

Results from the 2012 AP Computer Science A Exam Administration

Jody Paul

Chief Reader AP Computer Science (2008–2012)
The College Board, Educational Testing Service

Associate Professor of Computer Science
Metropolitan State University of Denver

jody@computer.org • www.jodypaul.com



AP Annual Conference
July 2012 • Orlando FL

Results from the 2012 AP Computer Science A Exam Administration by Dr. Jody Paul is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.



AP Computer Science A

An introductory course in computer science (CS) that corresponds to a first CS course at colleges/universities

Curriculum and examinations are developed and maintained by the College Board through the AP Computer Science Development Committee

Examinations are administered and scored on behalf of the College Board by Educational Testing Service

AP Computer Science A

The Essentials

- Provide students with college credit and placement
- Reflect content of college introductory CS courses
- Teach more than just writing code
 - *Problem solving, algorithms, logic, ...*
 - *Introduce a way of thinking and means of expression*
- Object-Orientation & Java
 - *For the foreseeable future*

AP CS “History”

- 1984 *First Examination (Pascal)*
- 1988 *Exam split into A and AB (A subscore)*
- 1992 *A subscore eliminated*
- 1995 *First Case Study*
- 1999 *C++*
- 2004 *Java*
- 2008 *GridWorld Case Study*
- 2009 *Last year of AB exam*



2012 Exam Reading

Reading (Process & More)

At the pre-reading ...

- ▶ Question Leaders vet the rubrics and develop reader training
- ▶ Table Leaders vet the training and prepare mentoring support for readers

2009 General Scoring Guidelines

Apply the question-specific rubric first. The question-specific rubric always takes precedence. Refer to the general error categorization below only for cases not already covered by the question-specific rubric. Points can only be deducted if the error occurs in a part which has earned credit via the question-specific rubric.

Even if it occurs on different parts of the question, every 1/2 point is the only maximum score of one or two. If a specific error score exactly once when the point is required in an example.

penalty if it occurs on different parts of the question, every 1/2 point is the only maximum score of one or two. If a specific error score exactly once when the point is required in an example.

2009 A Question 3: Battery Charger — Assessment Rubric

Part A: getChargingCost 3 pts

- +1/2 access array elements
 - % accesses any element of carState
 - % accesses an element of carState using an index derived from carState
 - % accesses multiple elements of carState with no out of bounds access penalty
- +2/2 accumulate values
 - % declares and initializes an accumulator
 - % accumulates values from elements of carState
 - % indicates values from carState using an index derived from carState and charging
 - % determines correct sum of values from carState based on carState and charging
- +1 value returned
 - % returns any non-constant (derived) value
 - % returns accumulated value

Part B: getChargingTime 4 pts

- +1/2 invokes getChargingCost or replicates functionality with no errors
- +1 determine charging cost
 - % considers all potential start times, must include at least 0 ... 23
 - % determines charging cost for potential start times

Note: No penalty here for parameter passed to getChargingCost that violates its precondition (e.g., 24)
- +1 compares charging costs for two different start times
- +1 determines minimum charging cost based on potential start times
 - Note: Penalty here for using result of call to getChargingCost that violates its precondition (e.g., 24)
- +1/2 returns start time for minimum charging cost

Center Scoring Standards — <http://www.ets.org/s/standards> — © College Board — www.collegeboard.org

7/2009 1:34:57 PM

2009 A Question 4: Tile Game — Canonical Solution

1. Determine where to insert tiles, on the current iteration, into game board.
 2. Remove tiles that tile to be placed on the game board.
 3. Iterate the position of tile where tile to be inserted:
 a. If the board is empty:
 i. If tile does not fit on board, or not, an otherwise way existing tiles.
 ii. Otherwise, if a position returned 3 boards ahead.
 4. Return the first position that fits on the board.
 5. If no position returned, return null.
 6. If the board is empty, return null.
 7. If the board is empty, return null.
 8. If the board is empty, return null.
 9. If the board is empty, return null.
 10. If the board is empty, return null.
 11. If the board is empty, return null.
 12. If the board is empty, return null.
 13. If the board is empty, return null.
 14. If the board is empty, return null.
 15. If the board is empty, return null.
 16. If the board is empty, return null.
 17. If the board is empty, return null.
 18. If the board is empty, return null.
 19. If the board is empty, return null.
 20. If the board is empty, return null.
 21. If the board is empty, return null.
 22. If the board is empty, return null.
 23. If the board is empty, return null.
 24. If the board is empty, return null.
 25. If the board is empty, return null.
 26. If the board is empty, return null.
 27. If the board is empty, return null.
 28. If the board is empty, return null.
 29. If the board is empty, return null.
 30. If the board is empty, return null.
 31. If the board is empty, return null.
 32. If the board is empty, return null.
 33. If the board is empty, return null.
 34. If the board is empty, return null.
 35. If the board is empty, return null.
 36. If the board is empty, return null.
 37. If the board is empty, return null.
 38. If the board is empty, return null.
 39. If the board is empty, return null.
 40. If the board is empty, return null.
 41. If the board is empty, return null.
 42. If the board is empty, return null.
 43. If the board is empty, return null.
 44. If the board is empty, return null.
 45. If the board is empty, return null.
 46. If the board is empty, return null.
 47. If the board is empty, return null.
 48. If the board is empty, return null.
 49. If the board is empty, return null.
 50. If the board is empty, return null.
 51. If the board is empty, return null.
 52. If the board is empty, return null.
 53. If the board is empty, return null.
 54. If the board is empty, return null.
 55. If the board is empty, return null.
 56. If the board is empty, return null.
 57. If the board is empty, return null.
 58. If the board is empty, return null.
 59. If the board is empty, return null.
 60. If the board is empty, return null.
 61. If the board is empty, return null.
 62. If the board is empty, return null.
 63. If the board is empty, return null.
 64. If the board is empty, return null.
 65. If the board is empty, return null.
 66. If the board is empty, return null.
 67. If the board is empty, return null.
 68. If the board is empty, return null.
 69. If the board is empty, return null.
 70. If the board is empty, return null.
 71. If the board is empty, return null.
 72. If the board is empty, return null.
 73. If the board is empty, return null.
 74. If the board is empty, return null.
 75. If the board is empty, return null.
 76. If the board is empty, return null.
 77. If the board is empty, return null.
 78. If the board is empty, return null.
 79. If the board is empty, return null.
 80. If the board is empty, return null.
 81. If the board is empty, return null.
 82. If the board is empty, return null.
 83. If the board is empty, return null.
 84. If the board is empty, return null.
 85. If the board is empty, return null.
 86. If the board is empty, return null.
 87. If the board is empty, return null.
 88. If the board is empty, return null.
 89. If the board is empty, return null.
 90. If the board is empty, return null.
 91. If the board is empty, return null.
 92. If the board is empty, return null.
 93. If the board is empty, return null.
 94. If the board is empty, return null.
 95. If the board is empty, return null.
 96. If the board is empty, return null.
 97. If the board is empty, return null.
 98. If the board is empty, return null.
 99. If the board is empty, return null.
 100. If the board is empty, return null.

7/2009 12:05



- Social Lounge (Hyatt Hotel)
- Opening Night Shmooze
- Meet the DC & CB Forum
- Professional Night
- Toy Night (Pedagogical Practices)
- Dine Out & Reds Baseball Game
- Puzzle Night by Scott Weiss!
- APCS Closing Night Social

D I C A M A



reading, intra question (re)calibration, time metrics, ...

D I C M A



Reading, intra-question (re)calibration, time metrics, ...

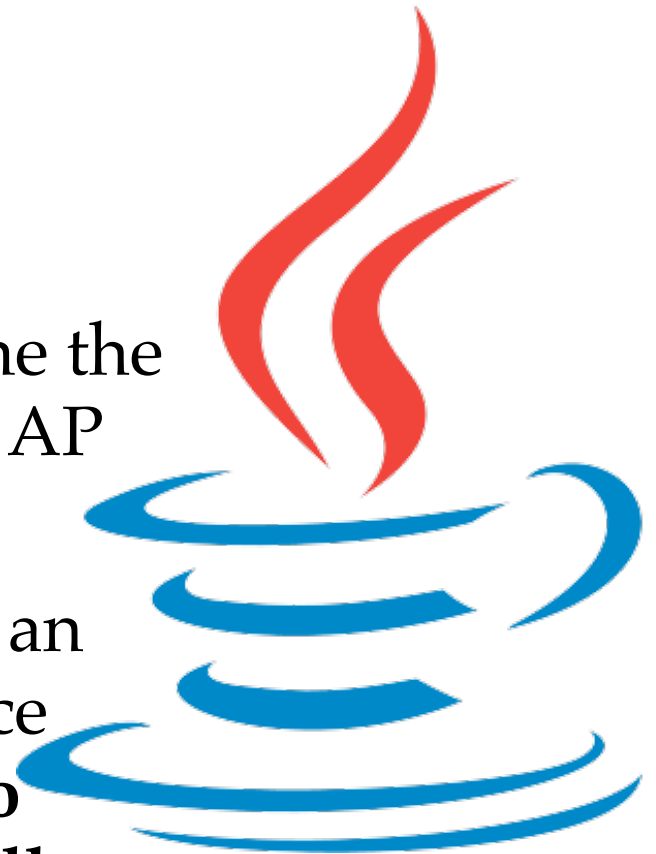
2012 exam in Java

Questions designed with the APCS “testable” Java subset in mind

Appendix A: AP Computer Science Java Subset

The AP Java subset is intended to outline the features of Java that may appear on the AP Computer Science A Exam.

The AP Java subset is **NOT** intended as an overall prescription for computer science courses — **the subset itself will need to be supplemented in order to address all topics** in a typical introductory curriculum.



2012 exam in Java

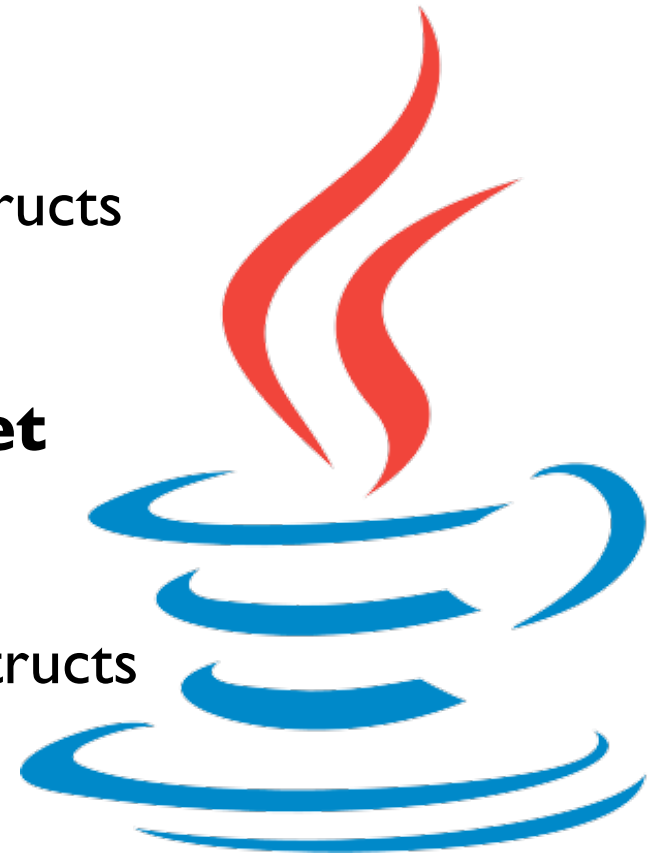
Questions designed with the APCS “testable” Java subset in mind

No question or solution required any constructs or classes outside that subset

Solutions not restricted to that subset

All correct solutions earn full credit

Solutions may utilize any standard Java constructs or classes (inside or outside the subset)



As in previous years, **some minor errors are ignored** when scoring
For example, a few missing semicolons, unambiguous case discrepancies, ...

CS A Exam

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total score—50

26 minutes per question
(13 minutes per part if a / b)
to **read, understand,**
design/solve, code, check

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

AP CS A 2012 Exam Questions

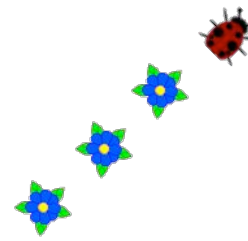
Q1: **CimbingClub**

List manipulation, design analysis



Q2: **RetroBug**

Extend **Bug** in GridWorld



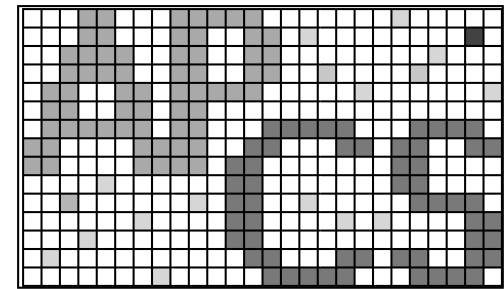
Q3: **HorseBarn**

Manipulation of 1D array of objects



Q4: **GrayImage**

Manipulation of 2D array of primitives



Q3: HorseBarn — *Preface*

3. Consider a software system that models a horse barn. Classes that represent horses implement the following interface.

```
public interface Horse
{
    /** @return the horse's name */
    String getName();

    /** @return the horse's weight */
    int getWeight();

    // There may be methods that are not shown.
}
```

A horse barn consists of N numbered spaces. Each space can hold at most one horse. The spaces are indexed starting from 0; the index of the last space is $N - 1$. No two horses in the barn have the same name.

The declaration of the `HorseBarn` class is shown below. You will write two unrelated methods of the `HorseBarn` class.

Q3: HorseBarn — Preface

```
public class HorseBarn
{
    /** The spaces in the barn. Each array element holds a reference to the horse
     * that is currently occupying the space. A null value indicates an empty space.
     */
    private Horse[] spaces;

    /** Returns the index of the space that contains the horse with the specified name.
     * Precondition: No two horses in the barn have the same name.
     * @param name the name of the horse to find
     * @return the index of the space containing the horse with the specified name;
     *         -1 if no horse with the specified name is in the barn.
     */
    public int findHorseSpace(String name)
    { * to be implemented in part (a) */ }

    /** Consolidates the barn by moving horses so that the horses are in adjacent spaces,
     * starting at index 0, with no empty space between any two horses.
     * Postcondition: The order of the horses is the same as before the consolidation.
     */
    public void consolidate()
    { * to be implemented in part (b) */ }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

Q3: HorseBarn — Part a

- (a) Write the `HorseBarn` method `findHorseSpace`. This method returns the index of the space in which the horse with the specified name is located. If there is no horse with the specified name in the barn, the method returns -1.

For example, assume a `HorseBarn` object called `sweetHome` has horses in the following spaces.

0	1	2	3	4	5	6
"Trigger" 1340	null	"Silver" 1210	"Lady" 1575	null	"Patches" 1350	"Duke" 1410

The following table shows the results of several calls to the `findHorseSpace` method.

Method Call	Value Returned	Reason
<code>sweetHome.findHorseSpace("Trigger")</code>	<u>0</u>	A horse named Trigger is in space 0.
<code>sweetHome.findHorseSpace("Silver")</code>	<u>2</u>	A horse named Silver is in space 2.
<code>sweetHome.findHorseSpace("Coco")</code>	<u>-1</u>	A horse named Coco is not in the barn.

Q3: HorseBarn — Part b

- (b) Write the `HorseBarn` method `consolidate`. This method consolidates the barn by moving horses so that the horses are in adjacent spaces, starting at index 0, with no empty spaces between any two horses. After the barn is consolidated, the horses are in the same order as they were before the consolidation.

For example, assume a barn has horses in the following spaces.

0	1	2	3	4	5	6
"Trigger" 1340	null	"Silver" 1210	null	null	"Patches" 1350	"Duke" 1410

The following table shows the arrangement of the horses after `consolidate` is called.

0	1	2	3	4	5	6
"Trigger" 1340	"Silver" 1210	"Patches" 1350	"Duke" 1410	null	null	null

Q3: HorseBarn – *Example Solution*

Part a.

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

Part b.

```
public void consolidate() {
    Horse[] newSpaces = new Horse[this.spaces.length];
    int nextSpot = 0;
    for (Horse nextHorse : this.spaces) {
        if (nextHorse != null) {
            newSpaces[nextSpot] = nextHorse;
            nextSpot++;
        }
    }
    this.spaces = newSpaces;
}
```

Q3: HorseBarn – Student Solution

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

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

Q3: HorseBarn – Scoring Rubric

2012 AP Computer Science A

Question 3: Horse Barn

Assessment Rubric

Part A: `findHorseSpace`

4 pts

Intent: Return index of space containing horse with specified name.

- +1 Accesses all entries in spaces (No bounds errors.)
- +1 Checks for null reference in array and avoids dereferencing it (in context of loop)
- +1 Checks for name equality between array element and parameter (Must use String equality check.)
- +1 Returns correct index if present; -1 if not

Part B: `consolidate`

Intent: Repopulate spaces such that the order of non-null entries is preserved and all null entries are found contiguous.

- +1 Accesses all entries.
- +1 Identifies and provides indices of non-null entries in array.
- +1 Assigns element to destination (non-null or destination as null).
- +1 On exit: The non-null entries in spaces is preserved.
- +1 On exit: All non-null elements are contiguous, starting at index 0 (no destruction of data).

**Whole
Point
Scoring**

Question-Specific Penalties

- 1 (z) Attempt to return a value from `consolidate`
- 2 (v) Consistently uses incorrect array name instead of `spaces`

Q3: HorseBarn – Scoring Rubric Applied

Part A: findHorseSpace

4 pts

Intent: Return index of space containing horse with specified name.

- +1 Accesses all entries in spaces (No bounds errors.)
- +1 Checks for null reference in array and avoids dereferencing it (in context of loop)
- +1 Checks for name equality between array element and parameter (Must use String equality check.)
- +1 Returns correct index if present; -1 if not

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

- Earns 1 point for accessing all entries of the spaces array [no bounds errors]
- Earns 1 point for preventing null dereference of array [in context of loop]
- Earns 1 point for checking name equality between array element and parameter
- Earns 1 point for returning correct index and -1 if not present

General Scoring Guidelines

2012 AP Computer Science A General Scoring Guidelines

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question.

1-Point Penalty

- (w) Extraneous code that causes side-effect or prevents earning points in rubric (e.g., writing to output, failure to compile, runtime error)
- (x) Local variables used but none declared
- (y) Destructor for persistent data (e.g., changing reference by parameter)
- (z) Void method or constructor that returns a value

No Penalty

- o Extraneous code that causes side-effect
- o Extraneous code
- o Spelling/case discrepancy
- o Local variable not declared
- o private qualifier on class
- o Missing method
- o Missing constructor
- o Missing `return`
- o `++` instead of `+`
- o Array/collection element
- o Array/collection
- o `length/size`
- o Extraneous `return`
- o `return` with `return`
- o Extraneous code used with `return`
- o Missing `return`; provided line break
- o Missing `{}` where indentation indicates otherwise
- o Missing `()` on parameter-less constructor
- o Missing `()` around `if` or `while`
- o Use of local variables outside of scope (must be within method body)
- o Failure to cast object retrieved from non-generic collection

Simplified General Scoring Guidelines

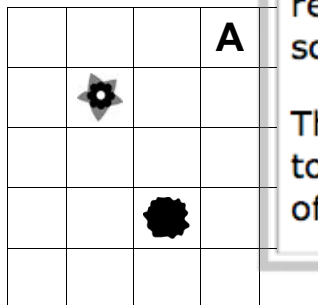
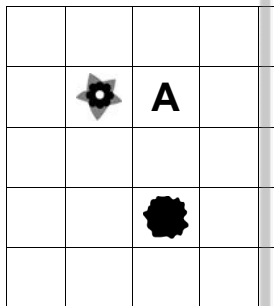
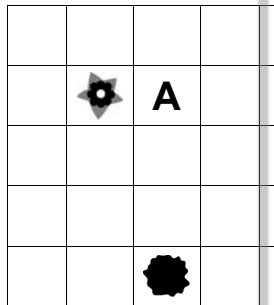
"Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be unambiguously inferred from context. For example, "ArrayList" instead of "Arraylist". As a counter example, note that if the code declares "Bug bug", then uses "Bug.move()" instead of "bug.move()", the context does not allow for the reader to assume the object instead of the class.

Question Readability 2011

2. This question involves reasoning about the GridWorld case study. Reference materials are provided in the Appendix.

An attractive critter is a critter that processes other actors in the grid, including other attractive critters. The closer to itself in the direction specified by its location into which it would be relocated. After trying to move all other actors, the

The following series of figures represents the grid.



The order in which the actors in the grid are processed is not specified, making it possible to get different results from the same grid of actors.

instance variables and required methods. Do not violate the postconditions of the methods. Do not change the state of that object.

Readability index calculator

*** Text:** This question involves reasoning about the GridWorld case study. Reference materials are provided in the Appendix. An attractive critter is a critter that processes other actors by attempting to relocate all of the other actors in the grid, including other attractive critters. The attractive critter attempts to move each other

Method: Flesch-Kincaid (English)

Calculate score

Result

Method used: Flesch-Kincaid (English).

Flesch-Kincaid Grade level: **14.**

Flesch-Kincaid Reading Ease score: **29.**

The [Flesch-Kincaid Reading Ease](#) score indicates how easy a text is to read. A high score implies an easy text. In comparison comics typically score around 90 while legalese can get a score below 10.

The *Flesch-Kincaid Grade level* indicates the grade a person will have to have reached to be able to understand the text. E.g. a grade level of 7 means that a seventh grader will be able to understand the text.

Question Readability 2012

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendices.

Example 2

A retro bug behaves like a regular bug. When a retro bug acts, it maintains information about its previous location and direction. A retro bug has a `restore` method that occurs at the beginning of its previous act. A retro bug can move multiple times, and each time it moves, it updates its previous location and direction. A retro bug's `restore` method has no effect if it is called before the bug has moved.

The `restore` method takes no parameters and has the following functionality.

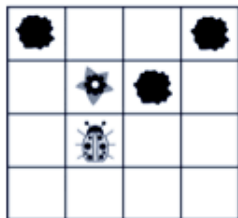
- If the previous location of the retro bug is not null, it places the retro bug in that previous location and direction, and prevents the retro bug from being replaced.
- The `restore` method always occurs at the beginning of its most recent act.

The following examples illustrate the behavior of the `restore` method.

Example 1

The retro bug acts once and later calls `restore`. The retro bug is not replaced as a result of the call to `restore`, which, in this case, is the same as the call to `act`.

Initial Configuration



Readability index calculator

*** Text:** This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendices. A retro bug behaves like a regular bug. It also has the ability to revert to its previous location and direction. When a retro bug acts, it maintains information about its location and direction at the beginning of the act. The retro bug has a `restore` method that occurs at the beginning of its previous act. A retro bug's `restore` method has no effect if it is called before the bug has moved.

Method: Flesch-Kincaid (English)

Calculate score

Result

Method used: Flesch-Kincaid (English).

Flesch-Kincaid Grade level: **11.**

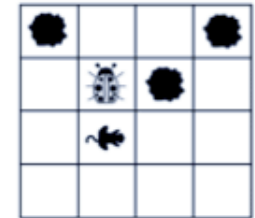
Flesch-Kincaid Reading Ease score: **45.**

The [Flesch-Kincaid Reading Ease](#) score indicates how easy a text is to read. A high score implies an easy text. In comparison comics typically score around 90 while legalese can get a score below 10.

The [Flesch-Kincaid Grade level](#) indicates the grade a person will have to have reached to be able to understand the text. E.g. a grade level of 7 means that a seventh grader will be able to understand the text.

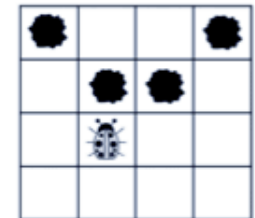
Example 2
The retro bug acts once and later calls `restore`. The retro bug is returned to its previous location. The retro bug is returned to its previous location and direction.

After Call to Restore



Example 3
The retro bug acts once and later calls `restore`. The retro bug is blocked from moving forward, it stays in its current location (the same as the call to `act`).

After Call to Restore



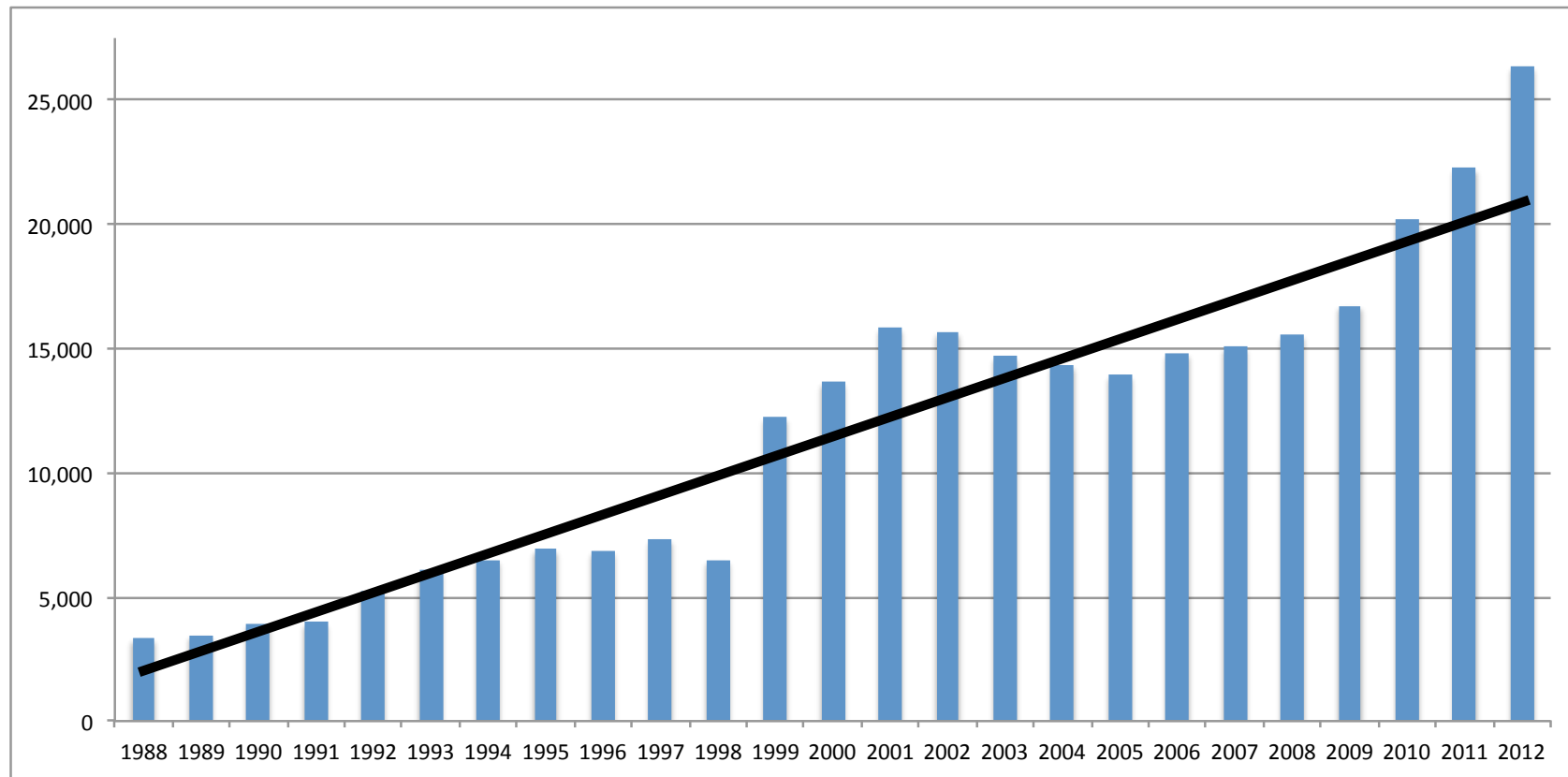
Question 2 continues on the next page.

NEXT PAGE.

Results

AP Computer Science A

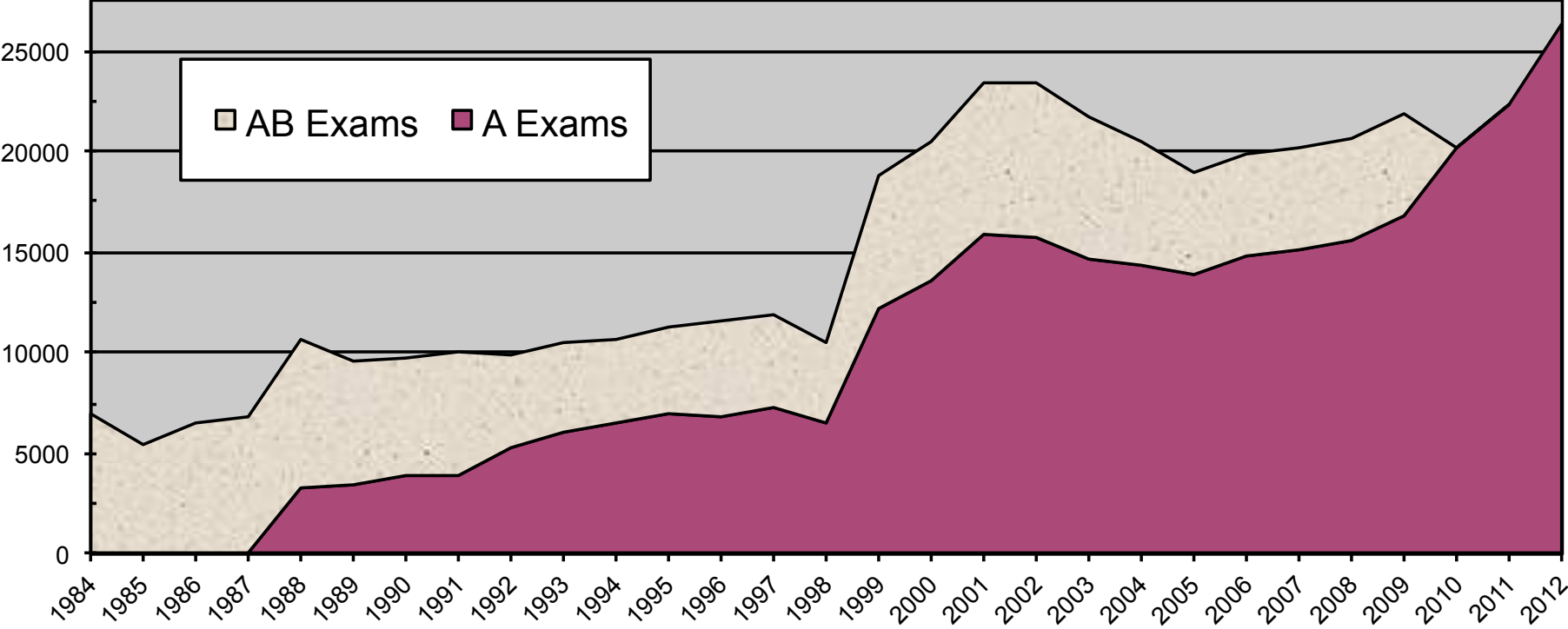
26,350 Exams



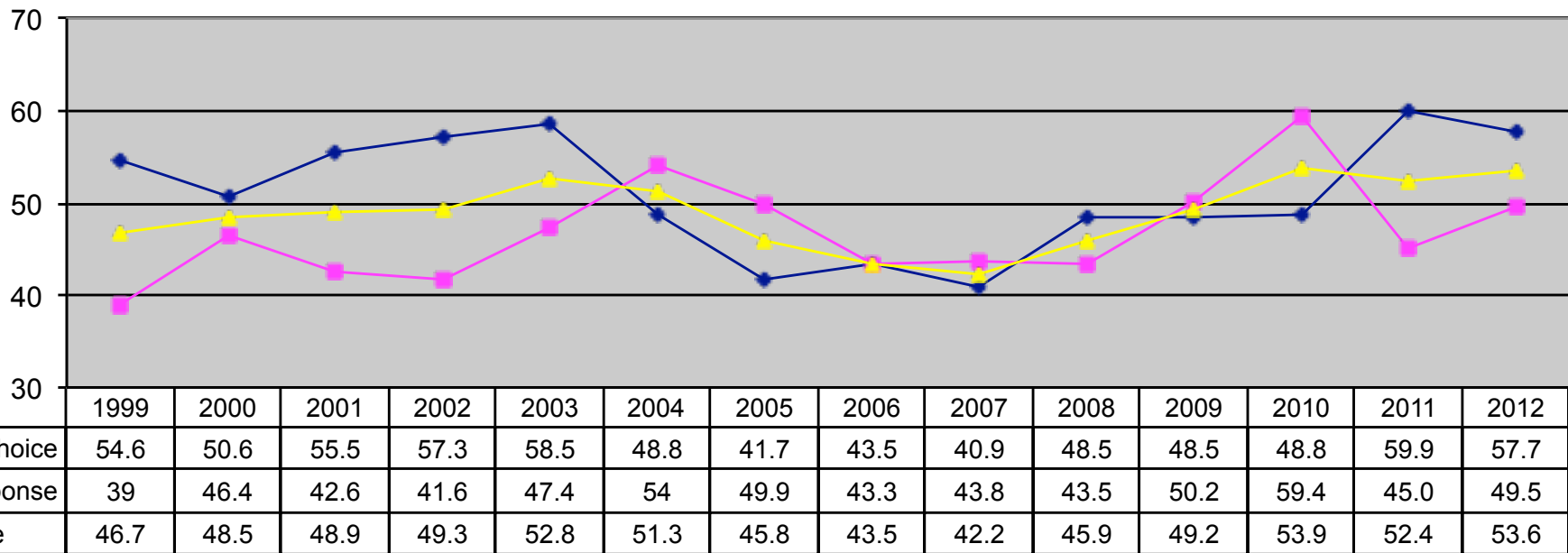
+18% from 2011 22,250

+30% from 2010 20,200

Number of APCS Exams

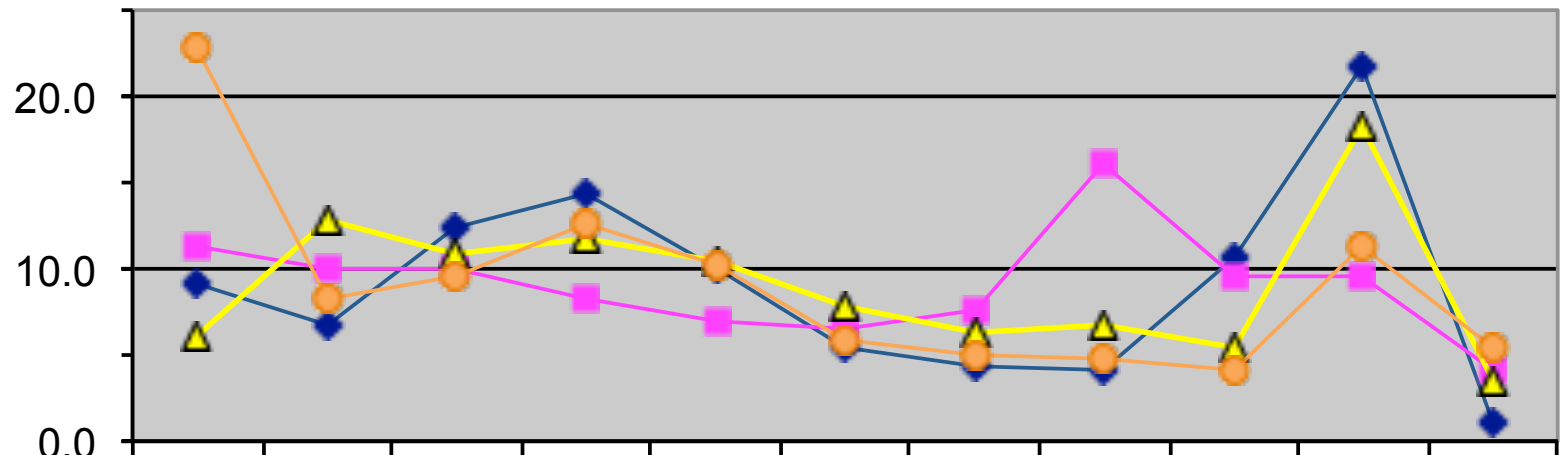


Mean as % of Max Possible



APCS A Exam: Mean as % of Max Possible Score

Question Score Distributions



	9	8	7	6	5	4	3	2	1	0	-
◆ Q1: ClimbingClub	9.2	6.7	12.3	14.4	9.9	5.5	4.3	4.1	10.7	21.8	1.0
■ Q2: RetroBug	11.2	10.0	10.0	8.3	7.0	6.6	7.5	16.0	9.6	9.6	4.1
▲ Q3: HorseBarn	6.1	12.8	10.8	11.8	10.5	7.8	6.4	6.6	5.3	18.2	3.6
● Q4: GrayImage	22.8	8.2	9.6	12.7	10.2	5.8	5.0	4.8	4.1	11.4	5.3

2012 APCS A: % of Exams Receiving Score

Mean Scores: **q1 4.13** **q2 4.27** **q3 4.26** **q4 5.17**

Mean no 0/-: **q1 5.35** **q2 4.95** **q3 5.44** **q4 6.21**

Common Errors

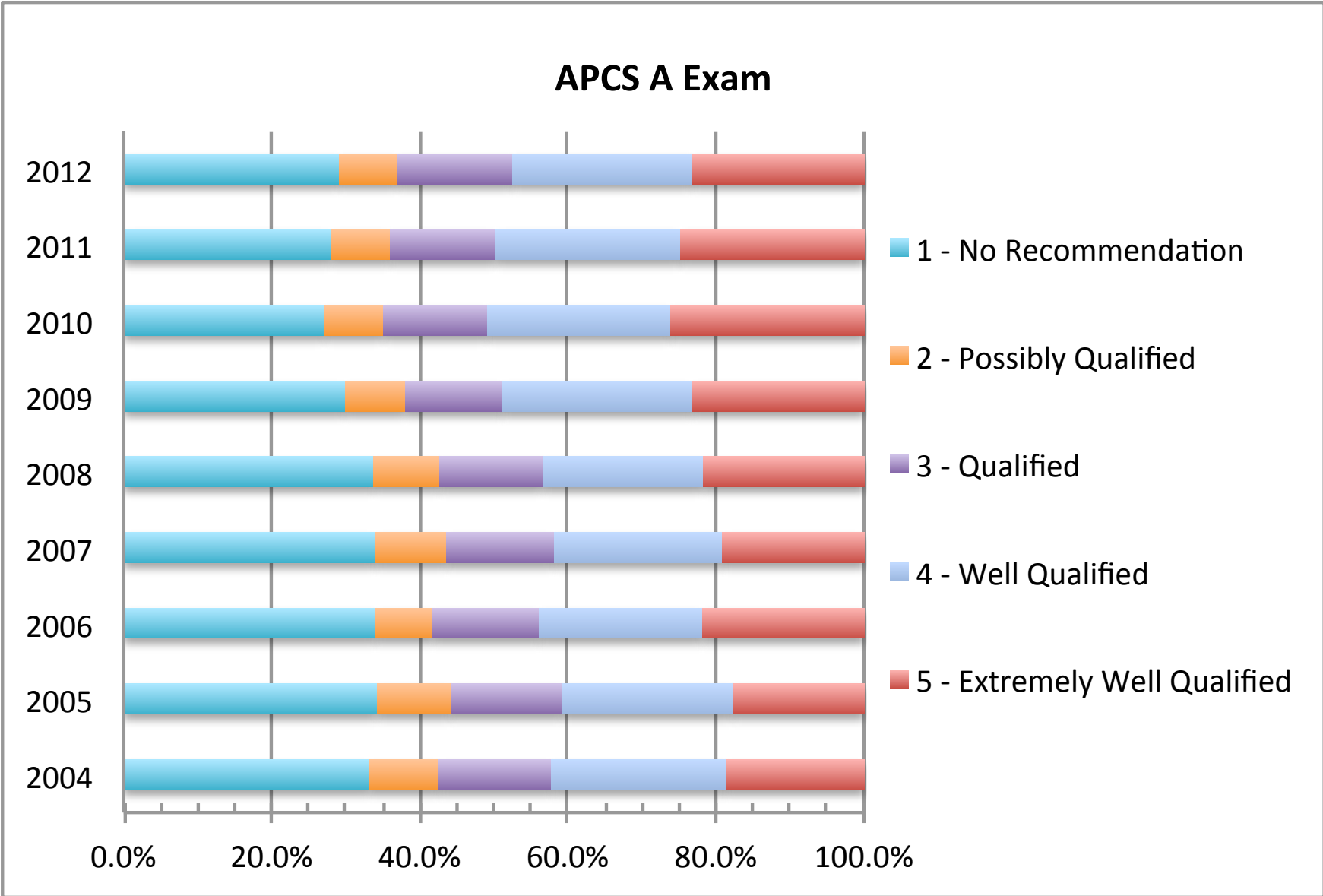
	object construction	
<i>array</i>	array access & modification	<i>bounds</i>
<i>length</i>	2D arrays	<i>ArrayList</i>
<i>for, while,</i>	ArrayList manipulation	
<i>enhanced for</i>	loop-based iteration	<i>OOP</i>
	method overriding	<i>compareTo</i>
<i>instance variables</i>	String comparison	
	class design	
	method invocation	<i>object reference</i>
	<u>compound boolean expressions</u>	
<i>not just Critter</i>	case-study breadth	
	null	<i>null</i>

Grade Setting

The Chief Reader, ETS content assessment specialists, College Board representatives, and ETS statisticians are involved in a grade-setting meeting where the data and analyses are presented, including comparisons with previous years, and at which the parties establish and agree to finalized cut-points (mapping scores to reports of 1, 2, 3, 4 & 5).

Last year there was no such grade-setting meeting because the ETS statisticians were overloaded with addressing the change in scoring of the multiple choice section.

Score (1...5) Distribution



Determined primarily by using a subset of the multiple-choice questions (“equators”)

Score (1...5) Distribution

AP Computer Science A

Grade	2004	2005	2006	2007	2008	2009	2010	2011	2012
5	18.6%	17.7%	21.8%	19.1%	21.7%	23.3%	26.1%	24.8%	23.3%
4	23.6%	23.1%	22.1%	22.7%	21.7%	25.6%	24.8%	25%	24.2%
3	15.2%	15.0%	14.4%	14.6%	13.9%	13.1%	14%	14.2%	15.6%
2	9.5%	10.0%	7.7%	9.6%	9.0%	8.1%	8.0%	8.0%	7.8%
1	33.1%	34.2%	34.0%	34.0%	33.7%	29.9%	27.1%	28%	29.1%

Recommendations

Recommendations

Address the Common Errors

	object construction	
<i>array</i>	array access & modification	<i>bounds</i>
<i>length</i>	2D arrays	<i>ArrayList</i>
<i>for, while,</i>	ArrayList manipulation	
<i>enhanced for</i>	loop-based iteration	<i>OOP</i>
	method overriding	<i>compareTo</i>
	String comparison	
<i>instance variables</i>	class design	
	method invocation	<i>object reference</i>
	<u>compound boolean expressions</u>	
<i>not just Critter</i>	case-study breadth	
	null	<i>null</i>

Recommendations Cross-out to “erase”

```
public void consolidate() {
    int start = 0;
    for (int s = 0; s < sample.size(); s++) {
        if (sample.get(s) == 0) {
            start = s;
        }
    }
    // for (int s = start; s < sample.size(); s++) {
    //     if (sample.get(s) == 0) {
    //         // do something
    //     }
    // }
    int i;
    for (int s = 0; s < sample.size(); s++) {
        if (sample.get(s) == 0) {
            // do something
        }
    }
}
```

Start reading here

```
{
    for (int s = 0; s < sample.size(); s++) {
        if (sample.get(s) == 0) {
            // do something
        }
    }
}
```

Recommendations

Use AP CS A exam as model
for course assessment
(especially formative)

Questions

both MC & FR

(historical, samples, reviews, student-generated)

Rubrics

(applied by students as well as instructor)

Recommendations

Execute solution code

All “solutions” are verifiable!

Applies to MC & FR questions

Write test harnesses as necessary

Recommendations

The image displays a Java IDE interface with three overlapping windows:

- Class Diagram:** Shows a class hierarchy with `ClimbInfo` as a base class for `ClimbingClub`. `ClimbingClub` has a unit test `<<unit test>> ClimbingClubTest`. Other classes like `RetroBug` are partially visible.
- GridWorld:** A window titled "GridWorld" with a menu bar (World, Location, Help) and a grid. A yellow instruction bar says "Click on a grid location to construct or manipulate an actor." The grid contains several black circular actors and a red ladybug actor.
- BlueJ: Test Results:** A window showing the results of a test run. It lists several test methods with green checkmarks, indicating they passed. A green progress bar is shown above the summary.

BlueJ: Test Results Summary:

- Runs: 9/9
- Errors: 0
- Failures: 0
- Total Time: 1309ms

The test results list includes:

- ✓ GrayImageTest.countWhitePixels (7ms)
- ✓ GrayImageTest.processImage (1ms)
- ✓ RetroBugTest.runRetroBug (1290ms)
- ✓ ClimbingClubTest.addClimbPartA (3ms)
- ✓ ClimbingClubTest.addClimbPartB (1ms)
- ✓ ClimbingClubTest.distinctPeakNamesA (4ms)
- ✓ ClimbingClubTest.distinctPeakNamesB (1ms)
- ✓ HorseBarnTest consolidate (1ms)
- ✓ HorseBarnTest.findHorse (1ms)

Buttons at the bottom of the test results window include "Show Source" and "Close".

Online Resources

<http://apcentral.collegeboard.com>

AP Central: AP info, course descriptions, materials, ...

<http://www.collegeboard.com>

College Board: general info about CB, the AP program, ...

<http://apcs.jodypaul.com>

Chief Reader Jody Paul's unofficial APCS site

jody@computer.org



AP Annual Conference
July 2012 • Orlando FL