIt(347)`?

(A) 74
(B) 47
(C) 734
(D) 743
(E) 347

20. A large list of numbers is to be sorted into ascending order. Assuming that a "data movement" is a swap or reassignment of an element, which of the following is a *true* statement?

(A) If the array is initially sorted in descending order, then insertion sort will be more efficient than selection sort.
(B) The number of comparisons for selection sort is independent of the initial arrangement of elements.
(C) The number of comparisons for insertion sort is independent of the initial arrangement of elements.
(D) The number of data movements in selection sort depends on the initial arrangement of elements.
(E) The number of data movements in insertion sort is independent of the initial arrangement of elements.

21. Refer to the definitions of `ClassOne` and `ClassTwo` below.

```
public class ClassOne
{
    public void methodOne()
    {
        ...
    }

    //other methods not shown
}

public class ClassTwo extends ClassOne
{
    public void methodTwo()
    {
        ...
    }

    //other methods not shown
}
```

Consider the following declarations in a client class. You may assume that `ClassOne` and `ClassTwo` have default constructors.

```
ClassOne c1 = new ClassOne();
ClassOne c2 = new ClassTwo();
```

Which of the following method calls will cause an error?

I `c1.methodTwo();`

II `c2.methodTwo();`

III `c2.methodOne();`

(A) None
(B) I only
(C) II only
(D) III only
(E) I and II only

22. Consider the code segment

```
if (n == 1)
    k++;
else if (n == 4)
    k += 4;
```

Suppose that the given segment is rewritten in the form

```
if (/* condition */)
    /* assignment statement */;
```

Given that `n` and `k` are integers and that the rewritten code performs the same task as the original code, which of the following could be used as (1) `/* condition */` and (2) `/* assignment statement */`?

(A) (1) `n == 1 && n == 4` (2) `k += n`

(B) (1) `n == 1 && n == 4` (2) `k += 4`

(C) (1) `n == 1 || n == 4` (2) `k += 4`

(D) (1) `n == 1 || n == 4` (2) `k += n`

(E) (1) `n == 1 || n == 4` (2) `k = n - k`

23. Which of the following will execute *without* throwing an exception?

I
```
String s = null;
String t = "";
if (s.equals(t))
    System.out.println("empty strings?");
```

II
```
String s = "holy";
String t = "moly";
if (s.equals(t))
    System.out.println("holy moly!");
```

III
```
String s = "holy";
String t = s.substring(4);
System.out.println(s + t);
```

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

24. Three numbers a, b, and c are said to be a *Pythagorean Triple* if and only if the sum of the squares of two of the numbers equals the square of the third. A programmer writes a method `isPythTriple` to test if its three parameters form a Pythagorean Triple:

```
//Returns true if a * a + b * b == c * c; otherwise returns false.
public static boolean isPythTriple(double a, double b, double c)
{
    double d = Math.sqrt(a * a + b * b);
    return d == c;
}
```

When the method was tested with known Pythagorean Triples, `isPythTriple` sometimes erroneously returned `false`. What was the most likely cause of the error?

(A) Round-off error was caused by calculations with floating-point numbers.
(B) Type `boolean` was not recognized by an obsolete version of Java.
(C) An overflow error was caused by entering numbers that were too large.
(D) `c` and `d` should have been cast to integers before testing for equality.
(E) Bad test data were selected.

25. Refer to the following class, containing the `mystery` method.

```
public class SomeClass
{
    private int[] arr;

    //Constructor. Initializes arr to contain nonnegative
    // integers k such that 0 <= k <= 9.
    public SomeClass()
    { /* implementation not shown */ }

    public int mystery()
    {
        int value = arr[0];
        for (int i = 1; i < arr.length; i++)
            value = value * 10 + arr[i];
        return value;
    }
}
```

Which best describes what the `mystery` method does?

(A) It sums the elements of `arr`.
(B) It sums the products `10*arr[0]+10*arr[1]+···+10*arr[arr.length-1]`.
(C) It builds an integer of the form $d_1d_2d_3\ldots d_n$, where $d_1 =$ `arr[0]`, $d_2 =$ `arr[1]`, ..., $d_n =$ `arr[arr.length-1]`.
(D) It builds an integer of the form $d_1d_2d_3\ldots d_n$, where $d_1 =$ `arr[arr.length-1]`, $d_2 =$ `arr[arr.length-2]`, ..., $d_n =$ `arr[0]`.
(E) It converts the elements of `arr` to base-10.

Questions 26 and 27 refer to the `search` method in the `Searcher` class below.

```
public class Searcher
{
    private int[] arr;

    //Constructor. Initializes arr with integers.
    public Searcher()
    { /* implementation not shown */ }

    /* Precondition:  arr[first]...arr[last] sorted in ascending order.
     * Postcondition: Returns index of key in arr. If key not in arr,
     *                returns -1. */
    public int search(int first, int last, int key)
    {
        int mid;
        while (first <= last)
        {
            mid = (first + last) / 2;
            if (arr[mid] == key)      //found key, exit search
                return mid;
            else if (arr[mid] < key)  //key to right of arr[mid]
                first = mid + 1;
            else                      //key to left of arr[mid]
                last = mid - 1;
        }
        return -1;                    //key not in list
    }
}
```

26. Which assertion is true just before each execution of the `while` loop?
    (A) `arr[first]` $<$ `key` $<$ `arr[last]`
    (B) `arr[first]` $\leq$ `key` $\leq$ `arr[last]`
    (C) `arr[first]` $<$ `key` $<$ `arr[last]` or `key` is not in `arr`
    (D) `arr[first]` $\leq$ `key` $\leq$ `arr[last]` or `key` is not in `arr`
    (E) `key` $\leq$ `arr[first]` or `key` $\geq$ `arr[last]` or `key` is not in `arr`

27. Consider the array `a` with values as shown:

    4, 7, 19, 25, 36, 37, 50, 100, 101, 205, 220, 271, 306, 321

    where 4 is `a[0]` and 321 is `a[13]`. Suppose that the `search` method is called with `first` $= 0$ and `last` $= 13$ to locate the `key` 205. How many iterations of the `while` loop must be made in order to locate it?
    (A) 3
    (B) 4
    (C) 5
    (D) 10
    (E) 13

28. Consider the following `RandomList` class.

```
public class RandomList
{
    private int[] myList;

    //constructor
    public RandomList()
    { myList = getList(); }

    /* Read random Integers from 0 to 100 inclusive into array list. */
    public int[] getList()
    {
        System.out.println("How many integers? ");
        int listLength = IO.readInt();      //read user input
        int[] list = int[listLength];
        for (int i = 0; i < listLength; i++)
        {
            /* code to add integer to list */
        }
        return list;
    }

    /* Print all elements of this list. */
    public void printList()
    { ...
}
```

Which represents correct `/* code to add integer to list */`?

(A) `list[i] = (int) (Math.random() * 101);`

(B) `list.add((int) (Math.random() * 101));`

(C) `list[i] = (int) (Math.random() * 100);`

(D) `list.add(new Integer(Math.random() * 100))`

(E) `list[i] = (int) (Math.random() * 100) + 1;`

Questions 29 and 30 refer to method `insert` described here. The `insert` method has two string parameters and one integer parameter. The method returns the string obtained by inserting the second string into the first starting at the position indicated by the integer parameter. For example, if `str1` contains `xy` and `str2` contains `cat`, then

```
insert(str1, str2, 0)   returns   catxy
insert(str1, str2, 1)   returns   xcaty
insert(str1, str2, 2)   returns   xycat
```

Here is the header for method `insert`.

```
//Precondition:  0 <= pos <= str1.length().
//Postcondition: Returns /* somestring */.
public static String insert(String str1, String str2, int pos);
```

29. If `str1` $= a_0 a_1 \ldots a_{n-1}$ and `str2` $= b_0 b_1 \ldots b_{m-1}$, which of the following is a correct replacement for `/* somestring */`?
    (A) $a_0 a_1 \ldots a_{pos} b_0 b_1 \ldots b_{m-1} a_{pos+1} a_{pos+2} \ldots a_{n-1}$
    (B) $a_0 a_1 \ldots a_{pos+1} b_0 b_1 \ldots b_{m-1} a_{pos+2} a_{pos+3} \ldots a_{n-1}$
    (C) $a_0 a_1 \ldots a_{pos-1} b_0 b_1 \ldots b_{m-1} a_{pos} a_{pos+1} \ldots a_{n-1}$
    (D) $a_0 a_1 \ldots a_{n-1} b_0 b_1 \ldots b_{m-1}$
    (E) $a_0 a_1 \ldots a_{pos-1} b_0 b_1 \ldots b_{pos-1} a_{pos} a_{pos+1} \ldots a_{n-1}$

30. Method `insert` follows:

```
//Postcondition: Returns /* somestring */.
public static String insert(String str1, String str2, int pos)
{
    String first, last;
        /* more code */
    return first + str2 + last;
}
```

Which of the following is a correct replacement for `/* more code */`?

(A)
```
first = str1.substring(0, pos);
last = str1.substring(pos);
```

(B)
```
first = str1.substring(0, pos - 1);
last = str1.substring(pos);
```

(C)
```
first = str1.substring(0, pos + 1);
last = str1.substring(pos + 1);
```

(D)
```
first = str1.substring(0, pos);
last = str1.substring(pos + 1, str1.length());
```

(E)
```
first = str1.substring(0, pos);
last = str1.substring(pos, str1.length() + 1);
```

Use the following program description for Questions 31–33.

A programmer plans to write a program that simulates a small bingo game (no more than six players). Each player will have a bingo card with 20 numbers from 0 to 90 (no duplicates). Someone will call out numbers one at a time, and each player will cross out a number on his card as it is called. The first player with all the numbers crossed out is the winner. In the simulation, as the game is in progress, each player's card is displayed on the screen.

The programmer envisions a short driver class whose `main` method has just two statements:

```
BingoGame b = new BingoGame();
b.playBingo();
```

The `BingoGame` class will have several objects: a `Display`, a `Caller`, and a `PlayerGroup`. The `PlayerGroup` will have a list of `Players`, and each `Player` will have a `BingoCard`.

31. The relationship between the `PlayerGroup` and `Player` classes is an example of
    (A) an interface.
    (B) encapsulation.
    (C) composition.
    (D) inheritance.
    (E) independent classes.

32. Which is a reasonable data structure for a `BingoCard` object? Recall that a `BingoCard` has 20 integers from 0 to 90, with no duplicates. There should also be mechanisms for crossing off numbers that are called, and for detecting a winning card (i.e., one where all the numbers have been crossed off).

I
```
int[] myBingoCard;   //will contain 20 integers
                     //myBingoCard[k] is crossed off by setting it to -1.
int numCrossedOff;   //player wins when numCrossedOff reaches 20.
```

II
```
boolean[] myBingoCard;  //will contain 91 boolean values, of which
                        //20 are true. All the other values are false.
                        //Thus, if myBingoCard[k] is true, then k is
                        //on the card, 0 <= k <= 90. A number k is
                        //crossed off by changing the value of
                        //myBingoCard[k] to false.
int numCrossedOff;  //player wins when numCrossedOff reaches 20.
```

III
```
ArrayList<Integer> myBingoCard;  //will contain 20 integers.
      //A number is crossed off by removing it from the ArrayList.
      //Player wins when myBingoCard.size() == 0.
```

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

33. The programmer decides to use an `ArrayList<Integer>` to store the numbers to be called by the `Caller`:

```
public class Caller
{
    private ArrayList<Integer> myNumbers;

    //constructor
    public Caller()
    {
        myNumbers = getList();
        shuffleNumbers();
    }

    //Return the numbers 0...90 in order.
    private ArrayList<Integer> getList()
    { /* implementation not shown */ }

    //Shuffle the numbers.
    private void shuffleNumbers()
    { /* implementation not shown */ }
}
```

When the programmer tests the constructor of the `Caller` class she gets a `NullPointerException`. Which could be the cause of this error?

(A) The `Caller` object in the driver class was not created with `new`.

(B) The programmer forgot the `return` statement in `getList` that returns the list of `Integers`.

(C) The declaration of `myNumbers` is incorrect. It needed to be

```
private ArrayList<Integer> myNumbers = null;
```

(D) In the `getList` method, an attempt was made to add an `Integer` to an `ArrayList` that had not been created with `new`.

(E) The `shuffleNumbers` algorithm went out of range, causing a `null Integer` to be shuffled into the `ArrayList`.

Questions 34–40 involve reasoning about the code from the GridWorld Case Study. A Quick Reference to the case study is provided as part of this exam. The actors in GridWorld are represented in this book with the pictures shown below. Each actor is shown facing north. These pictures almost certainly will be different from those used on the AP exam!

34. Which is a *false* statement about `Bug` movement?
    (A) A `Bug` can move on a diagonal line across the grid.
    (B) If a `Flower` is directly in front of a `Bug`, the `Bug` will replace the `Flower` in the `Flower`'s location.
    (C) If a `Rock` is directly in front of a `Bug`, the `Bug` will not change its location.
    (D) If a `Bug` is at the edge of the grid, the `Bug` will change its direction to `Location.RIGHT` + its current direction.
    (E) It is possible for a `Bug` to turn through a complete circle without changing its location.

35. Consider an actor whose current direction is `Location.NORTHWEST`. The following method call is made for this actor:

```
setDirection(getDirection() + Location.HALF_LEFT);
```

What is the `int` value of the actor's resulting direction?
(A) 0
(B) 45
(C) 90
(D) 225
(E) 270

36. Suppose `Bug` and `BoxBug` behavior will be modified to allow bugs and box bugs to move onto `Rocks` in the same way that they move onto `Flowers`. Which classes will need to be modified to effect this change?

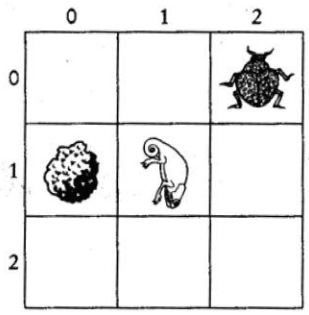
I `Actor`

II `Bug`

III `BoxBug`

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

37. Consider the small bounded grid in the diagram.



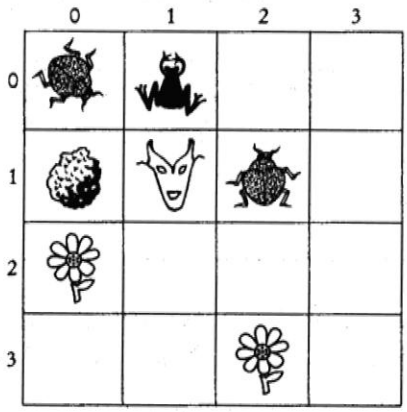
It shows a red `Bug` facing north at (0, 2); a black `Rock` at (1, 0); and a blue `ChameleonCritter` facing south at (1, 1). After one step of the simulation, the bug is facing northeast at location (0, 2), and the rock is still at location (1, 0). Which represents a legitimate state for the `ChameleonCritter`?

(A) Color is blue, location is (2, 1), and direction is south.
(B) Color is black, location is (1, 1), and direction is east.
(C) Color is red, location is (1, 2), and direction is northeast.
(D) Color is red, location is (0, 0), and direction is northwest.
(E) Color is black, location is (2, 0), and direction is south.

38. The `moveTo` method of the `Actor` class would *not* be suitable for which of the following operations? You may assume that each scenario below is contained in the context of a two-dimensional grid.

(A) Capturing a piece in Chess (namely, removing that piece from the board and taking its place on the board).
(B) Moving a piece on a Checkers board into an empty location.
(C) Rearranging the furniture in a room that has a rectangular floor plan.
(D) Changing a reserved seat to another location in a theater that has a rectangular arrangement of seats.
(E) Placing two monkeys in the same cage at a zoo, where the zoo has a rectangular arrangement of cages.

39. Consider the bounded grid shown, and the `Actor` in location (1, 1).



If it is the `Actor`'s turn to act, which are valid possibilities for a new location, if the `Actor` is a

(1) `Critter`

(2) `ChameleonCritter`

(A) (1) (0, 0), (2, 0), (2, 1), (2, 2), (1, 2), (0, 2)
(2) (2, 1), (2, 2), (0, 2)

(B) (1) (0, 0), (2, 0), (1, 2)
(2) (3, 0), (3, 1), (2, 1), (2, 2), (3, 3), (2, 3), (1, 3), (0, 2), (0, 3)

(C) (1) (0, 0), (0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1), (2, 2)
(2) (2, 1), (2, 2), (0, 2)

(D) (1) (0, 0), (0, 1), (1, 1), (1, 2), (2, 0), (3, 2)
(2) (3, 0), (3, 1), (2, 1), (2, 2), (3, 3), (2, 3), (1, 3), (0, 2), (0, 3)

(E) (1) (0, 0), (0, 1), (0, 2), (1, 2), (2, 0), (2, 1), (2, 2)
(2) (0, 2), (2, 0), (2, 1), (2, 2)

40. Suppose the program is changed so that a `Critter` is allowed to age. A `Critter` will start out at age 1, and have its age incremented by 1 each time it acts. To make this change, a private instance variable `age` is added to the `Critter` class, as well as an accessor method, `getAge`. A constructor is provided that initializes `age` to 1. What other changes must be made?

I The `act` method of the `Actor` class must be modified.

II The `act` method of the `Critter` class must be modified.

III The `act` method of the `ChameleonCritter` class must be overridden.

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