



Ninth Edition

Be Prepared
for the
AP
Computer Science
Exam in Java

Chapter 5: Annotated Solutions
to Past Free-Response Questions

2024

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
14 Lincoln Street
Andover, MA 01810

web: www.skylit.com
e-mail: sales@skylit.com
support@skylit.com

www.skylit.com/beprepared/x2024all.zip contains complete Java code, including solutions and test programs for runnable projects.

The free-response questions for this exam are posted on AP Central:

apcentral.collegeboard.org

Scoring guidelines for teachers are usually posted over the summer.

Question 1

Part (a)

```
public void simulateOneDay(int numBirds)
{
    boolean bear = Math.random() < 0.05;
    if (!bear)
    {
        int standardFood = 10 + (int)(41*Math.random());1
        currentFood -= numBirds * standardFood;
        if (currentFood < 0)2
        {
            currentFood = 0;
        }
    }
    else
    {
        currentFood = 0;
    }
}
```

Notes:

1. Recall that `Math.random()` returns a double value greater than or equal to 0.0 and less than 1.0, so `(int)(41*Math.random())` will be from 0 to 40, inclusive of both ends.
2. Don't forget that `currentFood` cannot be negative.

Part (b)

```
public int simulateManyDays(int numBirds, int numDays)
{
    int days = 0;

    while (currentFood > 0 && days < numDays)
    {
        days++;
        simulateOneDay(numBirds);
    }
    return days;
}1
```

Notes:

1. This solution counts any day when there was some food in the feeder at the start of the day, even if there was not enough to feed all the birds.

Question 2

```
public class Scoreboard
{
    private String team1, team2, activeTeam;
    private int score1, score2;

    public Scoreboard(String name1, String name2) {
        team1 = name1;
        team2 = name2;
        activeTeam = team1;
        score1 = 0;1
        score2 = 0;1
    }

    public void recordPlay(int points)
    {
        if (activeTeam.equals(team1))
        {
            if (points > 0)
                score1 += points;
            else
                activeTeam = team2;
        }
        else
        {
            if (points > 0)
                score2 += points;
            else
                activeTeam = team1;
        }
    }

    public String getScore()
    {
        return score1 + "-" + score2 + "-" + activeTeam;
    }
}2
```

Notes:

1. `int` instance variables are initialized to 0 by default, so these statements are optional.
2. The question begs for a solution that uses two-element parallel arrays for the team names and scores, with the active team indicated by the index 0 or 1. Although FRQ #2 doesn't expect the use of arrays, such solutions, if implemented correctly, will receive full credit. For example:

```
public class Scoreboard
{
    private String[] teams = new String[2];
    private int[] scores = new int[2];
    private int activeTeam;

    public Scoreboard(String name1, String name2)
    {
        teams[0] = name1;
        teams[1] = name2;
        activeTeam = 0;
    }

    public void recordPlay(int points)
    {
        if (points == 0)
            activeTeam = 1 - activeTeam; // toggle between 0 and 1
        else
            scores[activeTeam] += points;
    }

    public String getScore()
    {
        return scores[0] + "-" + scores[1] + "-" +
                teams[activeTeam];
    }
}
```

Question 3

Part (a)

```
public boolean isWordChain()
{
    for (int i = 1; i < wordList.size(); i++)
    {
        String currentWord = wordList.get(i);
        String previousWord = wordList.get(i-1);

        if (currentWord.indexOf(previousWord) == -1)
            return false;
    }
    return true;
}
```

Part (b)

```
public ArrayList<String> createList(String target)
{
    ArrayList<String> targetList = new ArrayList<String>();

    for (String word : wordList)
    {
        if (word.indexOf(target) == 0)1
        {
            String ending = word.substring(target.length());
            targetList.add(ending);
        }
    }

    return targetList;
}
```

Notes:

1. Or: `if (word.startsWith(target))` -- not in the AP subset, but will receive full credit.

Question 4

Part (a)

```
public Location getNextLoc(int row, int col)
{
    if (row < grid.length - 1 && col < grid[0].length - 1)
    {
        if (grid[row+1][col] < grid[row][col+1])
            return new Location(row+1, col);
        else
            return new Location(row, col+1);
    }
    else if (row == grid.length - 1)
    {
        return new Location(row, col+1);
    }
    else
    {
        return new Location(row+1, col);
    }
}¹
```

Notes:

1. Alternative solution:

```
public Location getNextLoc(int row, int col)
{
    if (row == grid.length - 1)
        return new Location(row, col+1);
    else if (col == grid[0].length - 1)
        return new Location(row+1, col);
    else if (grid[row+1][col] < grid[row][col+1])
        return new Location(row+1, col);
    else
        return new Location(row, col+1);
}
```

Part (b)

```
public int sumPath(int row, int col)
{
    int sum = 0;

    while (true)
    {
        sum += grid[row][col];
        if (row == grid.length - 1 && col == grid[0].length - 1)
            return sum;

        Location loc = getNextLoc(row, col);
        row = loc.getRow();
        col = loc.getCol();
    }
} 1,2,3
```

Notes:

1. Alternative solution:

```
public int sumPath(int row, int col)
{
    int sum = 0;

    while (row < grid.length && col < grid[0].length)
    {
        sum += grid[row][col];
        if (row != grid.length || col != grid[0].length)
        {
            Location loc = getNextLoc(row, col);
            row = loc.getRow();
            col = loc.getCol();
        }
    }

    return sum;
}
```

2. Another solution, perhaps more straightforward, proposed by Lisa Ryder from Westlake High School in Westlake Village, CA:

```
public int sumPath(int row, int col)
{
    int sum = grid[row][col];

    while (row < grid.length - 1 || col < grid[0].length - 1)
    {
        Location next = getNextLoc(row, col);
        row = next.getRow();
        col = next.getCol();
        sum += grid[row][col];
    }

    return sum;
}
```

3. Yong Joo from Alameda High School in Alameda, CA suggested that a solution with recursion is called for. Here is our version:

```
public int sumPath(int row, int col)
{
    int sum = grid[row][col];

    if (row < grid.length - 1 || col < grid[0].length - 1)
    {
        Location next = getNextLoc(row, col);
        sum += sumPath(next.getRow(), next.getCol());
    }

    return sum;
}
```

Students are not expected, of course, to write recursive methods on the AP exam, but, if written correctly, a recursive solution will receive full credit.