



*Ninth Edition*

Be Prepared  
for the  
**AP**  
Computer Science  
Exam in Java

Chapter 5: Annotated Solutions  
to Past Free-Response Questions

**2013**

**Maria Litvin**

Phillips Academy, Andover, Massachusetts

**Gary Litvin**

Skylight Publishing, Andover, Massachusetts

Skylight Publishing  
Andover, Massachusetts

**Copyright © 2026 by  
Maria Litvin, Gary Litvin, and Skylight Publishing**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the authors and Skylight Publishing.

ISBN 978-0-9972528-3-5

Skylight Publishing  
9 Bartlet Street, Suite 70  
Andover, MA 01810

web: [www.skylit.com](http://www.skylit.com)  
e-mail: [sales@skylit.com](mailto:sales@skylit.com)  
[support@skylit.com](mailto:support@skylit.com)

The free-response questions for this exam are posted on [apstudent.collegeboard.org](http://apstudent.collegeboard.org) and, for teachers, on AP Central:

- For students: [apstudent.collegeboard.org](http://apstudent.collegeboard.org)
- For teachers: [apcentral.collegeboard.org/courses](http://apcentral.collegeboard.org/courses)

Scoring guidelines are usually posted over the summer.

The [www.skylit.com/beprepared/x2013all.zip](http://www.skylit.com/beprepared/x2013all.zip) file contains complete Java classes that include solutions and test programs for runnable projects.

## Question 1

### Part (a)

```
public DownloadInfo getDownloadInfo(String title)
{
    for (DownloadInfo di : downloadList)
        if (di.getTitle().equals(title))
            return di;
    return null;
}
```

### Part (b)

```
public void updateDownloads(List<String> titles)
{
    for (String title : titles)
    {
        DownloadInfo di = getDownloadInfo(title);
        if (di == null)
            downloadList.add(new DownloadInfo(title)); 1
        else
            di.incrementTimesDownloaded();
    }
}
```

### Notes:

1. Notice that the comment in `DownloadInfo`'s constructor states that it “sets the number of times downloaded to 1” — no need to increment it.

## Question 2

### Part (a)

```
public TokenPass(int playerCount)
{
    board = new int[playerCount];
    for (int i = 0; i < playerCount; i++)
        board[i] = (int)(10 * Math.random()) + 1;
    currentPlayer = (int)(playerCount * Math.random());
}
```

### Part (b)

```
public void distributeCurrentPlayerTokens()
{
    int numTokens = board[currentPlayer];
    board[currentPlayer] = 0;
    int i = currentPlayer;
    while (numTokens > 0)
    {
        i = (i+1) % board.length; 1
        board[i]++;
        numTokens--;
    }
}
```

### Notes:

1. Or:

```
i++;
if (i == board.length)
    i = 0;
```

### Question 3

#### Part (a)

```
public static ArrayList<Location> getEmptyLocations(Grid grid)
{
    ArrayList<Location> emptyLocs = new ArrayList<Location>();

    for (int r = 0; r < grid.getNumRows(); r++)
    {
        for (int c = 0; c < grid.getNumCols(); c++)
        {
            Location loc = new Location(r, c);
            if (grid.get(loc) == null)
                emptyLocs.add(loc);
        }
    }
    return emptyLocs;
}
```

#### Part (b)

```
public class JumpingCritter extends Critter
{
    public ArrayList<Location> getMoveLocations()
    {
        return GridWorldUtilities.getEmptyLocations(getGrid());
    }

    public Location selectMoveLocation(ArrayList<Location> locs)
    {
        if (locs.size() == 0)
            return null; 1
        return super.selectMoveLocation(locs);
    } 2
}
```

#### Notes:

1. You cannot remove this `JumpingCritter` from the grid right here, because this would violate `selectMoveLocation`'s precondition #2. Leave this job to the `makeMove` method.

2. Or:

```
public Location selectMoveLocation(ArrayList<Location> locs)
{
    int n = locs.size();
    if (n == 0)
        return null;
    int r = (int)(Math.random() * n);
    return locs.get(r);
}
```

## Question 4

### Part (a)

```

public SkyView(int numRows, int numCols, double[] scanned) 1
{
    view = new double[numRows][numCols];
    int row = 0, col = 0, step = 1;
    for (double amtLight : scanned)
    {
        view[row][col] = amtLight;
        int nextCol = col + step;
        if (nextCol >= 0 && nextCol < numCols)
            col = nextCol;
        else
        {
            row++;
            step = -step;
        }
    } 2
}

```

### Notes:

1. There is a typo in the statement of this question (Page 17): `SkyView(4, 3, values)` and `SkyView(3, 2, values)` probably was meant to be `SkyView(4, 3, scanned)` and `SkyView(3, 2, scanned)`. Thanks to Doug Vermes for mentioning this to us.
2. There are many different acceptable solutions to this part of the question — too many to list here. For example, you can get rid of the `step` variable and use `1 - 2*(row % 2)` instead:

```

view = new double[numRows][numCols];
int row = 0, col = 0;
for (double amtLight : scanned)
{
    view[row][col] = amtLight;
    int nextCol = col + 1 - 2*(row % 2);
    if (nextCol >= 0 && nextCol < numCols)
        col = nextCol;
    else
        row++;
}

```

Or use an `if` statement:

```

view = new double[numRows][numCols];
int row = 0, col = 0;
for (double amtLight : scanned)
{
    view[row][col] = amtLight;
    int nextCol = col;
    if (row % 2 == 0)

```

```

        nextCol++;
    else
        nextCol--;
    if (nextCol >= 0 && nextCol < numCols)
        col = nextCol;
    else
        row++;
}

```

Or fill the even rows and the odd rows in separate loops. Or for each element `view[row][col]` calculate the location of the corresponding element `scanned[i]`:

```

view = new double[numRows][numCols];
for (int row = 0; row < numRows; row++)
{
    for (int col = 0; col < numCols; col++)
    {
        int i = row * numCols;
        if (row % 2 == 0)
            i += col;
        else
            i += numCols - col - 1;
        view[row][col] = scanned[i];
    }
}

```

### Part (b)

```

public double getAverage(int startRow, int endRow,
                        int startCol, int endCol)
{
    double sum = 0.0;
    for (int r = startRow; r <= endRow; r++)
        for (int c = startCol; c <= endCol; c++)
            sum += view[r][c];
    return sum / ((endRow - startRow + 1) * (endCol - startCol + 1));
}

```

### Notes:

1. Or count the number of elements in the region:

```

double sum = 0.0;
int count = 0;
for (int r = startRow; r <= endRow; r++)
{
    for (int c = startCol; c <= endCol; c++)
    {
        sum += view[r][c];
        count++;
    }
}
return sum / count;

```