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

CollegeBoard inspiring minds™

AP Annual Conference
July 2012 • Orlando FL

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"

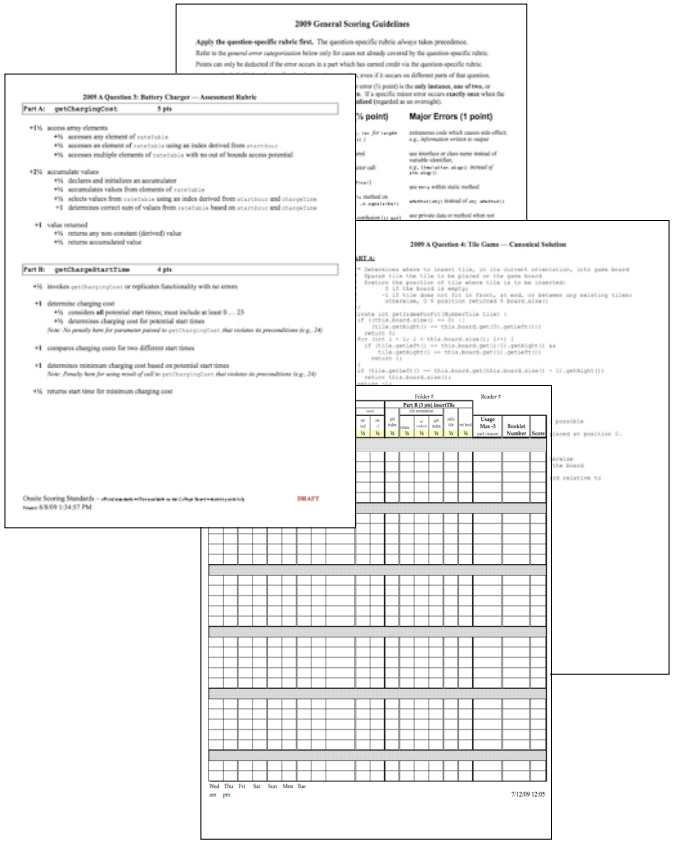
1984	*First Examination* (Pascal)
1988	*Exam split into* A *and* AB (A *subscore*)
1992	A *subscore eliminated*
1995	*First* Case Study
1999	C++
2004	Java
2008	GridWorld *Case Study*
2009	*Last year of* AB *exam*

2012 Exam Reading

Reading *(Process & More)*

Prior to the pre-reading …

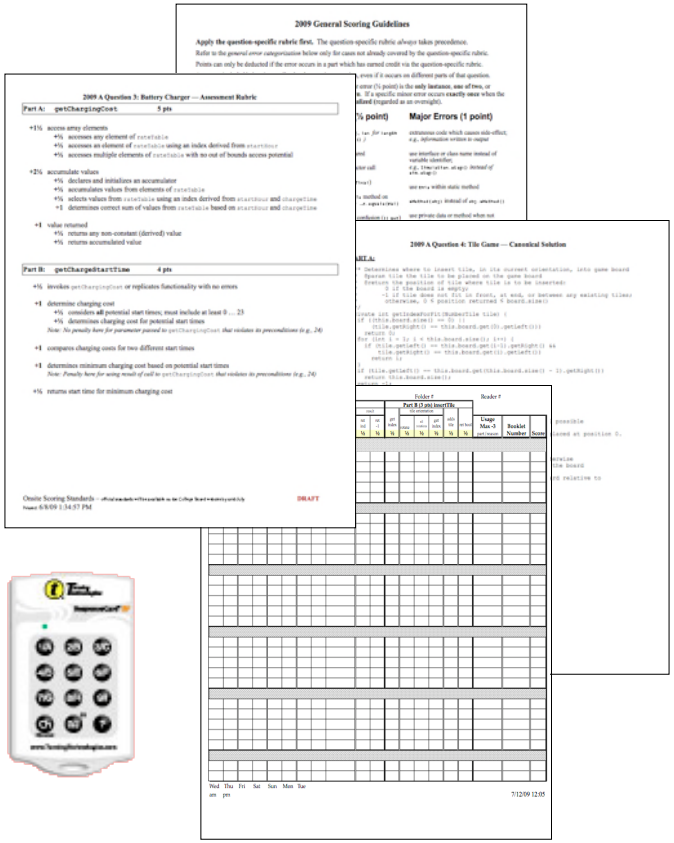
- The Chief Reader & CRD developed the scoring guidelines, rubrics, canonicals, processes
- Exam Leaders review and help refine them



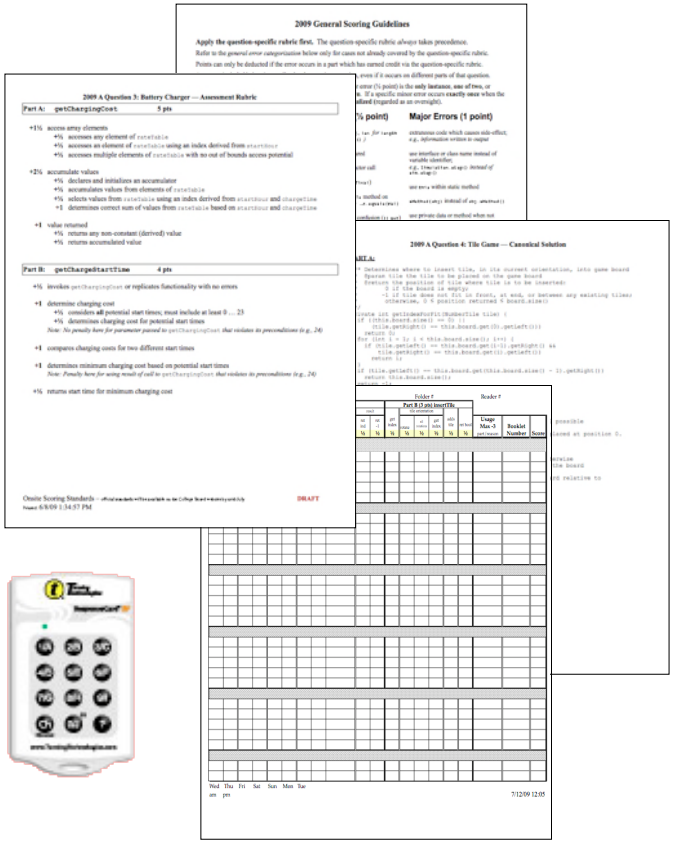
Reading *(Process & More)*

At the pre-reading …

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



Reading *(Process & More)*



At the reading …

- **Training** & **Reading**
- Table Leaders mentor readers and help in applying the rubric
- Readers work 8am – 5pm for 7 days
  (7.5 hours reading exams, 2x15-minute break, 1-hour lunch)
- Consistency checks built into the process to support readers
  - Training packs, “Swap” packs, Buddy system, Check-reading, Intra-question (re)calibration, RMS metrics, …

- 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

reading, intra-question (re)calibration, TIMS methods, …

reading, intra-question (re)calibration, TIMSS metrics, …

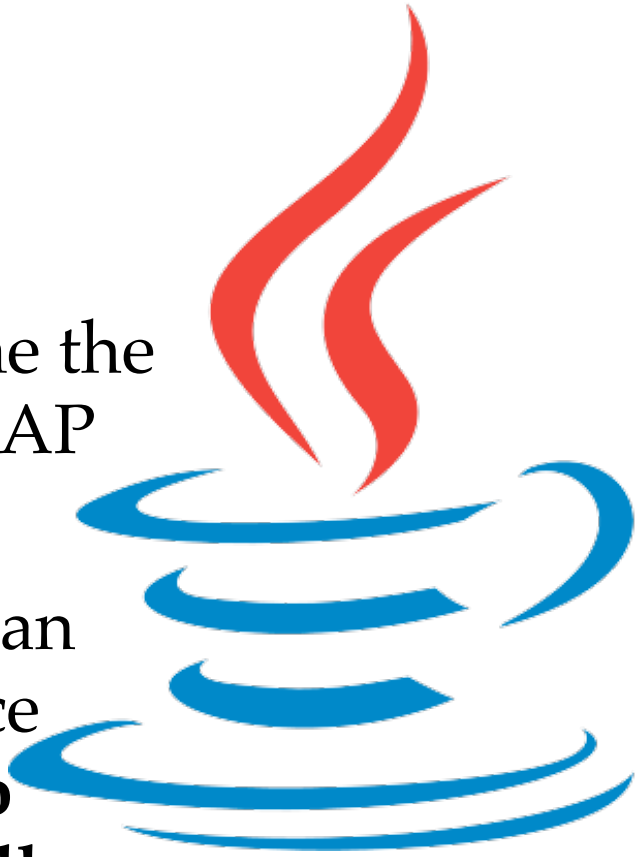
2012 exam in Java

***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

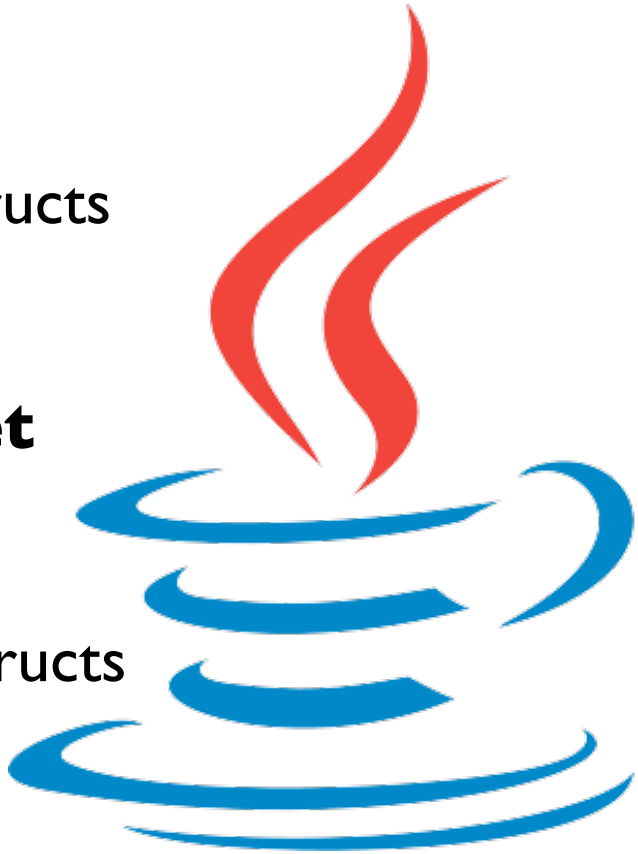
Questions designed with the APCS "testable" Java subset in mind

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

Solutions <u>not</u> 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

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

COMPUTER SCIENCE A
SECTION II
Time—1 hour and 45 minutes
Number of questions—4
Percent of total score—50

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

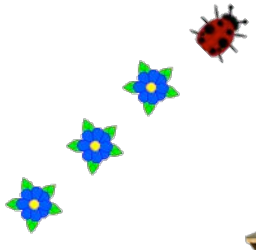
Notes:

- 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**

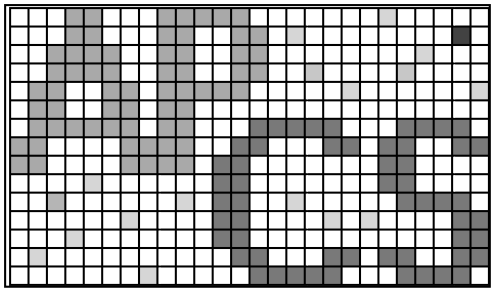
Extend **Bug** in GridWorld

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.

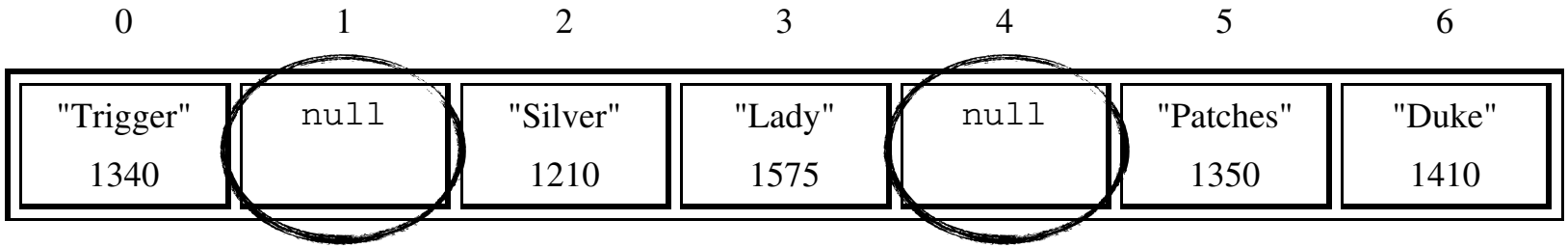
Q3: HorseBarn — *Preface*

```
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.



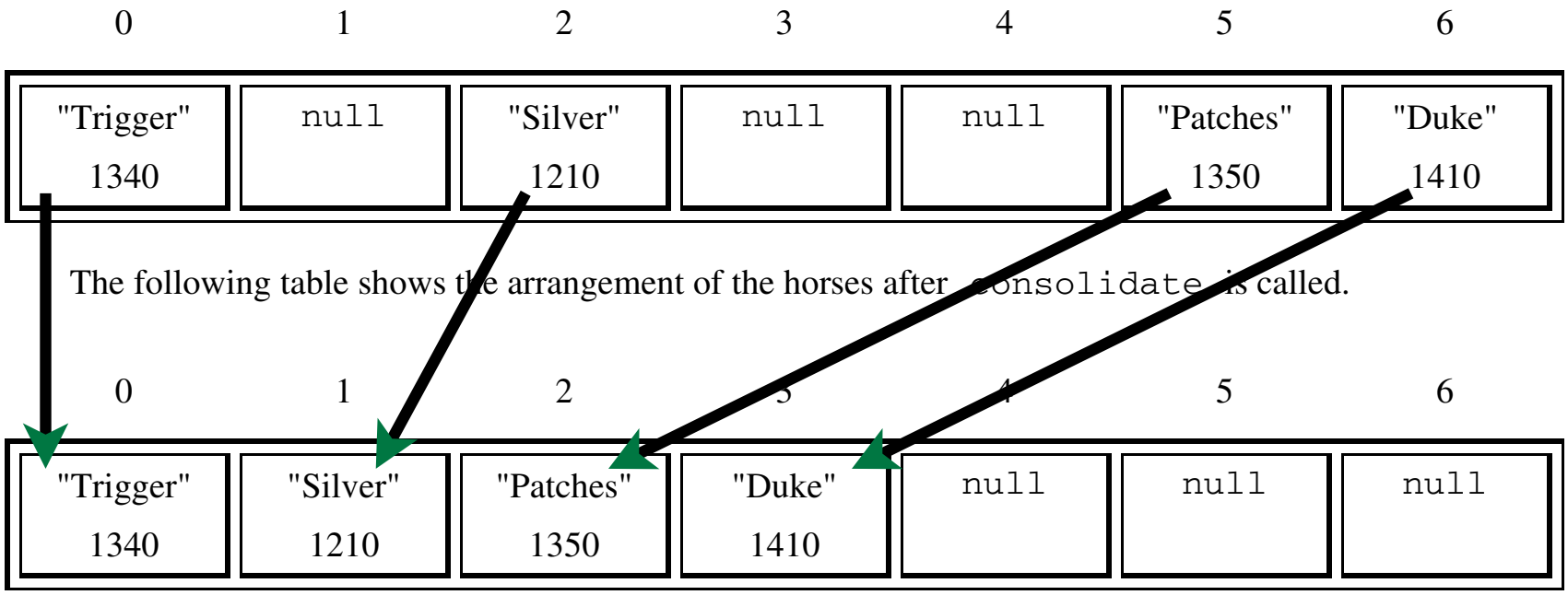
The following table shows the results of several calls to the `findHorseSpace` method.

Method Call	Value Returned	Reason
`sweetHome.findHorseSpace("Trigger")`	0	A horse named Trigger is in space 0.
`sweetHome.findHorseSpace("Silver")`	2	A horse named Silver is in space 2.
`sweetHome.findHorseSpace("Coco")`	-1	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.



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

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 sp… …h that the ord… …l entries is pr… …d all null entries are found con…*

+1 Accesses all entrie…

+1 Identifies and provides … …nents in array

+1 Assigns element … *non-null or destination as null)*

+1 On exit: The n… …ents in `spaces` is preserved and the number of nulls is pr…

+1 On exit: All non-null eleme… …cations beginning 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 pr ts earning points in rubric
(e.g., writing to output, failure to compile, runtim)

(x) Local variables used but none declared

(y) Destructior persistent data *(e.g., changin renced by parameter)*

(z) Void metho structor that returns a

No Penalty

- Extraneous con side-e
- Extraneous code c
- Spelling/case disc
- Local variable not d art
- `private` qualifier on
- Missing

Simplified General Scoring Guidelines

- `[]` v
- instead o
- Array/collection el
- Array/collection
- `length`/`si`
- Extranec
- `i, j`
- Ex ay uecl
- ; provided line brea t
- Missing { } where indentation ite elsewhere
- Missing ( ) on parameter-less structor
- Missing ( ) around `if` or `whi` s
- Use of local variables outsid scope (must be wi method body)
- Failure to cast object retrieve non-generic collection

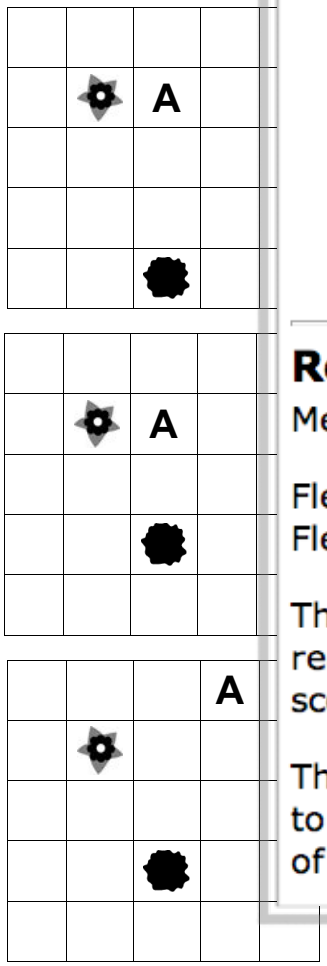
*"Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously** inferred from context. For example, "ArayList" instead of "ArrayList". As a counter example, note that if the code declares "Bug bug;", then uses "Bug.move()" instead 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 proces
grid, including other attractive critters. Th
closer to itself in the direction specified b
location into which it would be relocated
After trying to move all other actors, the a

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.

nstance variables and required methods. Do
not violate the postconditions of the methods
ble changes 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**

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

A retro bug behaves like a regular bug. When a retro bug acts, it maintains inf... retro bug has a `restore` method th... beginning of its previous act. A retro b... multiple calls to `restore` that occu... method has no effect if it is called befo...

The `restore` method takes no para... following functionality.

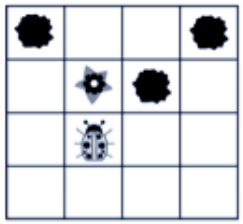
- If the previous location of the retr... places the retro bug in that previou... prevent the retro bug from being g...
- The `restore` method always e... at the beginning of its most recent...

The following examples illustrate the b...

Example 1

The retro bug acts once and later calls ... bug is not replaced as a result of the ca... which, in this case, is the same as the ...

Initial Configuration

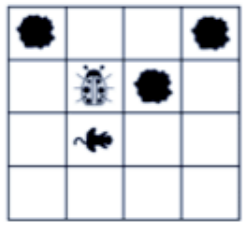


Question 2 continues on the next page.

Example 2

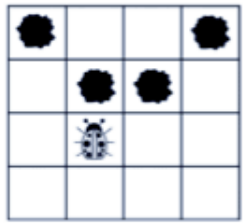
...ation that the retro bug originally held. The ...ion. The retro bug is returned to its previous ...

After Call to Restore



...bug is blocked from moving forward, it ...staying in its current location (the same as ...

After Call to Restore



...EXT PAGE.

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: Flesch-Kincaid (English)

Calculate score

Result

Method used: Flesch-Kincaid (English).

Flesch-Kincaid Grade level: **11**.
Flesch-Kincaid Reading Ease score: **45**.

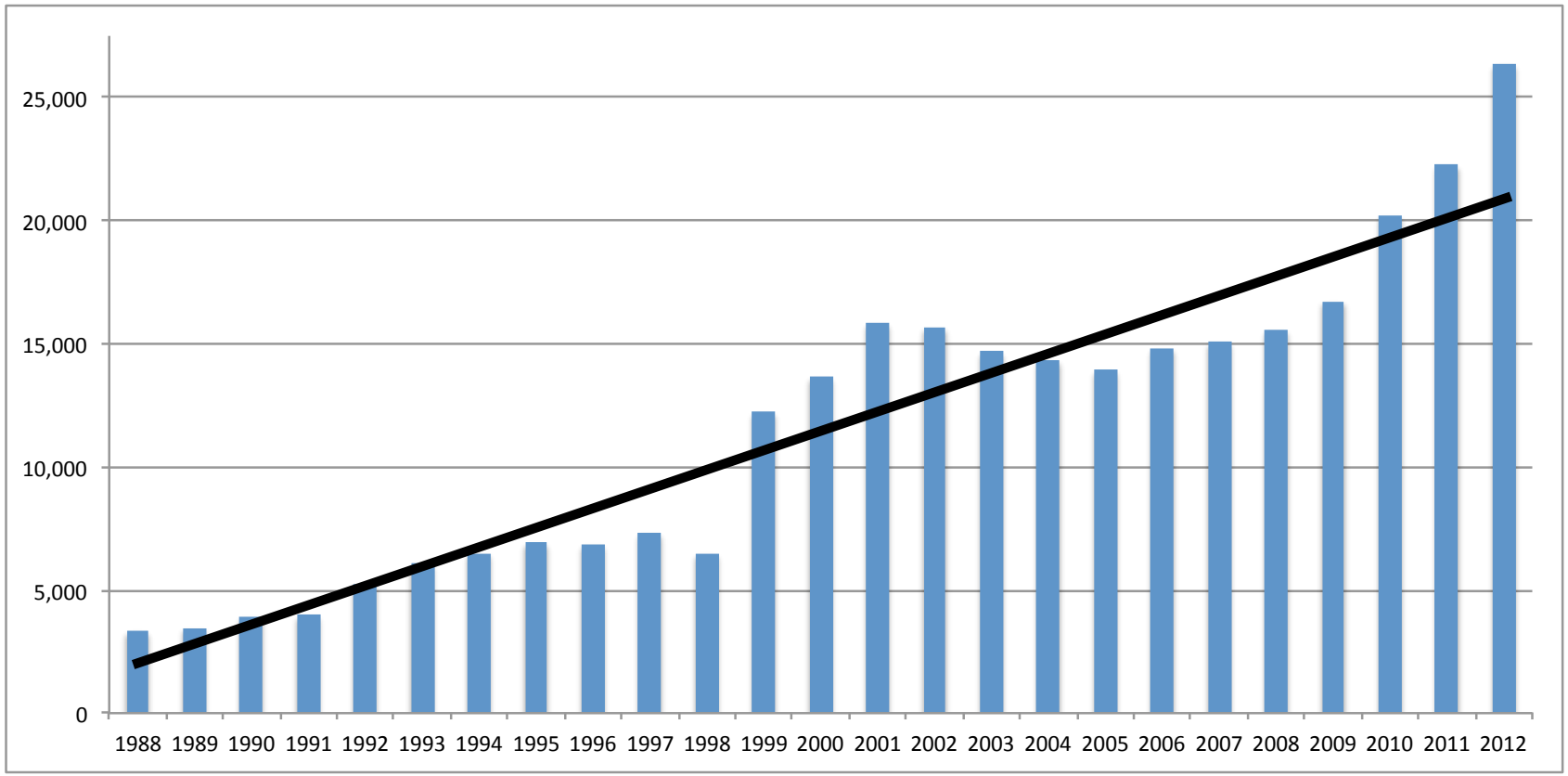
The *Flesch-Kincaid Reading Ease* score indicates how easy a text is to read. A high score implies an easy text. In comparison comics typically score around 90 while legalese can get a score below 10.

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.

Results

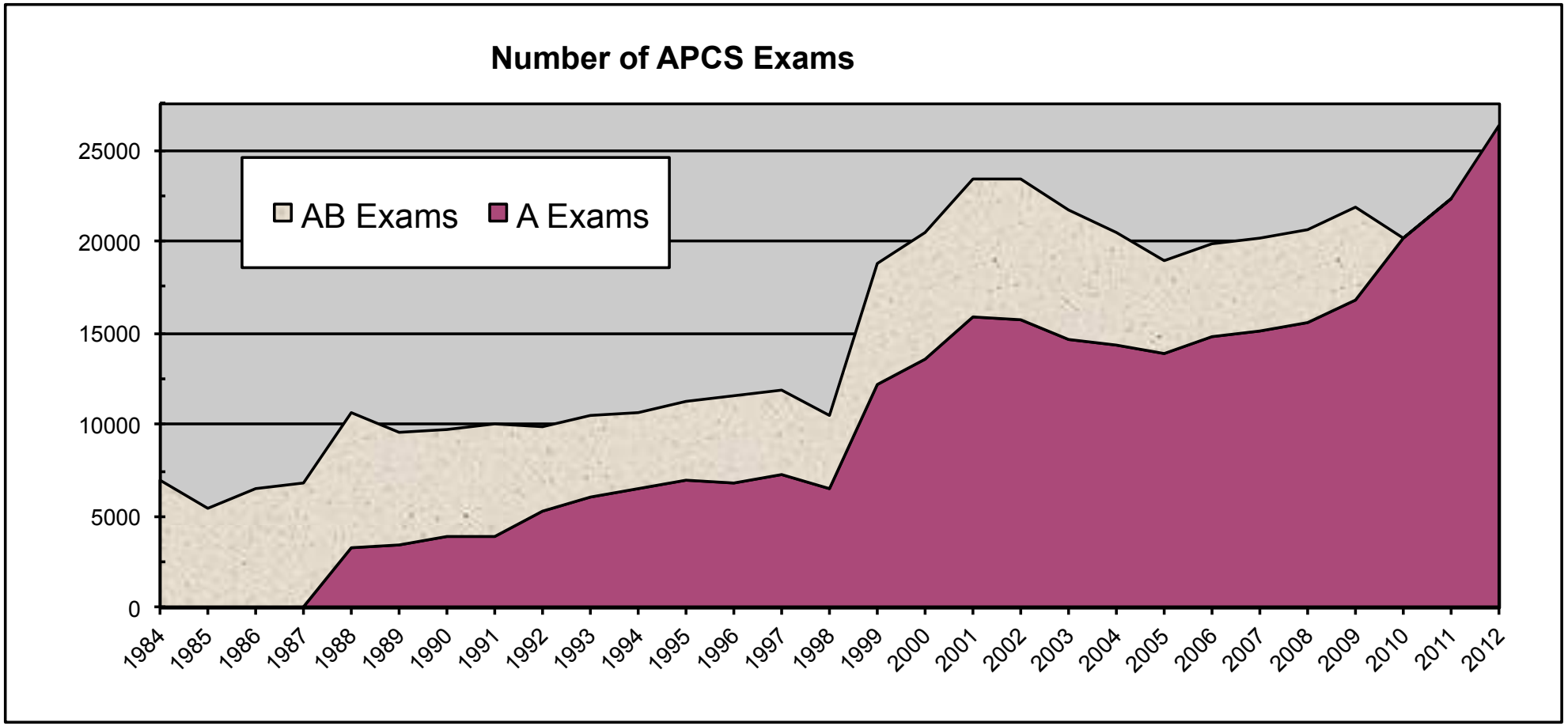
AP Computer Science A

26,350 Exams

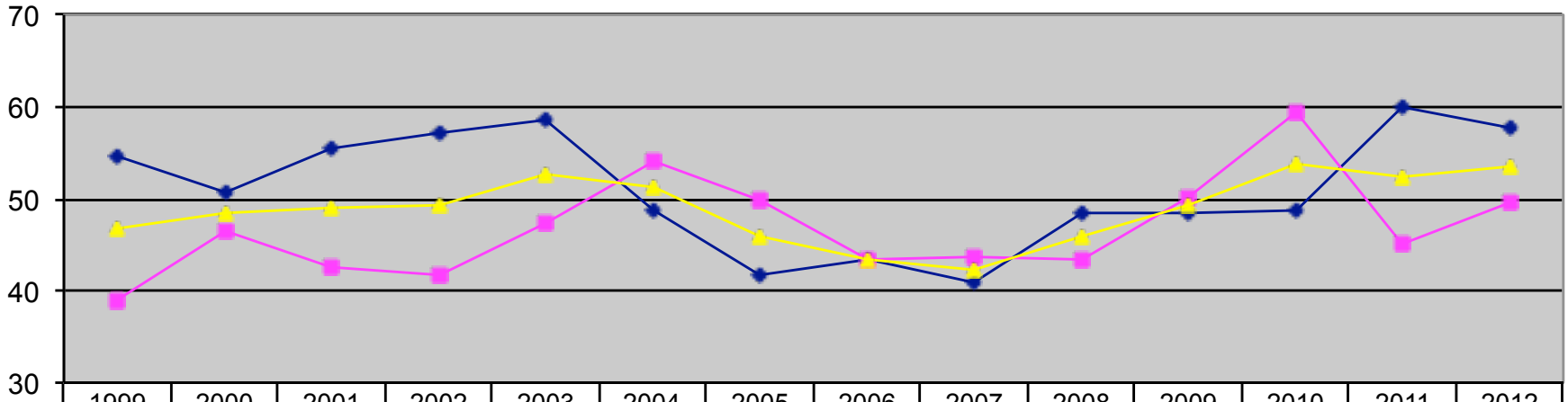


+18% from 2011 **22,250**

+30% from 2010 **20,200**


Number of APCS Exams
AB Exams
A Exams
25000
20000
15000
10000
5000
0
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012

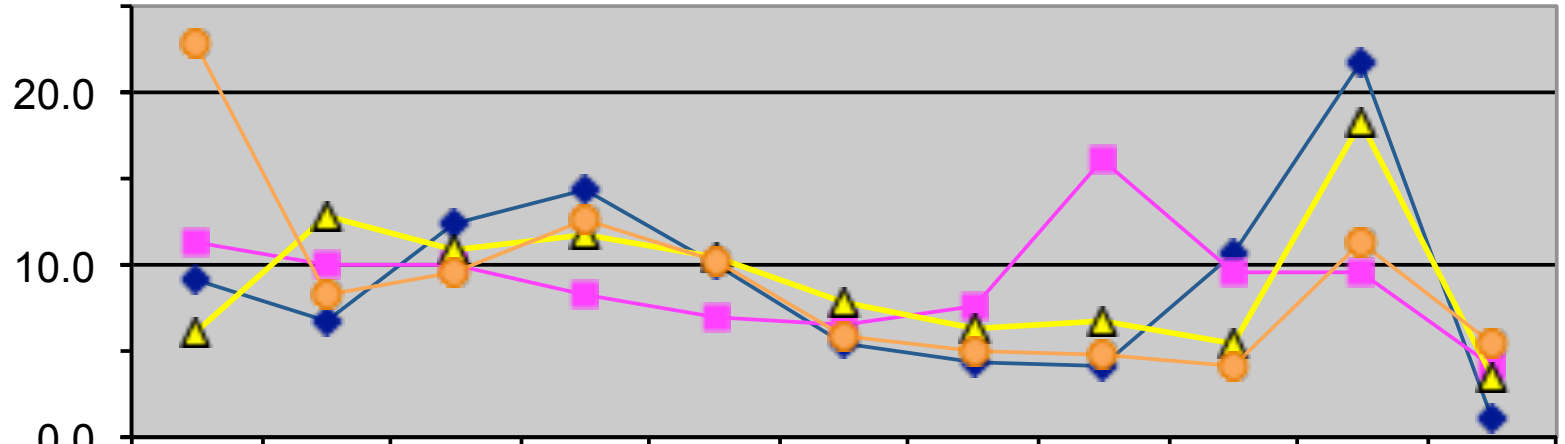
Mean as % of Max Possible



	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
Multiple Choice	54.6	50.6	55.5	57.3	58.5	48.8	41.7	43.5	40.9	48.5	48.5	48.8	59.9	57.7
Free Response	39	46.4	42.6	41.6	47.4	54	49.9	43.3	43.8	43.5	50.2	59.4	45.0	49.5
Composite	46.7	48.5	48.9	49.3	52.8	51.3	45.8	43.5	42.2	45.9	49.2	53.9	52.4	53.6

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

array array access & modification *bounds*

length 2D arrays *ArrayList*

ArrayList manipulation

for, while, loop-based iteration

enhanced for method overriding *OOP*

String comparison *compareTo*

instance variables class design

method invocation *object reference*

compound boolean expressions

not just Critter case-study breadth

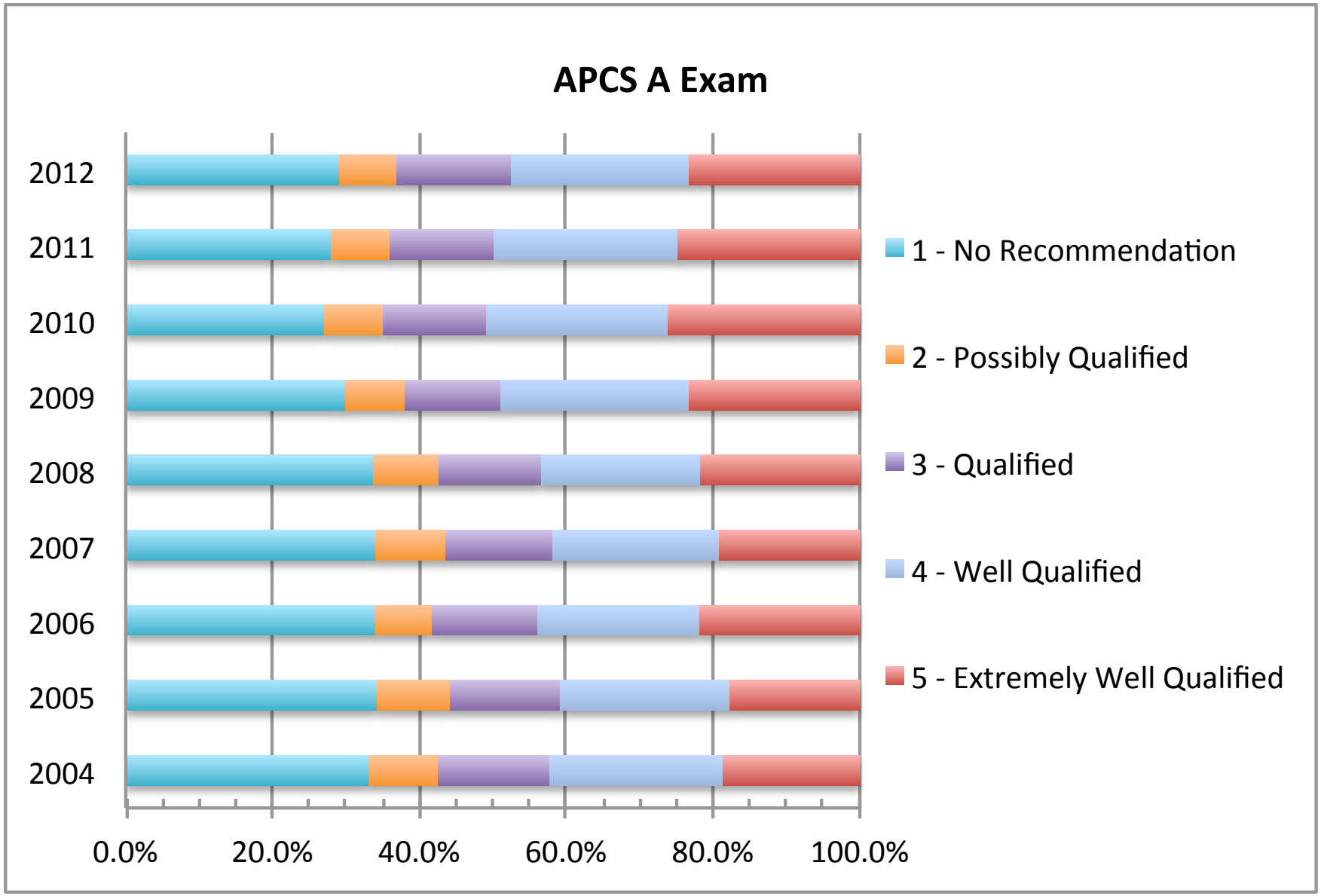
null ***null***

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

array array access & modification *bounds*

length 2D arrays *ArrayList*

ArrayList manipulation

for, while, loop-based iteration

enhanced for method overriding *OOP*

String comparison *compareTo*

instance variables class design

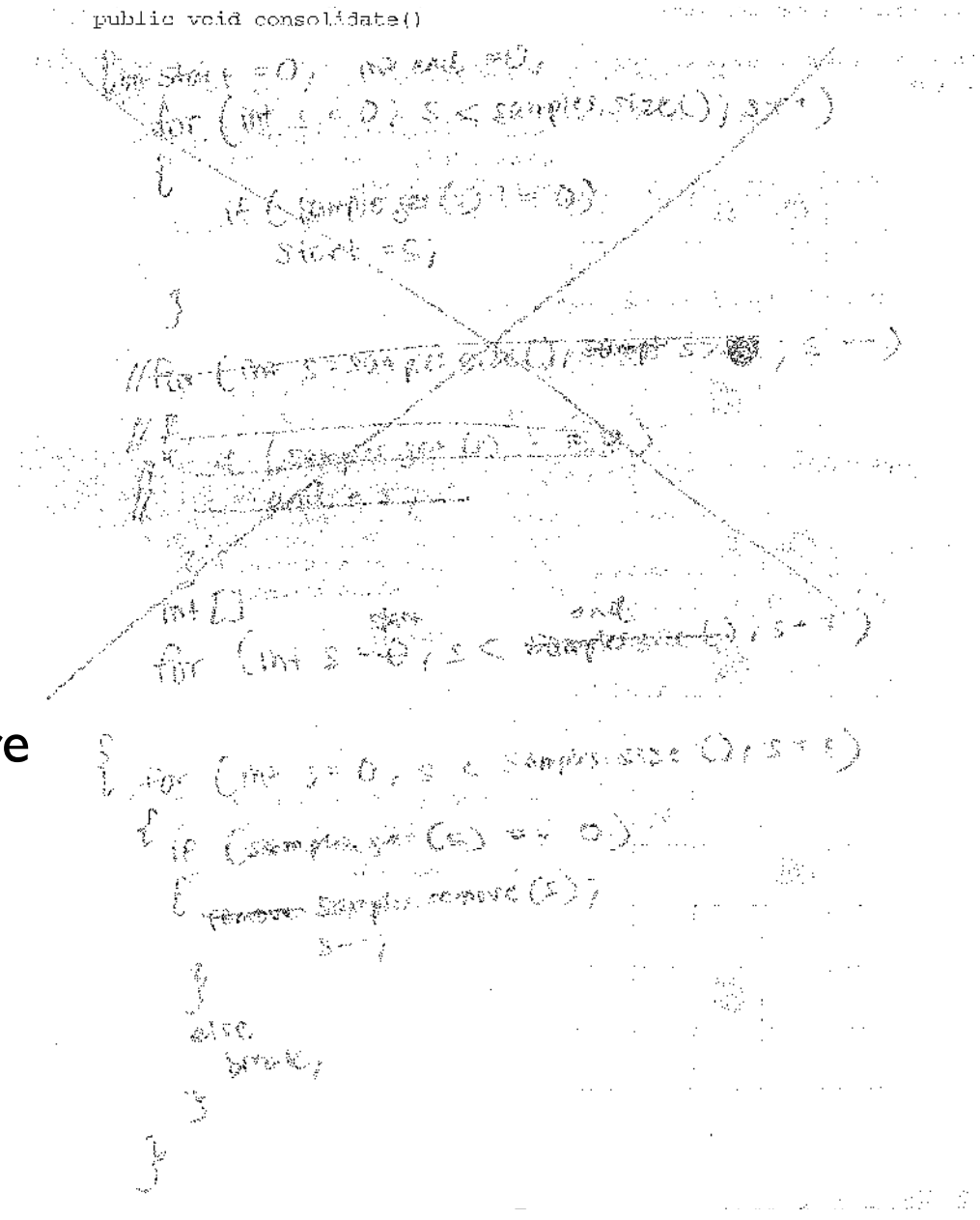
method invocation *object reference*

compound boolean expressions

not just Critter case-study breadth

null ***null***

Recommendations
Cross-out to “erase”



Start reading here

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

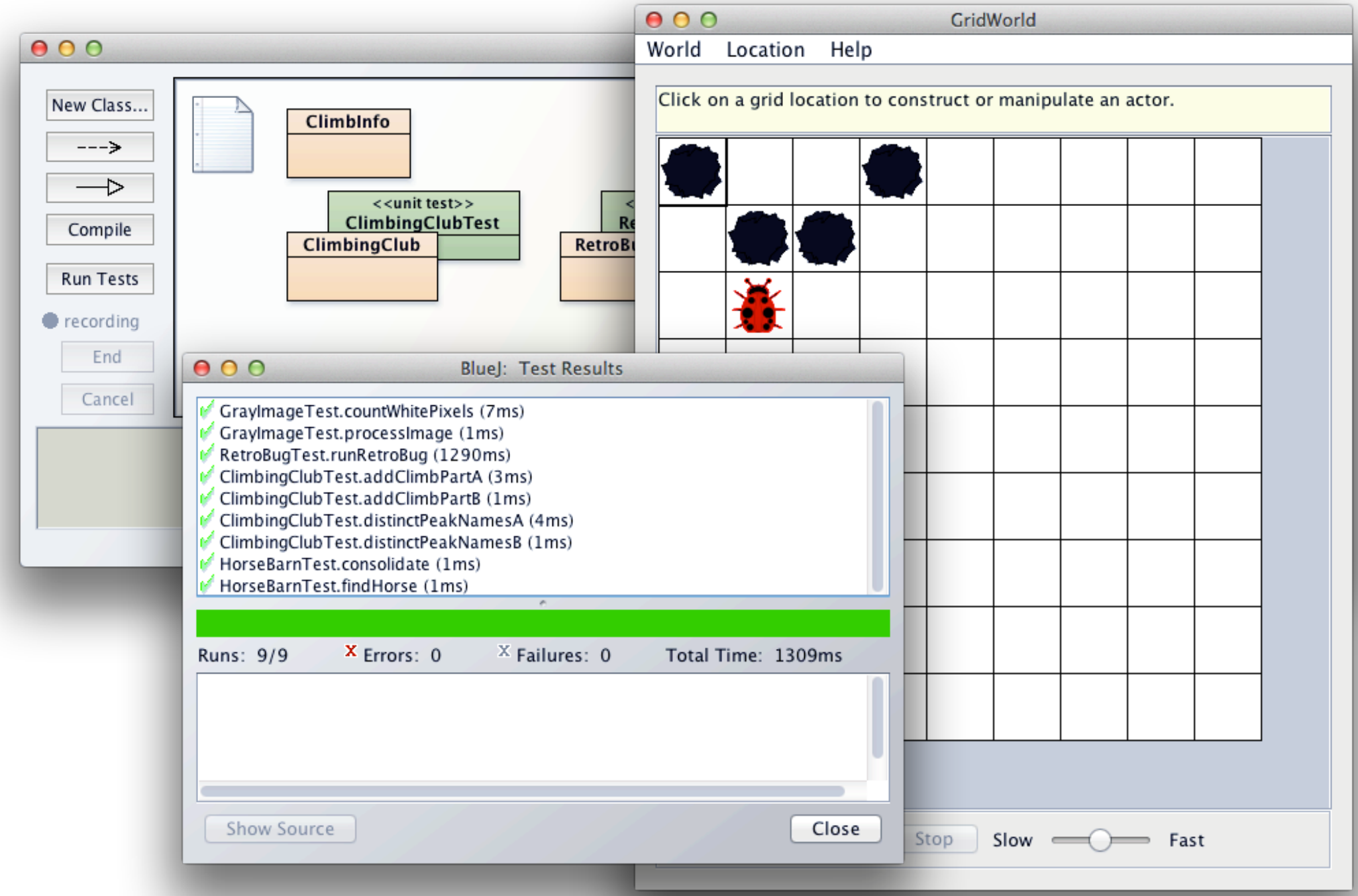
Execute solution code

All “solutions” are verifiable!

Applies to MC & FR questions

Write test harnesses as necessary

Recommendations



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