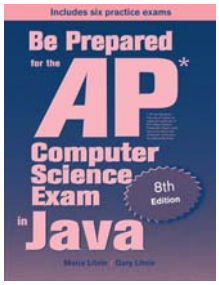

Eighth Edition

Be Prepared
for the
AP
Computer Science Exam in Java

Chapter 6: Annotated Solutions to Past Free-Response Questions

2012

Maria Litvin
Phillips Academy, Andover, Massachusetts

Gary Litvin
Skylight Publishing, Andover, Massachusetts

Skylight Publishing
Andover, Massachusetts

Library of Congress Control Number: 2021950662

ISBN 978-0-9972528-7-3

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

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

The free-response questions for this exam are posted on apstudent.collegeboard.org and, for teachers, on AP Central:

- For students: `apstudent.collegeboard.org`
- For teachers: `apcentral.collegeboard.org/courses`

Scoring guidelines are usually posted over the summer.

The www.skylit.com/beprepared/x2012all.zip file contains complete Java classes that include solutions and test programs for runnable projects.

Question 1

Part (a)

```
public void addClimb(String peakName, int climbTime)
{
  climbList.add(new ClimbInfo(peakName, climbTime));
}
```

Part (b)

```
public void addClimb(String peakName, int climbTime)
{
  int i = 0;
  while (i < climbList.size() &&
         climbList.get(i).getName().compareTo(peakName) <= 0) [1]
    i++; [2]
  climbList.add(i, new ClimbInfo(peakName, climbTime));
}
```

Notes:

1. Compare `i` and `climbList.size()` first to avoid `IndexOutOfBoundsException`.

2. Or, a bit more verbose, using break (which is not in the AP subset but OK):

```
...
while (i < climbList.size())
{
  if (climbList.get(i).getName().compareTo(peakName) <= 0)
    break;
  i++;
}
```

Part (c)

(i) YES (NO)

(ii) (YES) NO

Question 2

```
public class RetroBug extends Bug [1]
{
  private Location prevLocation = null; [2]
  private int prevDirection;

  public RetroBug() [3]
  {
    setColor(Color.RED); [4]
  }

  public RetroBug(Color bugColor)
  {
    setColor(bugColor); [5]
  }

  public void act()
  {
    prevLocation = getLocation();
    prevDirection = getDirection();
    super.act();
  }

  public void restore()
  {
    if (prevLocation == null) [6]
      return;

    Actor neighbor = getGrid().get(prevLocation);
    if (neighbor == null || neighbor instanceof Flower)
      moveTo(prevLocation);

    setDirection(prevDirection);
  }
}
```

Notes:

1. `import` statements are needed for the code to run but not required for full credit.

2. `= null` is optional: it is the default, but the rules for default initialization are not in the AP subset. It won't hurt if you add a redundant statement

   ```
   prevLocation = null;
   ```

   to each constructor (after a call to super, if present).

3. The question of whether any or what constructors are needed to receive full credit will be answered at the AP Exam Reading in June. Good style calls for two constructors for this class, similar to `Bug`'s constructors.

4. Or:

```
super();
```

   or simply empty braces (`super()` is called by default).

5. Or:

```
super(bugColor);
```

6. That is, if `act` has not been called before.

Question 3

Part (a)

```
public int findHorseSpace(String name)
{
  for (int i = 0; i < spaces.length; i++)
    if (spaces[i] != null && spaces[i].getName().equals(name)) [1]
      return i;
  return -1;
}
```

Notes:

1. Check for `null` first to avoid `NullPointerException`.

Part (b)

```
public void consolidate()
{
  Horse[] temp = new Horse[spaces.length]; [1]
  int j = 0;
  for (int i = 0; i < spaces.length; i++)
  {
    if (spaces[i] != null)
    {
      temp[j] = spaces[i];
      j++;
    }
  }
  spaces = temp;
} [2]
```

Notes:

1. The values are initialized to `null` by default.

2. This safe and expedient but inelegant solution that uses a temporary array is acceptable in the context of the exam. An alternative solution:

```
public void consolidate()
{
  // 1. Find the first empty space:
  int emptySpaceIndex = 0;
  while (emptySpaceIndex < spaces.length &&
                           spaces[emptySpaceIndex] != null)
    emptySpaceIndex++;

  // 2. Move horses:
  int i = emptySpaceIndex + 1;
```

```
  while (i < spaces.length)
  {
    if (spaces[i] != null)
    {
      spaces[emptySpaceIndex] = spaces[i];
      spaces[i] = null;
      emptySpaceIndex++;
    }
    i++;
  }
}
```

Another alternative, in which we look for the first empty space for each horse — not very efficient:

```
public void consolidate()
{
  int emptySpaceIndex = 0;
  for (int i = 0; i < spaces.length; i++)
  {
    if (spaces[i] != null)
    {
      // Find the first empty space, if any, preceeding i:
      while (emptySpaceIndex < i &&
                        spaces[emptySpaceIndex] != null)
        emptySpaceIndex++;
      if (i != emptySpaceIndex)
      {
        spaces[emptySpaceIndex] = spaces[i];
        spaces[i] = null;
      }
    }
  }
}
```

Question 4

Part (a)

```
public int countWhitePixels()
{
  int count = 0;
  for (int r = 0; r < pixelValues.length; r++)
    for (int c = 0; c < pixelValues[0].length; c++)
      if (pixelValues[r][c] == WHITE) [1]
        count++;

  return count;
}
```

Notes:

1. It would be acceptable to write 255 for `WHITE` (and 0 for `BLACK` in Part (b)).

Part (b)

```
public void processImage()
{
  for (int r = 0; r < pixelValues.length - 2; r++)
    for (int c = 0; c < pixelValues[0].length - 2; c++)
      pixelValues[r][c] = Math.max(BLACK,
                    pixelValues[r][c] - pixelValues[r+2][c+2]);
} [1]
```

Notes:

1. `Math.max` is not in the AP subset, but it is OK to use it. Alternative solution:

```
public void processImage()
{
  for (int r = 0; r < pixelValues.length - 2; r++)
  {
    for (int c = 0; c < pixelValues[0].length - 2; c++)
    {
      int p = pixelValues[r][c] - pixelValues[r+2][c+2];
      if (p < BLACK)
        p = BLACK;
      pixelValues[r][c] = p;
    }
  }
}
```