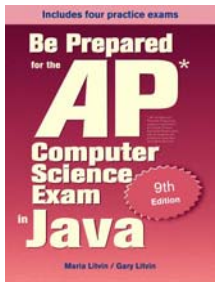

Ninth Edition

Be Prepared
for the
AP
Computer Science Exam in Java

Annotated Solutions to Sample Free-Response Questions in the 2025 Course and Exam Description

Maria Litvin
Phillips Academy, Andover, Massachusetts

Gary Litvin
Skylight Publishing, Andover, Massachusetts

Skylight Publishing
Andover, Massachusetts

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

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

www.skylit.com/beprepared/x2025allCED.zip contains complete Java code, including solutions and test programs for runnable projects.

These free-response questions and scoring rubrics are part of the AP Computer Science A Course and Exam Description effective Fall 2025, Page 161.

Introduction

The College Board has redesigned the AP Computer Science A course and released a new Course and Exam Description (CED), effective Fall 2025. The revised CED maintains a similar structure for the free-response questions, but most questions will be shorter than before.

Question 1: Methods and Control Structures: Part (a) (one method; creating objects, calling methods, using conditionals and loops); Part (b) (one method; calling `String` methods)

Question 2: Class Design (write a complete Java class with one constructor and one method)

Question 3: Data Analysis with `ArrayList` (one method)

Question 4: 2D Array (one method)

Note: No free-response question will require the use of one-dimensional arrays.

Only Question 1 consist of two parts; the remaining questions each have a single part. The free-response section is worth 25 points in total and accounts for 45% of the overall exam score. The point distribution is as follows:

Question 1: 7 points (Part (a) 4 points; Part (b) 3 points)
Question 2: 7 points
Question 3: 5 points
Question 4: 6 points

For each of the four sample free-response questions, the CED includes a "Canonical Solution" and "Scoring Criteria" (scoring rubric). We have supplemented the canonical solutions with notes, added several alternative solutions, and provide runnable files for the solutions and tests.

Question 1: Methods and Control Structures

Part (a)

Canonical solution:

```
public MessageBuilder(String startingWord)
{
  message = startingWord;
  String w = getNextWord(startingWord);
  numWords = 1;
  while(w != null)
  {
    message += " " + w;
    numWords++;
    w = getNextWord(w);
  }
}
```

Alternative solution:

```
public MessageBuilder(String startingWord)
{
  message = "";
  numWords = 0;
  String w = startingWord;
  while (w != null)
  {
    if (message.length() > 0)
      message += " ";
    message += w;
    numWords++;
    w = getNextWord(w);
  }
}
```

Part (b)

Canonical solution:

```
public String getAbbreviation() [1]
{
  String temp = message; [2]
  String result = temp.substring(0, 1);
  while (temp.indexOf(" ") >= 0)
  {
    int i = temp.indexOf(" ");
    temp = temp.substring(i + 1);
    result += temp.substring(0, 1);
  }
  return result;
}
```

Notes:

1. The name of this method might suggest that it is a simple "getter" method. Its functionality could, in theory, be implemented within the `MessageBuilder` constructor by storing the abbreviation in an instance variable, with `getAbbreviation` simply returning that value. However, this approach goes against the intent of the question and would not receive credit — see the rubric for details. (Additionally, `MessageBuilder` could have another constructor that builds the message differently, in which case `getAbbreviation` would no longer work as expected.)
2. We need to create a copy of `message` and work with the copy to ensure that the original message remains unchanged, as required by the postcondition.

Alternative solution — with `split`

```
public String getAbbreviation()
{
  String[] words = message.split(" ");
  String result = "";

  for (String word : words)
  {
    result += word.substring(0, 1);
                  // Or: result += word.charAt(0); [1]
  }
  return result;
} [2]
```

Notes:

1. `charAt` method is outside the AP Java Quick Reference, but may be used. If the solution is implemented correctly, it will earn full credit.
2. Although a solution can be implemented without an array or the `String` class's `split` method — as shown in the Canonical solution above — the use of an array and `split` is allowed and, when implemented correctly, such solution would earn full credit.

Alternative solution — with `indexOf(str, fromIndex)`:

```
public String getAbbreviation()
{
  String result = message.substring(0, 1);
                      // Or: result += word.charAt(0);
  int  i = message.indexOf(" ");

  while (i != -1)
  {
    i++;
    result += message.substring(i, i+1);
                      // Or: result += word.charAt(i);
    i = message.indexOf(" ", i); [1]
  }
  return result;
}
```

Notes:

1. The overloaded version of the `indexOf` method with two parameters is not in the AP Java Quick Reference. However, if used correctly, the code would earn full credit.

Alternative solution — with a helper method and recursion:

```
public String getAbbreviation()
{
  return getAbbreviationHelper(message); [1]
}

public String getAbbreviationHelper(String message)
{
  String result = message.substring(0, 1);
  int i = message.indexOf(" ");

  if (i != -1)
  {
    result += getAbbreviationHelper(message.substring(i+1));
  }
  return result;
}
```

Notes:

1. If implemented correctly, a solution that uses a helper method will earn full credit. However, we recommend that you use recursion only if you are completely confident in your solution and have a lot of experience with recursion; otherwise it's best to avoid using recursion in your solutions to free-response questions.

Question 2

Canonical solution:

```
public class CupcakeMachine
{
  private int availCupcakes;
  private double cupcakeCost;
  private int orderNum;

  public CupcakeMachine(int num, double cost)
  {
    availCupcakes = num;
    cupcakeCost = cost;
    orderNum = 1;
  }

  public String takeOrder(int quantity)
  {
    String message = "Order cannot be filled";
    if (quantity <= availCupcakes)
    {
      availCupcakes -= quantity;
      double cost = quantity * cupcakeCost;
      message = "Order number " + orderNum
                + ", cost $" + cost; [1]
      orderNum++;
    }
    return message;
  }
}
```

Notes:

1. `cost` is a `double`, and it may not be formatted correctly in the returned message with dollars and cents. You might be tempted to correct that, for example:

   ```
   DecimalFormat money = new DecimalFormat("$0.00");
   return "Order number " + orderNum + ", cost " +
                                          money.format(cost);
   ```

   However, this additional feature is not expected and it contradicts the examples shown in the question. In general, it's not advisable to add uncalled for features to your code: you won't earn extra points, and you may lose points if the additional code is incorrect. The AP exam is not the time to show off.

Alternative solution:

```
public class CupcakeMachine
{
  private int cupcakesInMachine;
  private double cupcakeCost;
  private int orderNum;

  public CupcakeMachine(int num, double cost)
  {
    orderNum = 0; // optional: default
    cupcakesInMachine = num;
    cupcakeCost = cost;
  }

  public String takeOrder(int orderQuantity)
  {
    if (orderQuantity <= cupcakesInMachine)
    {
      orderNum++;
      cupcakesInMachine -= orderQuantity;
      double cost = orderQuantity * cupcakeCost;
      return "Order number " + orderNum + ", cost $" + cost;
    }
    else
    {
      return "Order cannot be filled";
    }
  }
}
```

Question 3

Canonical solution:

```
public double averageWithinRange(double lower, double upper)
{
  double sum = 0.0;
  int count = 0;

  for (ItemInfo it : inventory)
  {
    if (it.isAvailable()
        && it.getCost() >= lower && it.getCost() <= upper)
    {
      sum += it.getCost();
      count++;
    }
  }
  return sum / count;
}
```

Question 4

Canonical solution:

```
public int columnWithFewest(String target)
{
  int minCol = 0;
  int minCount = sched.length; [1]
  for (int col = 0; col < sched[0].length; col++)
  {
    int count = 0;
    for (int row = 0; row < sched.length; row++)
    {
      if (sched[row][col].getStatus().equals(target)) [2]
      {
        count++;
      }
    }
    if (count < minCount)
    {
      minCol = col;
      minCount = count;
    }
  }
  return minCol;
}
```

Notes:

1. We need to return the column index, so we need to keep track of both the minimum count and the minimum column index.
2. Use `equals`, not ==, when comparing strings. Whether == works may depend on how an `Appointment` object is created. If the constructor simply stores a string literal and the same literal is used as the value passed to `columnWithFewest`, then == might work. However, if the `Appointment` constructor stores a modified version of the string — for example, using `status + ""` — then == will fail. Always use `equals` when comparing strings in AP exam solutions.