APCS in Java:
Looking Back / Looking Ahead

David Reed
Chief Reader of AP Computer Science

Department of Computer Science
Creighton University
davereed@creighton.edu

Overview

- APCS Facts You May Not Know
- APCS in Java
- grading process
- lessons learned so far
- possible points of emphasis
- upcoming changes to APCS
- questions/discussion

2

Facts You May Not Know

the Advanced Placement (AP) Program® is a cooperative effort between secondary schools and colleges and universities

- 35 courses in 20 subject areas, including AP Computer Science A & AB

in 2005, 1,221,016 students took 2,105,803 AP exams

- research has shown that students who place out of courses in college due to AP credit subsequently perform better than peers who don't

more than 60% of U.S. high schools offer AP courses

- more than 115,000 high school teachers

more than 90% of U.S. colleges and universities have published AP credit/placement policies

- APCS A: 88% APCS AB: 87%

3

APCS Exam

1984: first APCS exam, in Pascal
(6,911 exams)

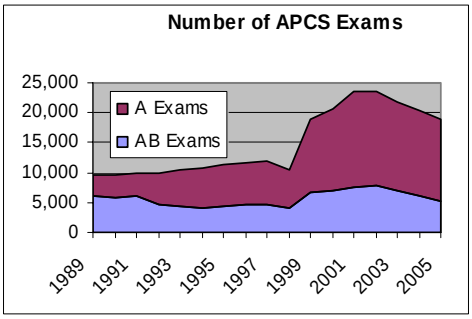
1992: split into separate A and AB exams
(5,230 + 4,643 = 9,873 exams)

1995: first case study introduced
(6,919 + 4,362 = 11,281 exams)

1999: exam language switched to C++
(12,218 + 6,619 = 18,837 exams)

2004: exam language switched to Java
(14,337 + 6.077 = 20,414 exams)

2005: 2nd year of Java
(13,924 + 5,097 = 19,021 exams)



4

College/High School Partnership

the AP program is a cooperative effort by colleges and high schools

- periodically, college faculty are surveyed regarding course content
- some exam content is tested in college classrooms to ensure that AP scores equate to college-level performance

Development Committee (3 college + 3 high school faculty) is responsible for curriculum development and exam writing

Scot Drysdale, Dartmouth (NH)
Cay Horstmann, San Jose State (CA)
Laurie White, Mercer (GA)
Don Allen, Troy High School (CA)
Reg Hahne, Atholton High School (MD)
Ann Shen, Bishop Strachan (Toronto)

Chief Reader is responsible for grading the exams and setting cutoffs

David Reed, Creighton (NE)

5

The Java Switch

in 1999-2000, an AP Ad Hoc committee surveyed colleges and universities, and recommended:

- a move to full object-oriented programming, as opposed to object-based
- a switch to (a manageable subset of) Java
  - *key advantages of Java:* safety, simplicity, better OO

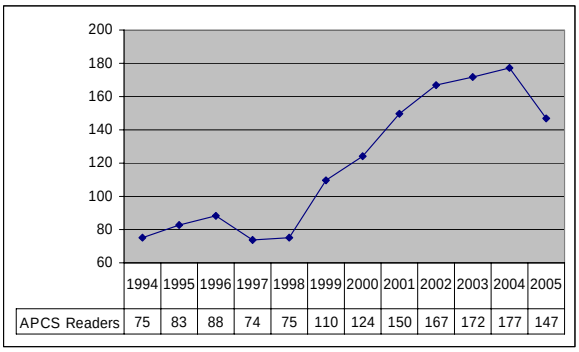
from 2000-2003, the APCS Development Committee:

- identified the Java subset to be tested
- developed the APCS curricula (A and AB courses)
- wrote the APCS course descriptions, sample tests, teacher's guide
- oversaw the development of the Java Marine Biology Case Study (MBCS)
  - including a 100+ page narrative that addresses design choices and implementation details

6

Performance Assessment

- multiple choice questions (40 per exam) are graded by computer
- free response questions (4 per exam) are graded by high school and college faculty in June at Clemson, SC
  - 2005: 111 readers, 17 table leaders, 16 question leaders, 2 exam leaders, CR



	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
APCS Readers	75	83	88	74	75	110	124	150	167	172	177	147

roughly a 45/55 college-to-high school ratio

the reading is a great professional development opportunity (and lots of fun!)

7

Reading Life

apply online at:

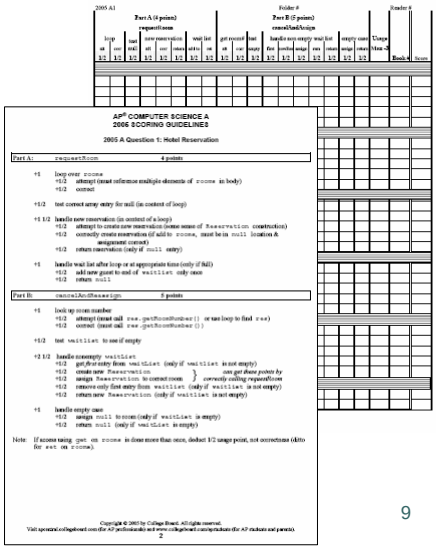
www.ets.org/reader/ap

Reading Process

- the Chief Reader develops the initial scoring rubrics
- Question Leaders refine the rubrics & train the readers
- Table Leaders mentor readers and help in applying the rubric

- a variety of consistency checks are built into the process to provide support for readers
  - buddy system, backreading, reader stats, reliability studies

CS is one of the top AP subjects in terms of reader reliability and consistency



AP® COMPUTER SCIENCE A
2005 SCORING GUIDELINES

2005 A Question 1: Hotel Reservation

Grade Setting

the Chief Reader is responsible for assigning final scores (1 – 5)

a score of 5 on the exam is meant to equate to *average* A-level performance in college
a score of 4 on the exam is meant to equate to *average* B-level performance in college
a score of 3 on the exam is meant to equate to *average* C-level performance in college
a score of 2 on the exam is meant to equate to *average* D-level performance in college
a score of 1 on the exam is meant to equate to F-level performance in college

- periodically, comparability studies are performed to equate scores with college-level performance
  - representative colleges administer sections of the exam in actual courses
  - they report each student's grade on the exam, and his/her final course grade
  - the correlation of exam score to college grade contributes to AP score setting

- in addition, some multiple choice questions are reused each year to allow for scoring consistency between exams

Student Performance with Java

the switch to Java in 2004 went fairly smoothly

- performance was comparable to previous C++ exams (2004 AB was "harder")
- *careful:* 2003-2004 comparisons are problematic due to score "realignment"

Grade	APCS A Exams				APCS AB Exams			
	2002	2003	2004	2005	2002	2003	2004	2005
5 (Extremely well qualified)	19.3%	17.0%	18.6%	17.7%	34.2%	37.6%	27.1%	31.0%
4 (Well qualified)	25.3%	24.3%	23.6%	23.1%	12.5%	13.8%	18.2%	19.6%
3 (Qualified)	18.3%	19.8%	15.2%	15.0%	26.5%	24.5%	17.6%	18.2%
2 (Possibly qualified)	8.8%	9.3%	9.5%	10.0%	10.6%	10.1%	12.1%	10.3%
1 (No recommendation)	28.4%	29.8%	33.1%	34.2%	16.2%	14.0%	25.0%	20.9%

Free Response Summaries

2004 A1: Word List
- abstraction, ArrayList manipulation

2004 A2: Pet Parade (Design)
- design/implement a class hierarchy

2004 A3: Pond Stocker (MBCS)
- work with MBCS classes, Environment

2004 A4: Robot Cleaner
- algorithmic, array manipulation

2004 AB1: Library Items (Design)
- design/implement interface & class

2004 AB2: Approval Voting
- abstraction, Set & Map manip., big Oh

2004 AB3: Predator Fish (MBCS)
- work with MBCS classes, inheritance

2004 AB4: Priority Queue
- algorithmic, recursion, BST traversal

2005 A1: Hotel Reservation
- abstraction, array & ArrayList manip.

2005 A2: Ticket Sales (Design)
- design/implement a class hierarchy

2005 A3: ZigZag Fish (MBCS)
- work with MBCS classes, inheritance

2005 A4: Improving Grades
- algorithmic, array manipulation

2005 AB1: Salmon (MBCS)
- work with MBCS classes, inheritance

2005 AB2: Postal Codes (Design)
- design data structure, Maps, big Oh

2005 AB3: Successor Nodes
- recursion, binary tree traversal

2005 AB4: Expanded Aliases
- algorithmic, Set & Queue manipulation

12

Grading Issues with Java

all questions are designed with the APCS Java subset in mind

- however, solutions that utilize constructs/classes outside the subset are NOT penalized (unless the question specifically forbids it)
- likewise, code based on Java 5.0 has NOT been penalized

as in previous years, some minor errors are ignored when grading

e.g., missing semicolons, = instead of == , case discrepancies

e.g., length vs. length() vs. size() confusion

e.g., failure to downcast when accessing a collection

```
String customer = waitlist.get(0);   instead of
String customer = (String)waitlist.get(0));
```

AP TEACHERS: ADVISE STUDENTS TO STAY WITHIN THE SUBSET!

13

What Patterns Have We Seen?

with two years of experience with Java, we can *begin* to look for patterns in student performance

- *DISCLAIMER:* these interpretations are my own, not AP sanctioned

topic areas to consider:

- object-oriented concepts
- design
- data structures
- case study

14

OOP Emphasis

with Java, object-oriented techniques are being emphasized

- classes, interfaces & abstract classes, inheritance, polymorphism, …
- all free response questions assume basic class structure
  - must access fields, implement methods, call other methods
- some involve interacting classes & calling provided methods
  - e.g., *2005 A1: Hotel Reservations*
    given a Reservation class with black box methods and the framework of a Hotel class, implement Hotel methods
- some involve designing interfaces or classes in an inheritance hierarchy
  - e.g., *2004 A2: Pet Parade*
    given an abstract Pet class, design/implement Dog & LoudDog classes

15

Observations about OOP

students have done reasonably well on basic class concepts

however, some confusion related to inheritance, polymorphism, …

2004	mean	% 0/-	adj. mean
A1: WordList (abstr/DS)	5.84	18.1%	7.13
A2: Pets (design/OOP)	5.21	5.5%	5.51
A3: Pond (MBCS)	4.04	16.3%	4.82
A4: Robot (alg/DS)	4.38	23.6%	5.73

2005	mean	% 0/-	adj. mean
A1: Hotel (abstr/DS)	4.54	18.6%	5.58
A2: Ticket (design/OOP)	4.04	20.1%	5.05
A3: ZigZag (MBCS)	4.30	24.5%	5.69
A4: Grades (alg/DS)	5.08	17.7%	6.18

common errors:

- overriding parent class instance variables & methods
- attempting to access private data from parent class
- reimplementing functionality from a previous part

AP TEACHERS: CONTINUE TO EMPHASIZE OOP & ABSTRACTION!

16

Design

on recent exams, the development committee has placed a greater emphasis on design

- may involve designing/implementing an interface or classes in a hierarchy
  - e.g., *2004 AB1: Library Items*

    given a Book class, design a LibraryItem interface and implement a LibraryBook class; also design and analyze a Library collection

- may involve designing/implementing/analyzing a data structure
  - e.g., *2005 AB2: Postal Codes*

    given a class that maps codes to cities, define a data structure for the reverse mapping, implement methods that meet big-Oh requirements

17

Observations about Design

student performance has been mixed

- 2004 A1 (Pet Parade) had 2nd highest mean; 2005 A2 (Tickets) had lowest
- AB performance has been stronger

2004	mean	% 0/-	adj. mean
AB1: Library (design/OOP)	5.23	4.8%	5.50
AB2: Voting (abstr/DS)	4.05	16.9%	4.88
AB3: Predator (MBCS)	5.60	5.7%	5.94
AB4: PQ (alg/DS)	3.65	20.8%	4.61

2005	mean	% 0/-	adj. mean
AB1: Salmon (MBCS)	6.39	3.3%	6.61
AB2: Postal (design/DS)	5.05	12.9%	5.80
AB3: TreeNode (rec/DS)	3.13	16.0%	3.73
AB4: Aliases (alg/DS)	5.54	13.4%	6.40

common errors:

- inheritance confusion
- big Oh analysis

AP TEACHERS: BE AWARE OF "DESIGN" IN ITS VARIOUS FORMS!

18

Data Structures

the APCS Java subset includes a variety of data structures

- A: array, ArrayList
- AB: array, ArrayList, LinkedList, List, HashSet, TreeSet, Set, HashMap, TreeMap, Map, Stack, Queue, PriorityQueue, heap ListNode, TreeNode

- may involve traversing/accessing a provided structure
  - e.g., *2005 A1: Hotel Reservations*

    manipulate an ArrayList of Reservations, and an array of names on a waiting list

- may involve designing and/or analyzing a data structure
  - e.g., *2004 AB2: Approval Voting*

    implement methods using TreeSet/HashSet & TreeMap/HashMap, analyze performance based on structures selected

19

Observations about Data Structures

on A exams, student performance has been good

- note: Java arrays & ArrayLists are similar to C++ arrays & vectors

2004 AB performance was weak

- possibly taken by surprise by the number of Collection classes
- performance improved in 2005, except on difficult tree recursion problem

2005	mean	% 0/-	adj. mean
AB1: Salmon (MBCS)	6.39	3.3%	6.61
AB2: Postal (design/DS)	5.05	12.9%	5.80
AB3: TreeNode (rec/DS)	3.13	16.0%	3.73
AB4: Aliases (alg/DS)	5.54	13.4%	6.40

common errors:

- confused access on arrays and ArrayLists
- not knowing access efficiency (e.g., HashSet vs. TreeSet)

AB TEACHERS: DON'T FORGET LINKED STRUCTURES & RECURSION!

20

Java Marine Biology Case Study

each exam has one free response question related to the case study

- may involve extending the Fish class to exhibit new behaviors
  - e.g., *2005 AB1: Salmon*

    a Salmon has additional state for its age and home location, after a certain age it will move back to home then breed & die

- may involve working with other MBCS classes
  - e.g., *2004 A3: Pond Stocker*

    completed methods of a class that maintained a minimum number of Fish in the environment

Observations about MBCS

student performance has been excellent overall, but lots of blanks on A

- suggests many A students do not know the MBCS, but those that do are fine

2005	mean	% 0/-	adj. mean
A1: Hotel (abstr/DS)	4.54	18.6%	5.58
A2: Ticket (design/OOP)	4.04	20.1%	5.05
A3: ZigZag (MBCS)	4.30	24.5%	5.69
A4: Grades (alg/DS)	5.08	17.7%	6.18

2005	mean	% 0/-	adj. mean
AB1: Salmon (MBCS)	6.39	3.3%	6.61
AB2: Postal (design/DS)	5.05	12.9%	5.80
AB3: TreeNode (rec/DS)	3.13	16.0%	3.73
AB4: Aliases (alg/DS)	5.54	13.4%	6.40

common errors:

- inheritance confusion
- unfamiliar with MBCS classes

AP TEACHERS: BE SURE YOUR STUDENTS KNOW THE MBCS!

Points of Emphasis

continue to emphasize OOP and abstraction
- inheritance, polymorphism, private vs. public

be aware that "design" will be tested
- could involve designing a class or data structure
- could also involve analyzing performance and tradeoffs

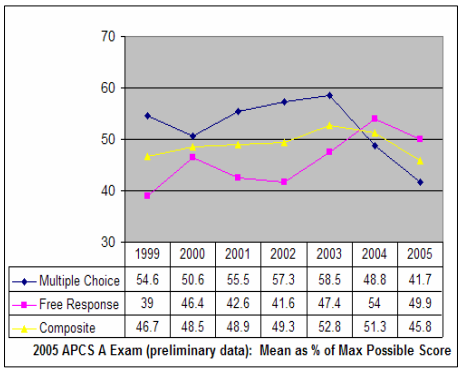
cover the full range of data structures
- A: array, ArrayList
- AB: array, ArrayList, LinkedList, TreeSet, HashSet, TreeMap, HashMap, …

be sure your students know the MBCS
- may be called upon to design/modify a Fish
- could also involve other MBCS classes, e.g., Location, Environment

23

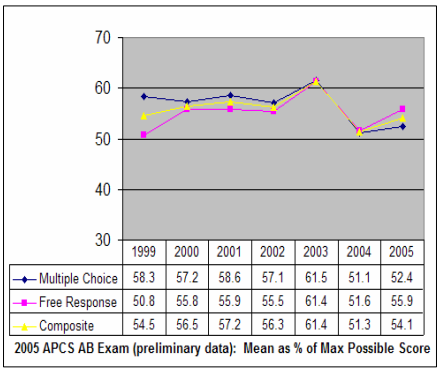
Multiple Choice vs. Free Response

goal: means of multiple choice and free response to be 50% of max
- all Java exams have been well within expectations
- interestingly, free response had higher score percentages in both years



	1999	2000	2001	2002	2003	2004	2005
Multiple Choice	54.6	50.6	55.5	57.3	58.5	48.8	41.7
Free Response	39	46.4	42.6	41.6	47.4	54	49.9
Composite	46.7	48.5	48.9	49.3	52.8	51.3	45.8

2005 APCS A Exam (preliminary data): Mean as % of Max Possible Score



	1999	2000	2001	2002	2003	2004	2005
Multiple Choice	58.3	57.2	58.6	57.1	61.5	51.1	52.4
Free Response	50.8	55.8	55.9	55.5	61.4	51.6	55.9
Composite	54.5	56.5	57.2	56.3	61.4	51.3	54.1

2005 APCS AB Exam (preliminary data): Mean as % of Max Possible Score

Looking Ahead

- college survey (2006-2007)
  - a survey is planned to assess what colleges are teaching and what they demand of AP students
  - potentially, could affect the required topics in the APCS curriculum
- incorporation of Java 5.0 (starting with the 2007 exam)
  - some (but not all) new features are being integrated into the APCS subset
- new case study (starting with the 2008 exam)
  - currently under development, the new case study will have similarities to the MBCS, but much shorter & simpler

25

Java 5.0 for the 2007 exam

Java 5.0 introduced several new features, including

- generics
- autoboxing/unboxing
- typesafe enumerations
- enhanced for loop
- Scanner class
- methods with variable arguments

the committee is taking a conservative approach to adopting features

- is the benefit worth requiring all AP teachers to include it?
- will the new feature make it easier to ask good exam questions?
- note: the APCS Java subset merely defines those features that are testable
  - a teacher can still teach features not included in the APCS Java subset
  - a student can use features from outside the subset on free response questions

26

IN: Generic Collections

in Java 1.4, collection classes (e.g., ArrayList, Set) hold objects

```
Set names = new TreeSet();
…
Iterator iter = names.iterator();
while (iter.hasNext()) {
    System.out.println(((String)iter.next()).toUpperCase());
}
```

Java 5.0 introduced generic collection (similar to C++ templates)

- less casting required
- more transparent data structure, compile-time type checking

```
Set<String> names = new TreeSet<String>();
…
Iterator<String> iter = names.iterator();
while (iter.hasNext()) {
    System.out.println(iter.next().toUpperCase());
}
```

27

IN: Generic Collections

generic collection classes also lead to better exam questions

- e.g., consider part of a class from the 2005 AB exam

```
public class PostalCodeDB
{
  private Map codeToCityMap; // each key is a postal code, its
                             // associated value is a set of cities
  public PostalCodeDB()
  {
    codeToCityMap = new HashMap();
  }
```

- data structures can be stated less ambiguously using generics

```
public class PostalCodeDB
{
  private Map<String, Set<String>> codeToCityMap;

  public PostalCodeDB()
  {
    codeToCityMap = new HashMap<String, Set<String>>();
  }
```

28

OUT: Writing Generic Classes

students must be able to use generic classes, but not write them

- writing generic classes can get very messy very quickly

```
public class TreeNode<E> { . . . }
```

vs.

```
public class TreeNode<E extends Comparable<E>> { . . . }
```

vs.

```
public class TreeNode<E extends Comparable<? super E>> { . . . }
```

it was decided that having to explain the intricacies of type parameters and wild cards was not worth it (at least for the exam)

- `ListNode` and `TreeNode` will remain as they are, and not be generic
- the `Comparable` interface will be used, and not the generic `Comparable<E>`

29

IN: Enhanced For Loop

in Java 1.4, iterators were required to step through a collection

```
Set names = new TreeSet();
…
Iterator iter = names.iterator();
while (iter.hasNext()) {
    System.out.println(((String)iter.next()).toUpperCase());
}
```

Java 5.0 introduced the enhanced for loop

- cleaner, more abstract notation for accessing each element in a collection
- reduces the need for iterators*

```
Set<String> names = new TreeSet<String>();
…
for (String nextWord : names) {
    System.out.println(nextWord.toUpperCase());
}
```

* iterators are still needed in some circumstances, e.g., if want to remove elements

30

IN: Stack, Queue, PriorityQueue

starting in 2007, the exam will assume standard Java classes

- will no longer have AP-specific interfaces for Stack, Queue, and PriorityQueue

  class java.util.Stack<E>: push, pop, peek, isEmpty

```
Stack<String> stk = new Stack<String>();
```

  interface java.util.Queue<E>: add, remove, peek, isEmpty

```
Queue<String> q = new LinkedList<String>();
```

  class java.util.PriorityQueue<E>: add, remove, peek, isEmpty

```
PriorityQueue<String> pq = new PriorityQueue<String>();
```

- *advantage:* more standard treatment of the structures
- *disadvantage:* must limit use to the standard methods

31

OUT: Autoboxing/unboxing

after much debate, it was decided not to include autoboxing/unboxing

- it allows for primitives to be transparently stored in collections

```
Map<String, Integer> wordCount = new HashMap<String, Integer>();
wordCount.put("foo", new Integer(0));
. . .
int count = wordCount.get("foo").intValue();
wordCount.put("foo", new Integer(count + 1));
```

- with autoboxing/unboxing:

```
Map<String, Integer> wordCount = new HashMap<String, Integer>();
wordCount.put("foo", 0);
. . .
wordCount.put("foo", wordCount.get("foo")+1);
```

- while this seems like a clear win, there are many subtle conversion rules
  - examples that require autoboxing/unboxing can easily be avoided in the exam
  - since not needed for the exam, it will not be required

32

OUT: Other Java 5.0 Additions

other features, while potentially useful, are not central to APCS

- Scanner class, printf (note: APCS does not prescribe any input methodology)

```
Scanner input = new Scanner(System.in);
while (input.hasNext()) {
    String nextWord = input.next();
    …
}
```

- type-safe enumerations

```
public enum Response { YES, NO, MAYBE, UNLIKELY, PROBABLY };
```

- methods with variable arguments

```
public static double average(double... values)
{
    double sum = 0;
    for (double v : values) sum += v;
    return sum / values.length;
}
```

- static imports, annotations, …

Again: OUT != BAD

just because a feature is not included in the APCS subset, doesn't mean it's a bad feature or that teachers shouldn't cover it

- it simply means that those features will not be tested on the exam
- it is expected that teachers will cover some of these features
  e.g., Scanner, printf, simple autoboxing/unboxing
- students are free to use any of these feature in writing free response solutions
  caveat: readers are human, so esoteric code runs a risk

some features (e.g., autoboxing/unboxing) may be revisited by the Development Committee as college practices become more uniform

New Case Study

a new case study is currently under development

- to make the transition easier for teachers, it will have similarities with the MBCS
- however, it will be MUCH simpler and more flexible

- the GridWorld framework can be used for fish-like activities
  - e.g., can create critters that move & interact on a 2D Grid, use inheritance to build complex behavior, etc.
- but will also be useful in a wide variety of applications
  - e.g., could use the generic Grid to store other types of things, such as colored tiles in a game board

code framework + short narrative will be available in summer 2006

- will not be on AP exams until 2008 → a year for teachers to play before teaching

35

Conclusions

- computer science is an incredibly dynamic field
  - college curricula are constantly being revised & updated
  - by definition, the AP program must follow the lead of colleges
- object-oriented programming is the dominant model, at least for the near future
  - Java is the language of choice, with many attractive features
  - more importantly for APCS, Java is the language most accepted by colleges
- expect continued emphasis on:
  - interfaces & class design, inheritance, abstraction, use of collections, …
- the case study will continue to be a tool for:
  - illustrating important ideas, ensuring exposure to complex software, and introducing deeper questions on the exam

36

Challenge to Teachers

the APCS curricula, both A & AB, are very ambitious

- almost all topics from the pre-OOP days are still there
- on top, we have added polymorphism, inheritance, interfaces, collections, …

how does a teacher squeeze all of this new material in, and still insure that students master the basics and are competent problem solvers?

- development/teaching tools? (e.g., BlueJ, KarelJ Robot, Jeliot, Alice, …)
- active learning? pair programming?
- Herculean effort?

note: colleges face the same challenge

- however, we have the flexibility to customize our courses, skip what we want
- AP teachers are bound to the provided curriculum (but usually have more time)

37

FYI

- http://apcentral.collegeboard.com
  AP Central: AP info, course descriptions, reference materials, …
- http://apcentral.collegeboard.com/program/research
  AP Research and Data: exam data, research studies, …
- http://www.collegeboard.com
  College Board: general info about the association, AP program
- http://cs.colgate.edu/APCS/Java/APCSJavaMaterials.html
  Unofficial APCS site, by Chris Nevison (former Chief Reader)

38