Growl…Roar…Are We Ready for Tiger? Review of the Current Climate and Changes to be Implemented for the 2007 AP CS Exam

Don Allen, Moderator, Troy High School

Reg Hahne, Marriotts Ridge High School

Cay Horstmann, San Jose State University

David Reed, Creighton University

Agenda

- Current State of APCS
  - Dave Reed
  - Creighton University
- APCS and Java 5
  - Reg Hahne
  - Marriotts Ridge High School
- New Case Study
  - Cay Horstmann
  - San Jose State University

Current State of APCS

David Reed

Creighton University

davereed@creighton.edu

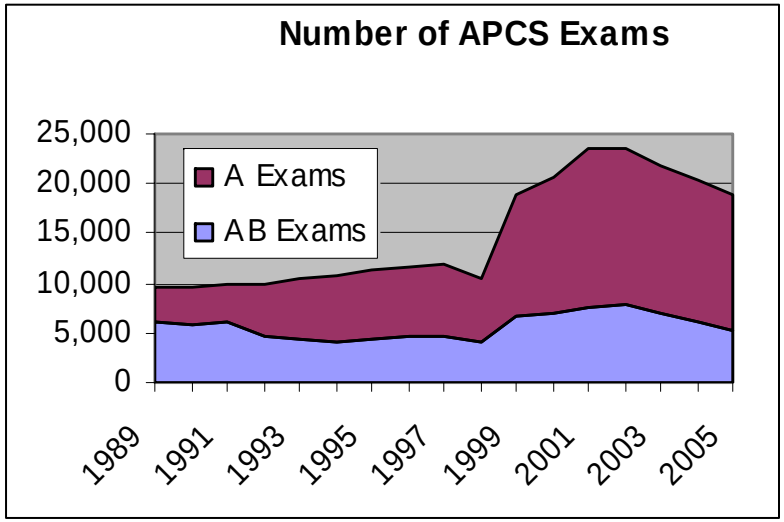
Chief Reader for AP CS

APCS Exam Count

2005: 13,924 A + 5,097 AB = 19,021 exams

2004: 14,337 A + 6,077 AB = 20,414 exams

note: not as dramatic a drop as most colleges have experienced, but still cause for some concern



147 readers graded the exams during a week in June

- roughly 45/55 college-to-high school ratio
- 2005 introduced table leaders to the reading hierarchy
  1 CR, 2 ELs, 16 QLs, 17 TLs, 111 Readers

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 is NOT penalized

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

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

e.g., no penalty if fail to downcast when accessing a collection

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

TEACHERS: ADVISE STUDENTS TO STAY WITHIN THE SUBSET!

OOP emphasis

with Java, object-oriented techniques are emphasized

- all problems utilized class design and/or implementation
- most problems utilized Java collections, class/method use
- A2 (Ticket Design), A3 (ZigZag Fish), AB1 (Salmon) utilized inheritance

students did reasonably well, with some confusion on OOP concepts

- common error: not recognizing when inherited data/methods could be used

  e.g., overriding parent class instance variables & methods

  e.g., attempting to access private data from parent class instead of calling super
- common error: attempting to reimplement existing functionality

  e.g., given `minNode` method on tree, but try to implement from scratch

TEACHERS: CONTINUE TO EMPHASIZE OOP & ABSTRACTION!

Design Questions

each exam included a question involving class/data structure design

- A2 (Ticket Design) involved designing/implementing classes in a hierarchy
- AB2 (Postal Codes) involved designing/analyzing/using a data structure

student performance was mixed

- A2 was lowest mean, despite similarity to 2004 design question
- AB2 absolute mean was reasonable, especially given its open-endedness

	mean score	% of 0/-	mean w/o 0/-
A1 (Hotel Res.)	4.54	18.6%	5.58
A2 (Ticket Design)	4.04	20.1%	5.05
A3 (ZigZag Fish)	4.30	24.5%	5.69
A4 (Impr. Grades)	5.08	17.7%	6.18

	mean score	% of 0/-	mean w/o 0/-
AB1 (Salmon)	6.39	3.3%	6.61
AB2 (Postal Code)	5.05	12.9%	5.80
AB3 (Succ Nodes)	3.13	16.0%	3.73
AB4 (Exp. Aliases)	5.54	13.4%	6.40

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

Java Collections

Java Collections are used extensively

- A1: array, ArrayList
- A4: array
- AB2: Set, Map
- AB3: binary search tree, recursion
- AB4: Set, Map, Queue

	mean score	% of 0/-	mean w/o 0/-
AB1 (Salmon)	6.39	3.3%	6.61
AB2 (Postal Code)	5.05	12.9%	5.80
AB3 (Succ Nodes)	3.13	16.0%	3.73
AB4 (Aliases)	5.54	13.4%	6.40

students knew Collection classes much better

- common error on A: confused access on arrays and ArrayLists
- common error on AB: not knowing access efficiency (HashSet vs. TreeSet)

but AB students had MAJOR troubles with AB3

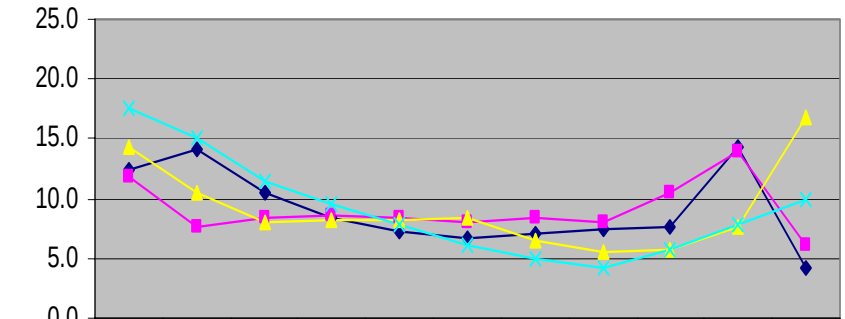
- typically, tree traversal and recursion are difficult topics on the exam
- this year was especially difficult – extra links introduced, nontrivial algorithm

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

Java Marine Biology Case Study

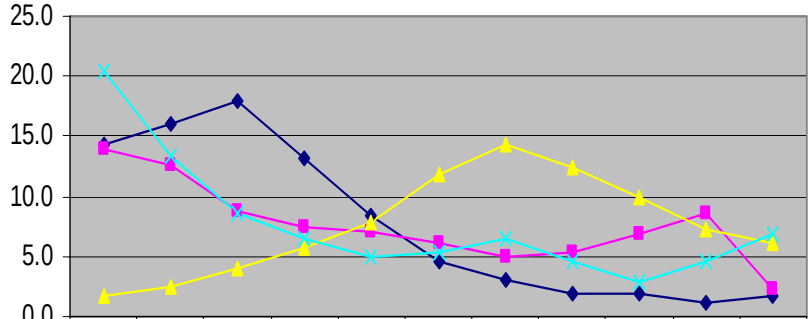
student performance on MBCS questions was excellent overall

- A3 (ZigZag) had MANY blanks, but 2nd highest mean if throw these out
  - suggests many A students did not know MBCS at all, but those that did were fine
  - knowledgeable students recognized similarity to DarterFish in the Appendix
- AB1 (Salmon) had highest mean



	9	8	7	6	5	4	3	2	1	0	-
A1	12.3	14.0	10.6	8.4	7.2	6.7	7.0	7.5	7.6	14.4	4.3
A2	11.8	7.6	8.3	8.7	8.5	8.1	8.5	8.1	10.4	14.0	6.1
A3	14.3	10.6	8.1	8.2	8.3	8.4	6.5	5.6	5.7	7.7	16.8
A4	17.6	15.0	11.5	9.5	7.7	6.1	4.9	4.1	5.8	7.7	10.0

2005 APCS A Exam: % of Exams Receiving Score



	9	8	7	6	5	4	3	2	1	0	-
AB1	14.4	15.9	17.9	13.2	8.5	4.6	3.0	1.8	1.9	1.1	1.7
AB2	13.9	12.7	8.8	7.4	7.0	6.2	5.0	5.4	6.9	8.7	2.2
AB3	1.7	2.5	4.0	5.8	7.8	11.9	14.4	12.5	10.0	7.3	6.1
AB4	20.3	13.3	8.6	6.5	4.9	5.3	6.5	4.5	2.8	4.5	6.8

2005 APCS AB Exam: % of Exams Receiving Score

'A' TEACHERS: BE SURE YOUR STUDENTS KNOW THE MBCS!

Comparison with Recent Exams

- A exam performance was comparable to 2004 & previous exams
- AB exam performance was better, closer to previous exams

	APCS A Exams				APCS AB Exams			
Grade	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%

APCS and Java 5

Reg Hahne

Marriotts Ridge High School

rhahne@hcpss.org

AP CS Development Committee Member

Why modify the AP subset ?

- support the teaching of Computer Science
- new features will make question writing easier
- new features will make question answering easier
- adopt college trends as we move to Java 5.0

Clearer simpler code that supports self-documenting entities

What you Need to Know (Testable)

- how to use generic collections such as `ArrayList<E>`
- the "for each" loop (enhanced for loop)
- the `Stack`, `Queue`, and `PriorityQueue` types from the `java.util` package

The AP interfaces (Stack, Queue, PriorityQueue) are no longer required

What you Don't Need to Know (non-Testable)

- boxing/unboxing
- `Scanner` class
- type-safe enumerations
- static imports
- variable number of arguments to methods (varargs methods)
- `printf` (formatting output)
- and many other Java 5.0 enhancements

All useful, but not testable

Generic Collections


```
ArrayList<City> cityList;
Map<Location, City> cityLocator;
```

- no guesswork as to the type of `Object` in the given collection
- you may be able to use `ArrayList` earlier in your course, as casting is eliminated

```
City firstCity = cityList.get(0);
     // no (City) cast required
```

Stick to the basics, avoid defining your own generic types

"For Each" Loop


```
City[] cities = ….;

// traditionally
for (int i = 0; i < cities.length; i++)
{
   City c = cities[i];
   System.out.println(c.getName());
}

//Java 5.0
for (City c : cities)
   System.out.println(c.getName());
```

Less room for indexing errors

"For Each" Loop (cont.)


```
// iterating through a Set
Set<city> citySet = …;
for (City c : citySet)
   System.out.println(c.getName());
```

```
// iterating through a Map
Map<Location, City> cityLocator = …;
for (Location loc : cityLocator.keySet())
{
   City c = cityLocator.get(loc);
   …;
}
```

Iterate through objects of any class that implements the `Iterable` interface

Additional Interfaces

- `Queue` interface
- `Queue` implementation

  use the `Queue` interface and instantiate it with a `LinkedList`

```
Queue<Customer> waitingCustomers =
      new LinkedList<Customer>();
```

Only the interfaces and the classes of the standard Java library will be required

Changes in Method Names

old	new
`peekTop`	
`peekFront`	`peek`
`peekMin`	
`enqueue`	`add`
`dequeue`	`remove`
`removeMin`	

Use conceptual stack, queue and priority queue data structures

Stack and PriorityQueue Usage

- use the `Stack` class provided in the Java API

```
Stack<Customer> waitingCustomers =
        new Stack<Customer>();
```

- use the `PriorityQueue` class provided in the Java API

```
PriorityQueue<Customer> waitingCustomers =
        new PriorityQueue <Customer>();
```

Use conceptual stack, queue and priority queue data structures

A word of Caution

- stacks, queues and priority queues always refer to the conceptual data structure

  e.g. don't add to the middle of a queue

- students who carry out operations in a free-response question that are incompatible with the conceptual nature of a data structure may not receive full credit

Use conceptual stack, queue and priority queue data structures

New Case Study

Cay Horstmann

San Jose State University

Cay@Horstmann.com

AP CS Development Committee Member

Lessons Learned from MBS

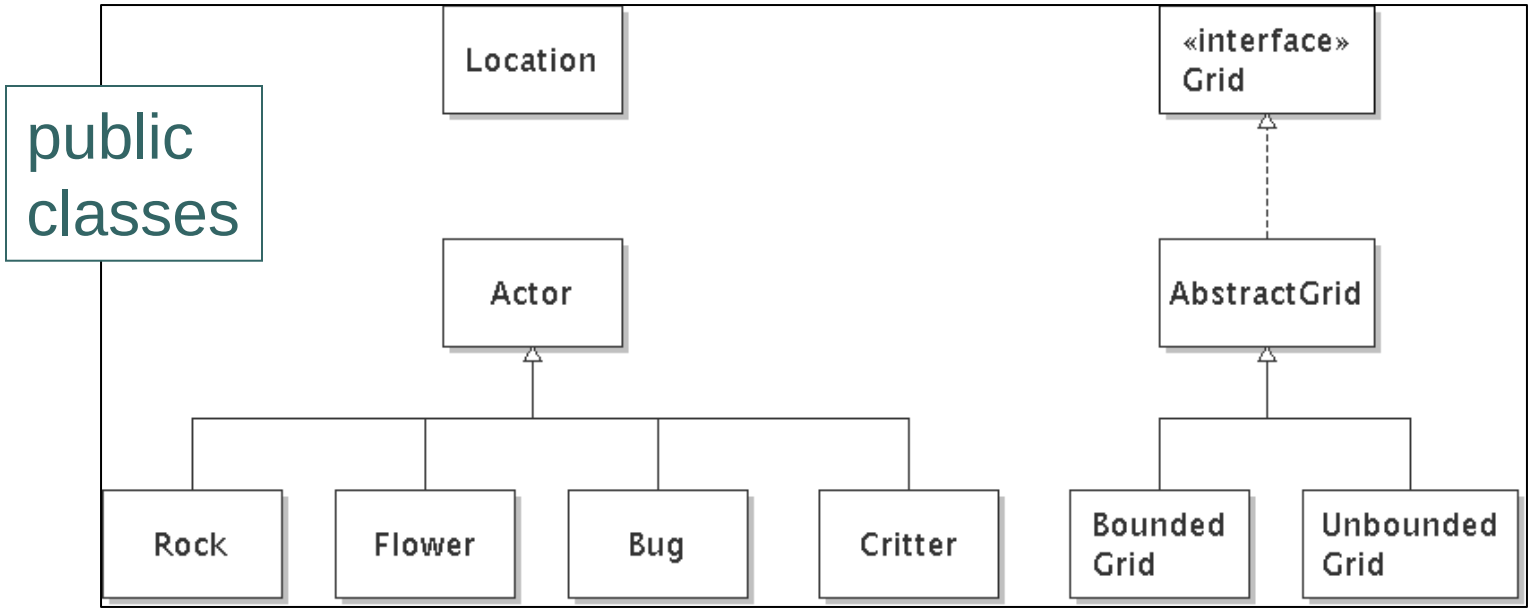
- visual framework is very motivating
- classes were flexible; supported many exam questions
- teachers took advantage of extension points
- complexity was a drawback

themes for new case study

- continuity
- simplicity
- testability
- extensibility

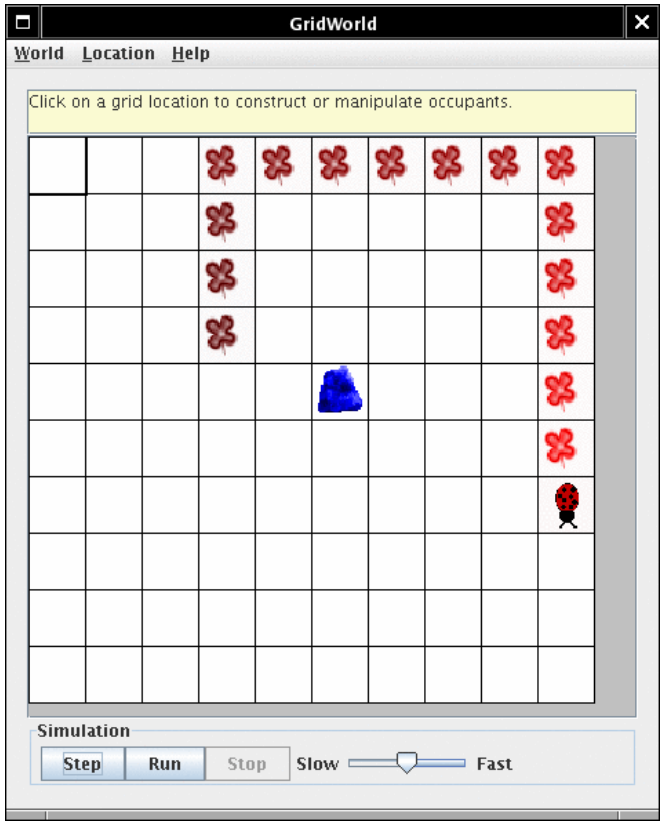
Overview

- framework derived from MBS (GNU license!)
- 10 testable classes/interfaces (MBS had 14)
- simplified API: easier to remember, easier to formulate questions
- layers: use framework at multiple points in your course
- much easier to add your own classes

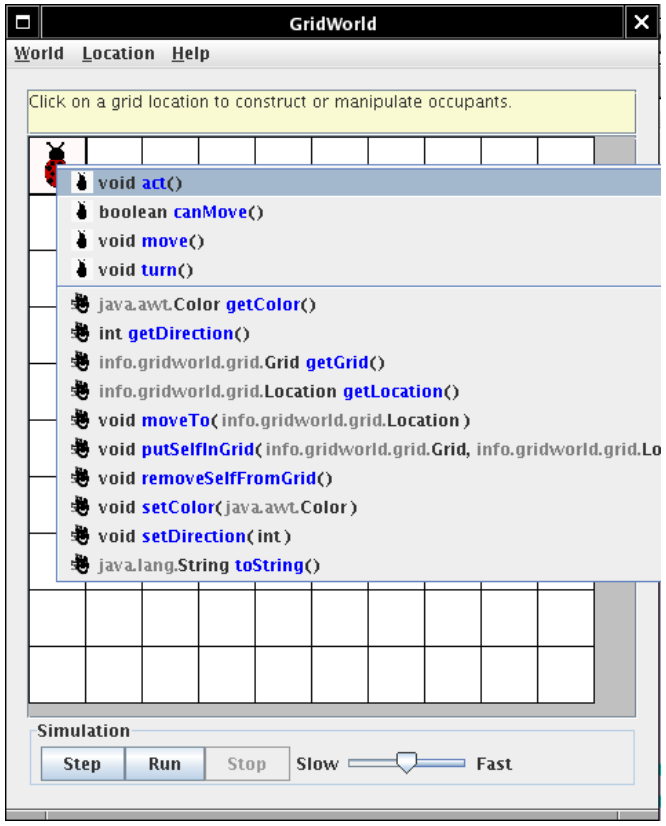


Layer 1: Objects

turtle-like graphics: bug drops flowers



direct manipulation (like in BlueJ)



Can be used as early motivation for objects

Layer 2: Inheritance

- uses template method, like `Fish.act`
- override methods to define how critters find neighbors
  - find neighbors
  - process neighbors
  - find candidates for move locations
  - select a move location
- clean and simple behavior, less randomness than MBS

Layer 3: Data Structures

- little change from MBS
- uses generics
- `Grid<E>` similar to `Map<Location, E>`
- can contain any type, not just `Locatable`
- `AbstractGrid` example of abstract class
- AB Material

Testability

- `Bug` subclasses lend themselves to multiple choice questions:

  e.g., "Which pattern does `MysteryBug` produce?"

- `Critter` subclasses are easily understood

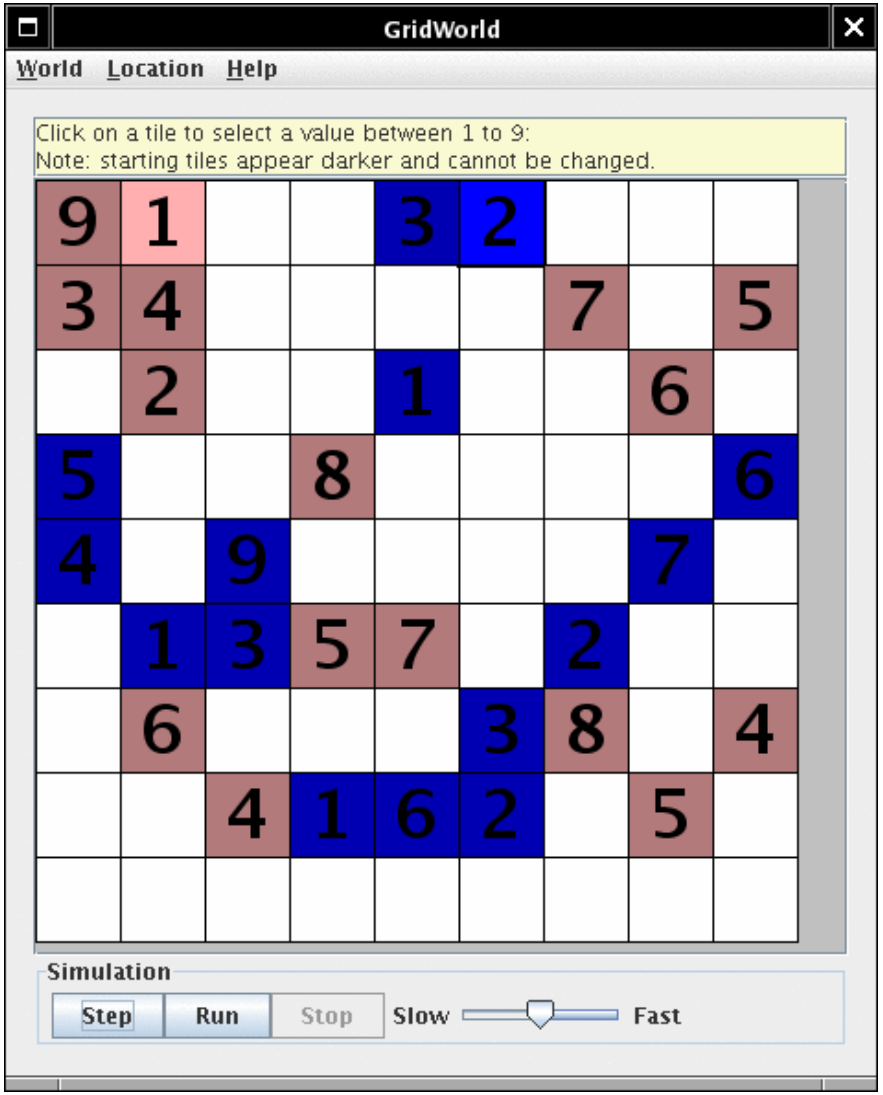
  e.g., no confusing breed/die issues

- `Grid` useful for a wide variety of questions

Extensibility

- to add new actor, simply supply subclass and GIF image
- image is colored and rotated automatically
- easy to do game worlds (e.g. Memory, Sudoku)
- extensions are optional

An Extension - Sudoku



Schedule

- code and narrative finalized at APCS Development Committee winter meeting

- "Fresh Eyes" review Spring 2006
  - sign up at `http://gridworld.info`
- open to the public Fall 2006
- first used in 2008 exam

- as always, schedule may change...
  - watch AP mailing list, AP Central for announcements