



*Ninth Edition*

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

Chapter 5: Annotated Solutions  
to Past Free-Response Questions

**2010**

**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/x2010all.zip](http://www.skylit.com/beprepared/x2010all.zip) file contains complete Java classes that include solutions and test programs for runnable projects.

## Question 1

### Part (a)

```
public int getTotalBoxes()
{
    int total = 0;
    for (CookieOrder o : orders)
        total += o.getNumBoxes();
    return total;
}
```

### Part (b)

```
public int removeVariety(String cookieVar)
{
    int total = 0;
    for (int i = orders.size() - 1; i >= 0; i--) 1
    {
        CookieOrder o = orders.get(i);
        if (o.getVariety().equals(cookieVar)) 2
        {
            total += o.getNumBoxes();
            orders.remove(i);
        }
    }
    return total;
} 3, 4
```

### Notes:

1. The list is traversed in the reverse order, so that the removed elements do not affect the indices of the remaining elements.
2. equals, not ==.
3. Alternative solution with a while loop, traversing the list from the beginning:

```
public int removeVariety(String cookieVar)
{
    int total = 0;
    int i = 0;
    while (i < orders.size())
    {
        CookieOrder o = orders.get(i);
        if (o.getVariety().equals(cookieVar))
        {
            total += o.getNumBoxes();
            orders.remove(i);
        }
    }
}
```

```
        else
            i++;
    }
    return total;
}
```

4. Alternative solution with an iterator (not in the AP subset):

```
public int removeVariety(String cookieVar)
{
    int total = 0;
    Iterator iter = orders.iterator();

    while (iter.hasNext())
    {
        CookieOrder o = iter.next();
        if (o.getVariety().equals(cookieVar))
        {
            total += o.getNumBoxes();
            iter.remove();
        }
    }
    return total;
}
```

## Question 2

```
public class APLine
{
    private int a, b, c;

    public APLine(int a1, int b1, int c1)
    {
        a = a1;
        b = b1;
        c = c1;
    }1

    public double getSlope()
    {
        return -(double)a / b; 2
    }

    public boolean isOnLine(int x, int y)
    {
        return a*x + b*y + c == 0;
    }
}
```

### Notes:

1. Or:

```
public APLine(int a, int b, int c)
{
    this.a = a;
    this.b = b;
    this.c = c;
}
```

2. The cast to `double` on one or both operands is necessary to avoid integer division; casting the result to `double` would be too late.

**Question 3****Part (a)**

```

public boolean isLevelTrailSegment(int start, int end)
{
    int max = markers[start];
    int min = markers[start];

    for (int i = start + 1; i <= end; i++)
    {
        if (markers[i] > max)
            max = markers[i];
        if (markers[i] < min)
            min = markers[i];
    }
    return max - min <= 10;
} 1

```

**Notes:**

## 1. Alternative solution:

```

public boolean isLevelTrailSegment(int start, int end)
{
    for (int i = start; i < end; i++)
        for (int j = i+1; j <= end; j++)
            if (Math.abs(markers[i] - markers[j]) > 10)
                return false;
    return true;
}

```

**Part (b)**

```

public boolean isDifficult()
{
    int count = 0;

    for (int i = 0; i < markers.length - 1; i++)
    {
        if (Math.abs(markers[i+1] - markers[i]) >= 30) 1
            count++;
    }
    return count >= 3;
}

```

**Notes:**

## 1. Or:

```

if (markers[i+1] - markers[i] >= 30 ||
    markers[i] - markers[i+1] >= 30)

```

## Question 4

### Part (a)

```
public Actor actorWithMostNeighbors()
{
    ArrayList<Location> locs = gr.getOccupiedLocations();
    if (locs.size() == 0)
        return null;

    int max = -1;
    Location bestLoc = null;

    for (Location loc : locs)
    {
        int n = gr.getOccupiedAdjacentLocations(loc).size(); 1

        if (n > max)
        {
            max = n;
            bestLoc = loc;
        }
    }

    return gr.get(bestLoc);
}
```

### Notes:

1. Or:

```
int n = gr.getNeighbors(loc).size();
```

### Part (b)

```
public List<Location> getOccupiedWithinTwo(Location loc)
{
    ArrayList<Location> list = new ArrayList<Location>();

    for (int r = loc.getRow() - 2; r <= loc.getRow() + 2; r++)
    {
        for (int c = loc.getCol() - 2; c <= loc.getCol() + 2; c++)
        {
            Location loc1 = new Location(r, c);
            if (gr.isValid(loc1) && gr.get(loc1) != null &&
                !loc1.equals(loc))
                list.add(loc1);
        }
    }

    return list;
} 1
```

**Notes:**

## 1. Alternative solution:

```
public List getOccupiedWithinTwo(Location loc)
{
    ArrayList<Location> list = new ArrayList();
    ArrayList<Location> occupied = gr.getOccupiedLocations();

    for (Location loc1 : occupied)
        if (Math.abs(loc.getRow() - loc1.getRow()) <= 2 &&
            Math.abs(loc.getCol() - loc1.getCol()) <= 2 &&
                !loc1.equals(loc))
            list.add(loc1);

    return list;
}
```